

И. Я. АКУШСКИЙ,
В. М. ТРОЯНОВСКИЙ

ПРОГРАММИРОВАНИЕ

НА «ЭЛЕКТРОНИКЕ-100»

ДЛЯ ЗАДАЧ АСУ ТП



«СОВЕТСКОЕ РАДИО» 1978

Акушский И.Я., Трояновский В.М. Программирование на «Электронике-100» для задач АСУ ТП. —М.: «Сов. радио», 1978, 296 с.

Книга посвящена описанию малой ЭВМ, обладающей большим быстродействием (300 тыс. сложений/с при интегральном исполнении), гибкой системой прерывания и развитой системой периферийных устройств. Программирование иллюстрируется примерами, взятыми из практики разработки математического обеспечения АСУ в промышленности. Рассматриваются такие вопросы построения алгоритмического обеспечения и отработки программ систем управления технологическими процессами с использованием «Электроники-100», где машина и программирование выступают в роли надёжного инструмента для достижения поставленных целей. Приводится программа-диспетчер, используемая для организации программ управления в реальном времени.

Книга может быть полезна для разработчиков математического обеспечения систем управления, аспирантов и студентов старших курсов, специализирующихся в области автоматического управления и применения вычислительной техники.

Рис. 107, табл. 35, библи. 43 назв.

Редакция кибернетической литературы

Издательство «Советское радио», 1978 г.

ПРЕДИСЛОВИЕ

В решениях Партии и Правительства постоянное внимание уделяется росту производства электронно-вычислительной техники и внедрению ЭВМ в сферу управления экономикой и производством. Применение ЭВМ в задачах управления очень специфично как в части используемой техники, так и в алгоритмическом и программном обеспечении. Появление нескольких типов отечественных малых машин и широкое развертывание работ по созданию автоматизированных систем управления технологическими процессами (АСУ ТП) во всех отраслях промышленности настоятельно требуют разработки целого комплекса методологических вопросов как по программированию на новых машинах, так и по их применению в различных системах. Накопленный опыт весьма разнороден и нуждается в анализе и обобщении.

Целью данной книги является попытка восполнить в некоторой мере этот пробел. Детальное описание вопросов программирования и практической работы на ЭВМ «Электроника-100»¹⁾ имеет также целью создание более или менее систематизированного руководства по программированию на этих машинах; круг практических приложений ограничен вопросами применения машин в системах, которые непосредственно влияют на разработку программ для АСУ ТП. В соответствии с этим книга содержит две части; основой для её написания послужили результаты разработок математического обеспечения ряда систем на базе «Электроники-100», проведенных под руководством и при участии авторов; материал второго раздела во многом использует результаты разработки алгоритмического обеспечения систем управления, проведенных ранее совместно с д-ром техн. наук Ю.М. Быковым.

В создании нестандартного программного обеспечения для машины принимали участие различные коллективы и специалисты (стандартное, или как ещё называют, штатное математическое обеспечение для «Электроники 100» поступает к потребителям вместе с каждым экземпляром машины). Ряд полезных для практики сервисных программ разработан под руководством и при участии И.А. Моисеева и Р.Н. Тимониной, большую работу в этом направлении провёл А.С. Жданов. Отладка первой программы-диспетчера для «Электроники-100» проводилась при активном участии Л.Н. Маровой, идеи построения программы-диспетчера для ЭВМ с расширенным объёмом МОЗУ обсуждались с Ю.Л. Черняковым, в отладке этой программы принимала участие Е.Н. Иткина. Полезные контакты авторы имели с В.И. Панфиловым, В.В. Плотниковым, А.П. Ереминым, В.И. Горячкиным, Н.М. Дюковым, В.В. Каревым и их сотрудниками, обсуждение общих идей программного и языкового обеспечения «Электроники-100» проводилось с Ю.М. Петровым, И.А. Моисеевым, А.Г. Георгиевским, Ю.Н. Котельниковым и А.П. Петренко.

Авторы выражают благодарность за помощь и внимание, оказанные им в процессе совместных работ со стороны названных товарищей, разработчиков и изготовителей машины и её периферийных устройств.

Авторы признательны Б.П. Кузьмину и В.В. Липаеву за ценные замечания, способствовавшие улучшению содержания книги.

С благодарностью будут приняты все замечания, которые следует направлять по адресу: Москва, Главпочтамт, а/я 693.

Часть первая

ПРОГРАММИРОВАНИЕ НА «ЭЛЕКТРОНИКЕ-100»

В разработке и использовании ЭВМ отчетливо наметилась тенденция перехода к малым машинам. Применение малогабаритных и быстродействующих элементов, обычно на микроэлектронной базе, малая длина машинного слова (начиная с 8 бит), минимальный аппаратный комплекс для реализации арифметических и логических операций, широкий набор внешних устройств, наличие одного или нескольких каналов прерывания — вот основные черты, обеспечивающие низкую стоимость и возможность значительного и разнообразного расширения возможностей вычислительных, информационных и управляющих систем на базе мини-ЭВМ. За счёт развитого математического обеспечения и высокого быстродействия процессора мини-машины по своим вычислительным характеристикам приближаются к современным средним ЭВМ, а использование трансляторов в мнемонических и алгоритмических языках позволяет реализовать на этих машинах накопленные библиотеки стандартных и специальных программ.

Одной из основополагающих идей перехода к мини-машинам является микропрограммирование (выполнение программ на уровне машинных команд состоит из отдельных простейших микроопераций, причем число таких операций может быть весьма ограничено). За счёт уменьшения длины разрядной сетки, ограничения числа операций и их предельного упрощения оказывается возможным максимально уменьшить аппаратную часть ЭВМ, одновременно повысив её быстродействие. Однако осуществление любых сколь-нибудь сложных функций

теперь достигается обязательно с помощью специальных программ, составленных из микроопераций. Таким образом, центр тяжести в разработках систем на базе мини-машин переходит к вопросам их алгоритмического и программного обеспечения, к созданию эффективных приемов, подпрограмм, программных блоков и систем программирования, опирающихся на исходную систему команд и реализующихся, в конечном счете, на уровне микроопераций конкретной машины.

Для дополнительного повышения быстродействия и уменьшения общего числа команд в программах в мини-ЭВМ обычно предусматривается возможность объединения нескольких микрокоманд непротиворечивого логического смысла в рамках одной команды, выполняемой, как и обычная микрокоманда, в течение одного машинного цикла. Такой прием получил название микропрограммирования.

Среди разнообразных применений малых машин одним из наиболее перспективных является их использование для управления технологическими процессами [1]; полученные здесь решения могут быть легко распространены и на другие системы, работающие в реальном времени. Именно в этой части малые машины удачно заполняют имевшиеся ранее пробелы в применении вычислительной техники, как на производстве, так и для автоматизации научных экспериментов. Но решение таких задач все ещё требует заметных усилий в разработке математического обеспечения. По зарубежной статистике 10 лет, назад лишь один программист из 200 занимался решением задач, обеспечивающих работу в реальном масштабе времени, теперь это соотношение составляет 1 : 20 и продолжает расти.

В данном разделе рассматриваются общие вопросы программирования на ЭВМ «Электроника -100», с которыми программисту приходится сталкиваться при решении любого круга задач (вопросы специальных применений машины см. в ч. 2).

При изложении материала приходится пользоваться различными системами счисления. Всюду, где это не оговорено, используется десятичная система, в иных случаях нижний индекс после числа указывает основание системы счисления, например:

2376₈ — код числа с восьмеричным основанием или восьмеричное число;
101₂ — двоичное число.

Значительное внимание уделяется связности материала отдельных глав и параграфов, поэтому некоторые сведения и формулы, особенно необходимые по ходу изложения, иногда повторяются. В тексте встречаются следующие сокращения наиболее употребительных наименований.

АСУ — автоматизированная система управления
АСУП — автоматизированная система управления производством
АСУ ТП — автоматизированная система управления технологическими процессами
АЦП — аналого-цифровой преобразователь
БП — (команда) безусловной передачи (управления)
БПЗ — (программный) блок плавающей запятой
БУК — блок управления коммутаторами
ВВУ — вводные и выводные устройства
ВКДС — входной коммутатор дискретных сигналов
ВУ — (команда обращения) к внешним устройствам
ДАБ — добавляемый арифметический блок
ДЗУ — долговременное запоминающее устройство
ДР (ДРС) — дополнительный разряд (дополнительный разряд сумматора)
КЛ — клавиатура (режим ввода с клавиатуры пишущей машинки)
КР — клавишный регистр
ЛУ — (команда) логического умножения
МК — микрокоманды
МОЗУ — магнитное оперативное запоминающее устройство
МЧ — (регистр) множителя-частного (в добавляемом арифметическом блоке)
НР (СМ) — накопительный регистр (сумматор)
ОЗУ — оперативное запоминающее устройство
ОП — (команда) обращения к подпрограмме
ОС — отсчет слов (в режиме «Разрыв данными»)
ПЗУ — постоянное запоминающее устройство
ПР — перфорация (режим вывода)
ПС — (команда) посылки кода (из накопительного регистра (сумматора) ячейку по указанному адресу и сброса (накопительного регистра))
ПЧ — печать (режим вывода на пишущую машинку)
РА — регистр адреса
РБ — буферный регистр
РГВВ (РГВЫВ) — регистр ввода (регистр вывода)
РЗП — регистр запоминания полей (поля данных и поля команд при работе с расширенной памятью)
РИ — регистр информации в БУКе
СК — счетчик команд
СЛ — (команда) сложения
СЧ — считыватель (режим ввода с перфоленты)
СЧШ (СТ) — счетчик шагов или счетчик тактов (в добавляемом арифметическом блоке)
УВВУ — блок управления вводными—выводными устройствами
УП — (команда) увеличение (содержимого ячейки по указанному адресу на

¹ Предполагается, что читатель знаком с основами цифровой вычислительной техники (см. [2, 3]).

единицу младшего разряда) и пропуск (следующей команды, если новое содержимое ячейки равно нулю)

УЦВМ — управляющая цифровая вычислительная машина
ЦАП — цифроаналоговый преобразователь
ЦВМ — цифровая вычислительная машина
ЭВМ — электронная вычислительная машина
ЭПМ — электрифицированная пишущая машинка ЭЦВМ — электронная цифровая вычислительная машина.

1 ХАРАКТЕРИСТИКА МАШИНЫ «ЭЛЕКТРОНИКА-100»

1.1. УСТРОЙСТВО И ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ МАШИНЫ

«Электроника-100» представляет собой малую быстродействующую одноадресную вычислительную машину с двоичной системой счисления. Машина оперирует 12-разрядными двоичными числами с фиксированной запятой и имеет всего 8 основных команд (общее число команд превышает 70). Минимальный комплект машины включает процессор с магнитным оперативным запоминающим устройством (МОЗУ) на 4096 слов, устройство ввода с перфоленты типа СП-3, выходной перфоратор ПЛ-150, печатающую машинку «Consul-254», а также устройство управления вводом — выводом и блок питания к нему. Основные технические характеристики устройств приведены в табл. 1.1.

Машина имеет два режима прерывания: программно обнаруживаемое (или программное) прерывание и режим разрыва данными с прямым доступом в память. В первом случае по сигналу от внешнего устройства выполнение любой программы может прерываться с запоминанием места прерывания, и управление передается в фиксированное место памяти, где должна быть размещена специальная программа обслуживания прерываний. Во втором случае по сигналу от внешнего устройства выполняемая программа приостанавливается на время, необходимое для автоматического обмена кодами (одним кодом или массивом) между внешним устройством и некоторым участком МОЗУ, после чего выполнение приостановленной программы продолжается.

Оба режима обслуживают соответствующие аппаратные блоки. Необходимые команды и работа с использованием систем прерывания будут описаны в гл. 3, 4.

Таблица 1.1

Основные технические характеристики машины

Система счисления	двоичная
Способ представления чисел	с фиксированной запятой
Система команд	одноадресная
Число основных команд	8
Емкость МОЗУ (минимального комплекта)	4096 ячеек по 12 двоичных разрядов
Скорость выполнения операций ¹	
логических	500 000 операций/с
сложения (с прямым адресованием)	250 000 операций/с
команд обращения к памяти с косвенной адресацией	167 000 операций/с
Число режимов прерывания	2
Максимальная скорость вывода и ввода с помощью пишущей машинки	8 символов/с
Минимальный комплект машины	процессор, считыватель с перфоленты, перфоратор, пишущая машинка, блок управления и блок питания устройств (ами) ввода—вывода
Носитель внешней информации	бумажная перфолента 8 дорожек (25,4 мм)
Скорость ввода с перфоленты	150 строк/с
Скорость вывода на перфоратор	150 строк/с
Питание машины	однофазная сеть переменного тока 220 В (+10% -15%), 50 Гц
Потребляемая мощность	500 В·А
Масса машины	100 кг
Габариты процессора	570×545×860

Два названных режима прерывания обеспечивают большую гибкость общения машины с внешним миром. Помимо возможности совмещения вычислений с работой устройства ввода—вывода, система прерывания необходима для построения систем управления в реальном времени, организации мультипрограммной работы, построения многомашинных комплексов и других подобных задач, где запросы к машине (процессору) могут поступать в произвольные моменты времени, а задержки в их обслуживании должны быть минимальными. С использованием прерывания строятся и системы приоритетного обслуживания.

С точки зрения программиста-пользователя в общей архитектуре машины выделяются быстродействующий процессор со своим пультом управления, блоки ввода—вывода со своим устройством управления, вмонтированные в машину блоки обслуживания прерываний. По быстродействию обмена с внешними устройствами на первом месте стоит режим разрыва данными (2

¹ При введении добавляемого арифметического блока (ДАБ): умножение с обычной точностью—10 мкс, деление с обычной точностью — 11 мкс

мкс/слово при обмене отдельными словами и 6 мкс/слово при обмене массивом)¹, затем вывод информации на перфоратор (150 строк/с, что эквивалентно 75 слов/с ввиду малой разрядности перфоленты) и ввод с перфоленты (150 строк/с) и, наконец, ввод—вывод информации с помощью ЭПМ «Консул» (8 символов/с).

Программисту для работы нужны следующие сведения об устройстве машины.

Процессор содержит сумматор, или накопительный регистр)², дополнительный разряд сумматора, буферный регистр, регистр адреса, счетчик команд, клавишный регистр и ряд клавиш для управления работой машины («Пуск», «Стоп», «Цикл» и другие — подробное описание их назначения и работы с ними будет дано в гл. 5).

Дополнительный разряд сумматора (ДРС) представляет собой одноразрядный регистр, все остальные регистры имеют по 12 разрядов. Сумматор (СМ), или накопительный регистр (НР) является основным регистром, используемым при арифметических и логических действиях и для промежуточного хранения результатов; через сумматор программным путем осуществляется загрузка кодов в ячейки памяти, связь с устройствами ввода — вывода и периферийным оборудованием. При арифметических и логических операциях НР (СМ) и ДР (ДРС) образуют 13-разрядный регистр: при арифметических операциях в ДР (ДРС) попадают единицы переполнения НР (единицы переполнения ДР теряются); в логических операциях сдвига участвует весь 13-разрядный код, заполняющий НР (СМ) и ДР (ДРС). Загрузка кодов в ячейки памяти осуществляется только из 12 разрядов НР (СМ); содержимое ДР (ДРС) при этом остается неизменным. Сумматор и ДРС программно доступны для анализа их содержимого, а также для изменения этого содержимого.

Буферный регистр (РБ) служит для временного хранения информации, вызываемой из оперативной памяти и записываемой в оперативную память. Он используется также при арифметических действиях, при формировании промежуточных данных внутри сложных машинных циклов, при работе аналого-цифрового преобразователя и реализации отдельных команд. Содержимое РБ программно недоступно для анализа и изменения, однако РБ имеет световую индикацию на передней панели процессора, что существенно облегчает контроль прохождения отдельных операций при отладке программ.

Регистр адреса (РА) используется для формирования абсолютного адреса ячейки МОЗУ, содержимое которой вызывается в РБ или в которую записывается содержимое РБ. Как и РБ, регистр адреса является программно недоступным, но имеет световую индикацию на передней панели процессора, что полезно при отладке. Счетчик команд (СК) своим содержимым предопределяет абсолютный адрес ячейки МОЗУ, куда будет передано управление. Этот регистр, таким образом, по своему назначению стоит несколько в стороне от других названных выше регистров — как и любая другая машина, «Электроника -100» не различает самостоятельно, является ли содержимое очередной ячейки командой или числовым материалом, и содержимое каждой ячейки, куда передано управление, интерпретируется как команда. Поэтому счетчик команд помогает программисту «удерживать» машину в тех участках МОЗУ, где расположена программа, и организовывать определенный порядок выполнения команд. Содержимое СК может изменяться и использоваться программным путем.

Клавишный регистр (КР) может использоваться для занесения вручную или по программному опросу различной информации (в виде 12-разрядных кодов) в машину. Вручную его содержимое может быть занесено в РБ (и затем немедленно в ячейку МОЗУ) или СК; при программном обращении содержимое КР может взаимодействовать с содержимым СМ (см. гл. 3). В настоящее время (до появления в составе машины постоянного запоминающего устройства) клавишный регистр является единственным средством, позволяющим занести в память машины минимальную исходную программу ввода. С помощью КР осуществляется занесение исходного адреса программы в СК перед пуском. Однако чаще всего КР используется при отладках программ и для задания с пульта различных вариантов при решении отложенных задач.

Помимо названных блоков процессор содержит устройство управления. Программисту важно знать, что благодаря входящему в него генератору главных состояний выполнение отдельных команд разбивается на один, два или три машинных цикла. Содержимое основных регистров машины после выполнения каждого цикла может быть доступно для визуального контроля благодаря наличию соответствующих режимов неавтоматической работы машины (см. гл. 5).

Внешние устройства, или устройства ввода — вывода (считыватель с перфоленты, перфоратор, печатающая машинка)³ имеют свой блок управления, позволяющий всем этим устройствам работать от ЭВМ или в автономном режиме. В последнем случае обычно работают попарно устройство ввода и вывода, чем обеспечивается

¹ Не считая времени, затрачиваемого на входение в режим прерывания

² Терминология здесь во многом ещё не установилась. Поэтому далее вначале будут указываться наименования, рекомендуемые для дальнейшего использования, а рядом — наименования и сокращения, применяемые во всей ранней документации

³ В этом параграфе имеется в виду минимальный комплект машины.

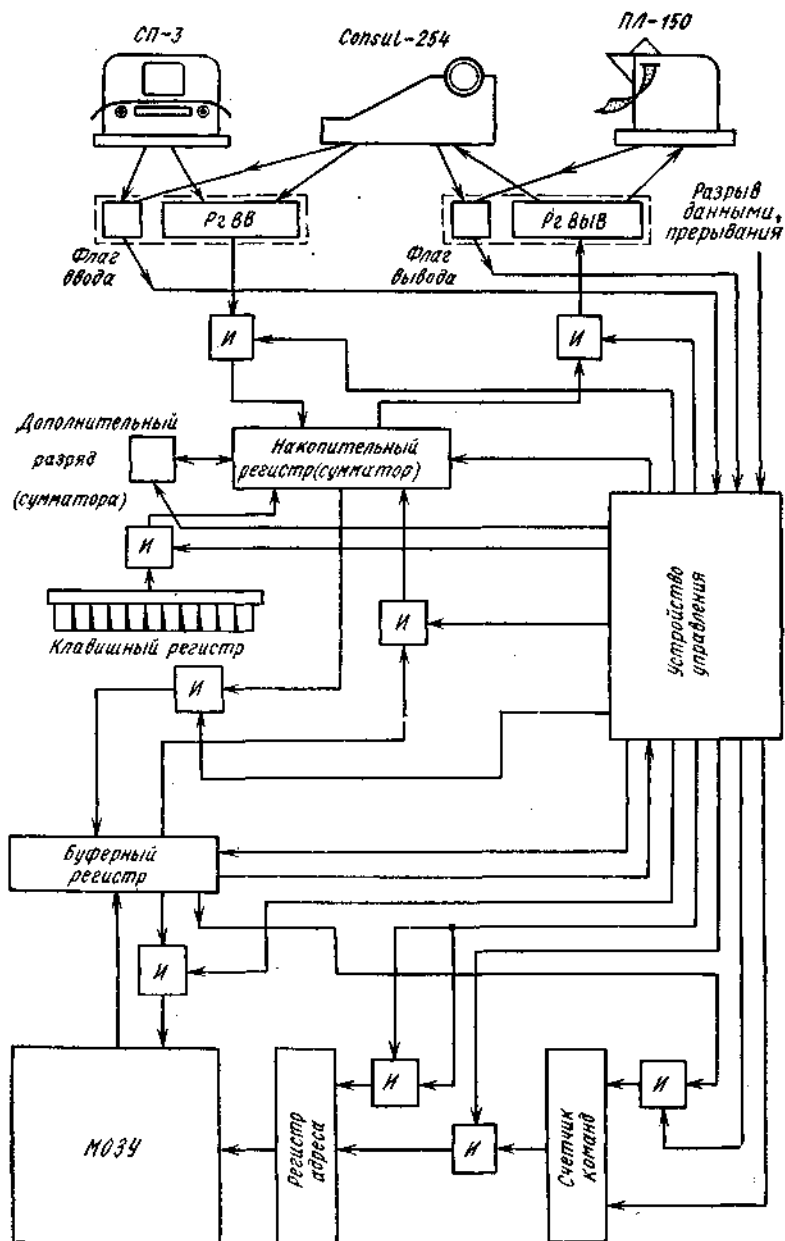


Рис. 1.1. Увеличенная структурная схема машины.

возможность дублирования и подготовки перфолент без дополнительного оборудования.

С точки зрения программиста, каждое внешнее устройство имеет свой номер и два характерных устройства: буферный регистр для числовой информации и так называемый «флаг» — устройство (триггер) для сигнализации о готовности внешнего устройства к новому взаимодействию с (более быстродействующим) процессором. Различают буферный регистр ввода (РгВВ) и буферный регистр вывода (РгВВВ). Оба регистра имеют 8 разрядов. Регистр

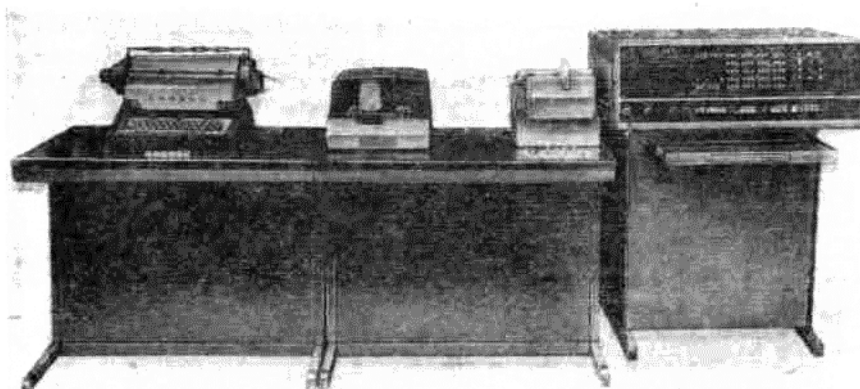


Рис. 1.2. Внешний вид минимального комплекта ЭВМ «Электроника-100И».

ввода является общим для всех устройств ввода; регистр вывода — общим для всех устройств вывода.

На РГВВ попадает информация со считывателя или при вводе с «Консула», одновременно «флаг» ввода устанавливается в единицу, сигнализируя о готовности информации, которая может быть по специальным командам переписана из РГВВ в НР (СМ). На РГВЫВ информация выводится из процессора (из НР), и её дальнейшее движение (печать или перфорация) определяется положением переключателя на передней панели блока управления вводными — выводными устройствами (ВВУ). После этого флаг вывода устанавливается в единицу только тогда, когда «Консул» или перфоратор готовы вновь принять (и отработать) очередной код.

Состояние флагов внешних устройств доступно программному контролю; флаг внешнего устройства, установленный в единицу, может приводить к прерыванию. Флаги внешних устройств и регистр ввода имеют световую индикацию на передней панели блока ВВУ.

Конструктивно все внешние устройства и блок управления ими, как и процессор, выполнены в виде настольных приборов, соединяемых между собой электрическими кабелями.

Укрупненная структурная схема машины приведена на рис. 1.1, внешний вид минимального комплекта машины показан на рис. 1.2.

1.2. ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА

При работе машины в системе управления важное значение приобретают периферийные устройства, через которые осуществляются её связь с внешним миром, общая синхронизация работ, перевод информации в нужную форму и т. п. В настоящее время системы управления на базе ЭВМ «Электроника-100» могут комплектоваться следующими периферийными устройствами;

- 1) таймер,
- 2) входной коммутатор аналоговых сигналов,
- 3) аналого-цифровой преобразователь,
- 4) входной коммутатор дискретных сигналов,
- 5) преобразователь для ввода частотных сигналов,
- 6) выходной коммутатор дискретных сигналов,
- 7) цифроаналоговый преобразователь,
- 8) блок управления шаговыми двигателями,
- 9) блок управления световым табло.

С точки зрения программного использования периферийные устройства можно представить как развитие (разновидность) внешних устройств машины. Некоторые особенности обращения к ним будут рассмотрены в гл. 3. Связаны эти особенности, как правило, с наличием дополнительного оборудования (в частности, регистров и ключей) в составе отдельных устройств.

Конструктивно часть периферийных устройств раннего выпуска располагается в так называемом центральном посте, другая часть выполнена в виде отдельных блоков.

Таймер или датчик точного времени содержит регистр-счетчик и генератор тактовых импульсов, поступающих на вход счетчика через схему управления. При заполнении счетчика вырабатывается сигнал, используемый схемой управления для остановки счета и сообщения в процессор об этом событии.

Выбор диапазона, занесение исходного кода в регистр таймера и его пуск могут осуществляться вручную (с его собственной клавиатуры) или автоматически (по командам процессора). По окончании счета таймер устанавливает свой флаг в единичное состояние и может вызывать прерывание вычислений в процессоре. Основным режимом работы таймера является автоматический, с использованием режима прерывания процессора. Располагается таймер в центральном посте.

Входной коммутатор аналоговых сигналов представляет собой блок программно управляемых транзисторных ключей, объединенных в 8 групп по 64 ключа в каждой. Одна группа расположена в центральном посте, семь других — выносные и подключаются к центральному посту с помощью кабелей. Благодаря применению бесконтактных ключей быстродействие коммутатора достигает 20 кГц (ключ переходит из закрытого состояния в открытое или обратно в течение 50 мкс). Выход коммутатора подключается к аналого-цифровому преобразователю.

Аналого-цифровой преобразователь (АЦП) служит для приведения исходной аналоговой информации, поступающей от источника непрерывных сигналов, к дискретной форме, пригодной для дальнейшего использования в ЦВМ. Схема управления производит поразрядное включение (набор) кода и в зависимости от знака усиливаемого сигнала рассогласования аналоговых сигналов (преобразуемого и устанавливающегося на выходе ЦАП) оставляет включенным очередной разряд кода или отключает его. В работе АЦП принимает участие буферный регистр; выходной 10-разрядный код набирается в старших разрядах НР(СМ).

Все блоки АЦП смонтированы в корпусе процессора в непосредственной близости от НР(СМ) и РБ; время преобразования составляет около 50 мкс, программное обращение к АЦП укладывается в одну команду.

Входной коммутатор дискретных сигналов представляет собой набор программно управляемых многопозиционных (12- и 6-разрядных) ключей, с помощью которых сигналы от источников дискретных сообщений коммутируются ко входу НР(СМ). Поскольку сигнал в каждом разряде может принимать только два значения (0 или —3,7 В, что соответствует 0 или 1 в соответствующем разряде), собственное быстродействие коммутатора дискретных сигналов существенно выше, чем у коммутатора аналоговых сигналов.

Коммутаторы входных сигналов могут иметь свои флаги для централизованной сигнализации о необходимости экстренного опроса определенной группы входов (датчиков).

Преобразователь для ввода частотных сигналов обеспечивает измерение длительности стандартного числа периодов входного сигнала. При диапазоне изменения частоты 4—8 кГц максимальное число счетных импульсов, накапливающееся в НР(СМ), не выходит за 10 разрядов. Время преобразования (измерения) не превышает 2 мс.

Выходной коммутатор дискретных сигналов имеет в своем составе промежуточный 12-разрядный регистр и 32 12-позиционных ключа. Выходная информация из НР(СМ) передается в промежуточный регистр, после чего открывается группа ключей на нужное направление (к нужному потребителю).

Потребитель получает 12-разрядный код, занесенный в промежуточный регистр, все время, пока открыты «его» ключи. Если потребитель не имеет (аппаратурных) возможностей для запоминания кода за время, соизмеримое с машинным тактом (2 мкс), то коммутатор оказывается занятым на все время, необходимое для обслуживания данного абонента. Вопросы организации программ с учётом этого обстоятельства рассмотрены в ч. 2.

Цифроаналоговый преобразователь (ЦАП) совершенно аналогичен используемому в АЦП, но имеет свой (промежуточный) регистр, куда из НР(СМ) заносится выходной преобразуемый код. Сигнал на выходе ЦАП может изменяться в пределах 0—10 В «ступенями» по $10 \cdot 2^{-10}$ В (чуть менее 10 мВ).

Набор периферийных устройств определяется при компоновке системы. Тогда же решаются вопросы и общего конструктивного оформления отдельных блоков. Одновременно для алгоритмиста и программиста очерчивается круг обязательных блоков программы по обслуживанию включенных в систему периферийных и внешних устройств.

1.3. ПЕРСПЕКТИВЫ РАЗВИТИЯ

Разработку и развитие ЭВМ «Электроника-100» нельзя рассматривать как полностью завершённые. Это относится как к блокам и узлам процессора, так и к внешним и периферийным устройствам ЭВМ.

Развитие процессора предусматривает возможность ускоренной аппаратурной реализации операций умножения, деления, сдвигов и нормализации за счет подключения добавляемого арифметического блока (ДАБ), введение аппаратного контроля по нечётности всех используемых кодов, увеличение объёма оперативной памяти до 8 кубов, появление постоянного запоминающего устройства (ПЗУ), а также увеличение объёма долговременной памяти за счет подключения стандартного накопителя на магнитной ленте (типа НМЛ-67).

Умножение двух чисел, записанных в стандартной форме, выполняется при использовании ДАБ за 10 мкс, а для чисел, занимающих по две ячейки каждое (см. гл. 2) — за 68 мкс; деление выполняется соответственно за 11 и 84 мкс.

Разрядная сетка машины будет дополнена служебным 13-м разрядом (в распоряжении программиста по-прежнему остаются 12-разрядные коды для чисел и команд). При записи 12-разрядных кодов в машину в этот служебный разряд будет автоматически заноситься 0 или 1 так, чтобы общее число единиц в 13-разрядном слове оказалось нечётным, и в таком виде 13-разрядные слова будут храниться в памяти. В процессе работы каждый вызываемый из МОЗУ код будет аппаратно проверяться на нечётность, с выходом на останов или прерывание в случае сбоев.

МОЗУ можно будет наращивать кубами по 4096 слов (по 4К)¹ до общего объёма 32К. Для «Электроники-100И» каждая пара кубов памяти будет содержать свой блок управления, «Электроника-100» имеет один блок управления для всей расширенной памяти.

ПЗУ позволит постоянно хранить в памяти набор сервисных и стандартных программ, а также «зашивать» отлаженные участки программы, за счет чего повысится общая надёжность работы машины. Объём ПЗУ—до 8192 слов, из которых 256 выполнены в виде ОЗУ для рабочих ячеек и т. п.

Стандартный накопитель на магнитной ленте типа НМЛ-67 позволит иметь во внешней долговременной памяти 10^6 — 10^7 слов на каждой кассете ленты. Устройство управления накопителем использует для связи с ЭВМ быстродействующий канал разрыва данными; оно может применяться для подключения восьми НМЛ.

Внешние и периферийные устройства машины пополнятся графопостроителем типа ЦГП-1, блоками сопряжения двух ЭВМ «Электроника-100», а также блоком сопряжения машин «Электроника 100» и «Минск-32»; блоками управления световым табло и цифровой индикации, блоком управления шаговыми двигателями. На выходе ЦАП появится блок фиксации уровня (аналогового сигнала), часть каналов выходного коммутатора дискретных сигналов будет также снабжена фиксирующими буферными регистрами. Перерабатывается структура управления всеми коммутаторами сигналов с целью обеспечения экономного блочного комплектования систем в соответствии с их конкретной конфигурацией. Предусматривается подключение к «Электронике-100» дисплея, накопителя на магнитных дисках, а также блока сопряжения с системой КАМАК.

Важно отметить, что все периферийные устройства (как уже выпускаемые, так и проектируемые) используют ту же элементную базу, что и сама ЭВМ—в виде её стандартных плат, из которых и набираются нужные схемы периферийных устройств.

Разработанный комплект устройств позволяет потребителям получать сразу не только минимальный комплект ЭВМ, но и целые комплексы аппаратуры различного назначения. Заводом-изготовителем предлагаются, в частности, типовые наборы ЭВМ и периферии для большого и малого управляющего комплекса, для большого и малого вычислительного комплекса, для информационных комплексов, а также ряд дополнительных устройств, с помощью которых возможности этих комплексов могут быть развиты в нужном направлении.²

Следует отметить также, что многие разработчики сложных машинных систем заранее ориентируются на применение «Электроники-100». Так, специализированный комплекс аппаратуры для ядерных исследований «Вектор» имеет в своем составе блок для сопряжения с «Электроникой-100».

¹ К=1024 байта

² См., в частности, «Электронная промышленность», 1975, № 1. 16

2. РАЗРЯДНАЯ СЕТКА МАШИНЫ И ЕЁ ИСПОЛЬЗОВАНИЕ

2.1. ОБЩИЕ СВЕДЕНИЯ

Большинство основных функциональных узлов машины (ячейки памяти, регистры) имеют двенадцать двоичных разрядов, что, естественно, связано с выбранной длиной машинного слова. Наряду с этим в ячейках памяти могут храниться команды и числа, константы и абсолютные адреса. Они имеют разное назначение и по-разному используются в машине, хотя, как правило, занимают одинаковые отдельные ячейки памяти.

Логика построения и работы отдельных блоков и функциональных узлов машины накладывает определенные ограничения уже на саму структуру использования разрядной сетки машины при представлении и хранении различной информации.

В дальнейшем для обозначения поразрядного содержимого каждого отдельного элемента машины независимо от его функционального назначения (ячейка памяти или регистр) будет использоваться условное обозначение рис. 2.1, где каждая клеточка соответствует определенному разряду ячейки памяти или регистра. Как видно из рисунка, принята нумерация разрядов слева направо, начиная с 0-го (старшего) и заканчивая 11-м (младшим). Каждый разряд может принимать только два значения — нуль или единица, образуя, таким образом, 12-разрядный двоичный код. Для краткости записи такие коды часто представляют в восьмеричной форме (или восьмеричном виде), записывая (кодируя) одной цифрой (от 0 до 7 включительно) содержимое трех двоичных разрядов (триад). Так, в примере на рис. 2.1 ячейка памяти (или регистр) содержит код 101110010010_2 или 5622_8 (способ



Рис. 2.1. Разрядная сетка ЭВМ «Электроника-100».

разбивки разрядной сетки на триады показан на рис. 2.1 внутренними рамками).

Одни и те же разряды при представлении различной информации имеют разное назначение, образуя всего четыре основные структуры.

2.2. ЧЕТЫРЕ СТРУКТУРЫ

Различие в структурах заполнения информацией разрядной сетки машины определяется, прежде всего, различным характером числовых данных и команд, а также особенностями различных типов команд.

Для представления чисел, записываемых в одну ячейку, необходимо определить в ней местоположение мантиссы и знака числа (как уже отмечалось, числа в машине имеют представление с фиксированной запятой и

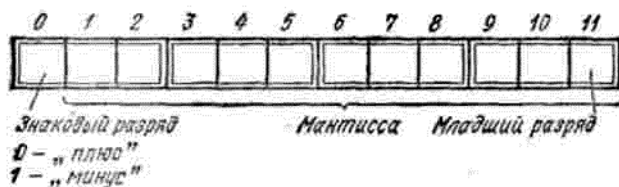


Рис. 2.2. Структура представления чисел в машине.

поэтому места для указания порядка числа не требуется). Именно такая структура для представления чисел и используется в «Электронике-100» (рис. 2.2): нулевой разряд разрядной сетки—для указания знака числа, разряды с 1-го по 11-й — для мантиссы.

Вторую характерную структуру заполнения разрядной сетки машины образуют команды обращения к памяти¹.

В такой команде необходимо указать код команды, а также адрес ячейки памяти, которая участвует в данной операции. При этом проявляется ограничение, связанное с малой разрядностью машины: для указания абсолютного адреса любой из ячеек МОЗУ в пределах куба памяти (с 0-й по 4095-ю ячейку) требуются все 12 разрядов и поэтому при использовании абсолютного адреса в существующей разрядной сетке негде поместить номер команды. В «Электронике-100» это противоречие разрешено путем страничной организации памяти и применением прямого и косвенного адресования. Для указания любого

¹ Подробно система команд разбирается в гл. 3. Здесь важен уже отмечавшийся факт, что число основных команд «Электроники-100» равно восьми, так что номер любой команды кодируется в одной триаде. Для этого в разрядной сетке отводится старшая триада (разряды 0—2), что легко проследить во всех структурах использования разрядной сетки для представления команд.

адреса на странице размером в 128 ячеек требуется 7 двоичных разрядов, 3 разряда требуются для кодирования номера команды и 2 оставшихся (до 12) разряда используются под признаки прямого или косвенного адресования и обращения к текущей или нулевой странице памяти (структура команд обращения к памяти представлена на рис. 2.3). Команды обращения к памяти имеют коды операции (номер команды) с 0-го по 5-й включительно; их работа, особенно в части формирования и использования адресов, является наиболее специфичной особенностью «Электроники-100». Подробно эти вопросы рассматриваются в гл. 3.

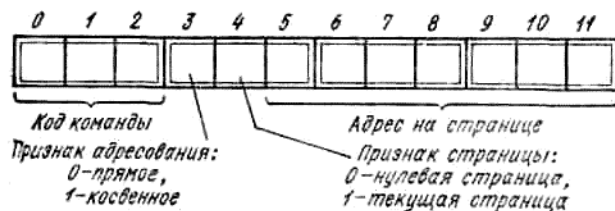


Рис. 2.3. Структура команд обращения к памяти.



Рис. 2.4. Структура команд обращения к внешним устройствам.

Третью структуру образуют команды обращения к внешним устройствам (рис. 2.4). В старшей триаде кодируется номер операции (б₈), две следующие триады отведены под код (номер) внешнего устройства и в младшей триаде записывается код выполняемого взаимодействия с данным внешним устройством. Обычно в младшей триаде каждый разряд «закреплен» за определенным действием¹ по опросу или очистке флага и обмену кодами между СМ и регистром внешнего устройства (подробно см. гл. 3). Часто допускается одновременное использование нескольких разрядов младшей триады, что приводит к выполнению, в определенной последовательности, всех запрашиваемых (указанных в команде) действий в течение одного такта работы процессора. Такой прием, существенно экономящий время процессора и объем программы, носит название *микропрограммирования*. Особенно характерно его использование внутри последней, четвертой структуры.

¹ Выполняемым при наличии «1» в соответствующем разряде

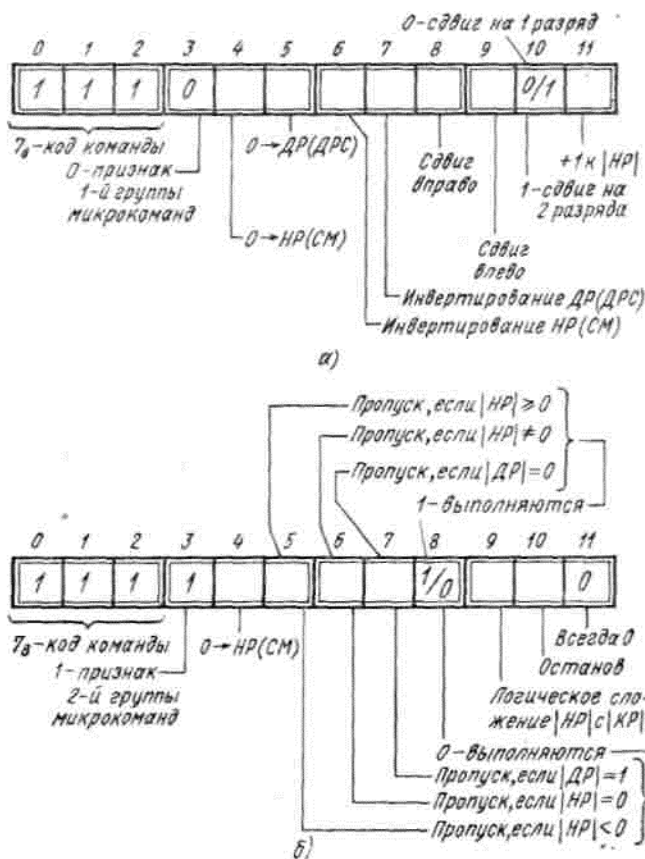


Рис. 2.5. Структура микрокоманд:
а — первой группы; б — второй группы.

Четвертую структуру образует группа микрокоманд (рис. 2.5). Код этих команд (7_8) записывается, как обычно, в старшей триаде. Третий разряд используется для указания одной из двух больших групп микрокоманд, первая из которых объединяет команды по изменению содержимого НР (СМ) и ДР (ДРС) (очистке, сдвигам, инвертированию, занесению определенных кодов и пр.), а вторая—в основном по анализу содержимого НР(СМ) и ДР (ДРС) и организации пропуска очередной команды при выполнении тех или иных условий. Разряды с 4-го по 11-й при этом соответствуют определенным действиям или условиям, и допускается возможность микропрограммирования, реализуемого одновременным использованием в команде нескольких из этих разрядов. Именно за счет микрокоманд наблюдается резкое увеличение общего числа команд «Электроники-100» по отношению к восьми основным командам.

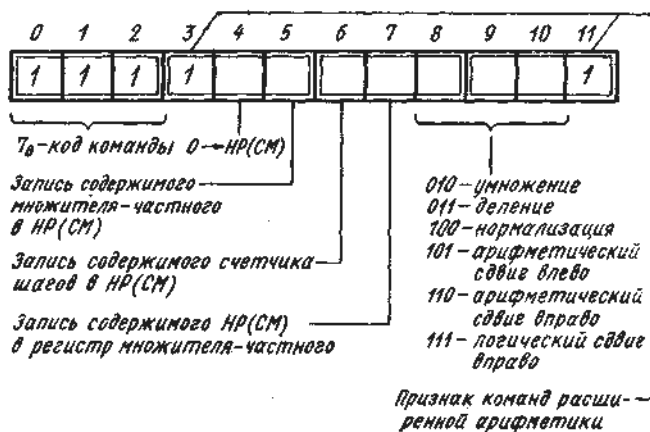


Рис. 2.6. Структура команд обращения к добавляемому арифметическому блоку.

Ко второй группе микрокоманд примыкают команды обращения к добавляемому арифметическому блоку (рис. 2.6).

Следует отметить, что для числовых данных и констант, не укладывающихся в стандартное представление, в программе могут использоваться и другие структуры произвольного характера. Однако в этих случаях программист должен постоянно иметь в виду, что основная система команд машины всегда жестко привязана к рассмотренным основным структурам, и поэтому работа с информацией иного формата обычно сопровождается использованием специальных подпрограмм.

В частности, для повышения точности представления данных иногда используется более чем 12-

разрядная сетка. В этом случае каждое число занимает уже не одну, а несколько (обычно последовательно расположенных) ячеек памяти. Если используется представление чисел с фиксированной запятой, то один (старший) разряд такого составного слова отводится под знак числа, а остальные разряды — под мантиссу. Если вводится структура представления чисел с плавающей запятой, то код в первой ячейке определяет порядок числа (в соответствии с обычной структурой представления чисел в машине), а в нескольких последующих ячейках располагается многоразрядная (более чем 12-разрядная) мантисса числа со знаком числа. Подробнее такие структуры описаны в гл. 4.

2.3. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В МАШИНЕ

Как уже упоминалось, «Электроника-100» оперирует с двоичными числами с фиксированной запятой. Структура использования разрядной сетки машины для представления таких чисел показана на рис. 2.2.

Имеются две особенности в представлении чисел. Одна из них связана с выбором диапазона изменения и достижимой точностью представления переменных; вторая — со способом представления отрицательных чисел.

Так как один (старший) разряд занимает под знак, двоичный код в оставшихся 11 разрядах может принимать $2^{11} = 2048$ значений. Соответственно на такое число ступеней разбивается (дискретизируется) и весь диапазон представления числовых данных. Обычно этот диапазон выбирают нормированным, а все переменные приводят к безразмерному виду¹, т. е. условливаются, что максимальное (по модулю) число равняется единице, и тогда единица младшего разряда (машинная единица) имеет вес $(2^{11} - 1)^{-1}$. Этой величиной (порядка 0,05%) и оценивается обычная относительная точность исходного представления данных в машине².

Отрицательные числа в машине записываются с использованием так называемого дополнительного кода. Правило его образования следующее: вначале от исходного кода образуют обратный (т. е. содержимое каждого разряда инвертируют: нули заменяют единицами, а единицы — нулями) и к обратному коду прибавляют единицу младшего разряда.

Формально это правило может быть записано следующим образом. Пусть исходный код для x есть

$$x = \{\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_{11}\},$$

где α_i — содержимое i -го разряда. Тогда обратный код определяется как $\bar{x} = \{\bar{\alpha}_0, \bar{\alpha}_1, \bar{\alpha}_2, \dots, \bar{\alpha}_{11}\}$, где

$$\bar{\alpha}_i = \begin{cases} 0, & \text{если } \alpha_i = 1, \\ 1, & \text{если } \alpha_i = 0, \end{cases}$$

а дополнительный код $x_{\text{доп}} = \bar{x} + 1_{\text{мл. разряда}} = \bar{x} + 0001_2$.

Пример. Исходное положительное число x представлено кодом 2511₂, требуется определить код числа $-x$:

$$\begin{array}{r} x = 010 \quad 101 \quad 001 \quad 001_2 \\ \quad \quad \quad \downarrow \downarrow \downarrow \downarrow \\ \text{Обратный код } \bar{x} = 101 \quad 010 \quad 110 \quad 110_2 \\ \quad \quad \quad + \\ \hline -x = x_{\text{доп}} = 101 \quad 010 \quad 110 \quad 111_2 = 5267_2 \end{array}$$

Нетрудно проследить, что для всех положительных чисел взаимно однозначно могут быть образованы соответствующие отрицательные, причем в нулевом разряде для положительных чисел всегда записывается нуль, а для отрицательных — единица (что и служит признаком для их разделения при анализе). Исключение составляют коды 0000₈ и 4000₈, для которых дополнительные коды совпадают с исходными. Для первого кода, обозначающего число нуль, это обстоятельство не имеет существенного значения, так как запись чисел +0 и —0 тождественна просто 0. Второй код, представляющий максимальное по модулю отрицательное число,

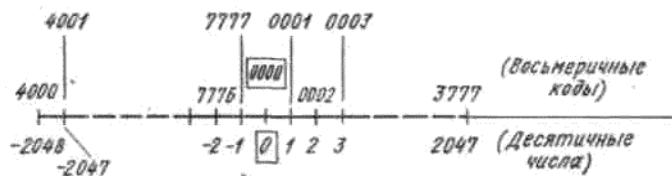


Рис. 2.7. Вид числовой оси и соответствующих кодов при вычислениях с обычной точностью.

не имеет соответствующего себе в силу ограничений разрядной сетки машины и принятой структуры

¹ О способе нормирования реальных исходных данных см. в гл. 4.

² При использовании двух ячеек памяти для одного числа точность представления возрастает до $0,15 \cdot 10^{-4}\%$, а при использовании трех ячеек памяти для одного слова 36-разрядная сетка предоставляет практически такие же возможности по точности представления данных, какими обладают средние ЭВМ.

представления чисел (максимальное по модулю положительное число имеет код 3777₈).

В результате числовая ось с нанесенными на нее кодами положительных и отрицательных чисел выглядит так, как показано на рис. 2.7. Справа и слева вблизи нуля числовой оси группируются малые по абсолютной величине числа, которые изображаются самыми малыми кодами для положительных чисел и самыми большими — для отрицательных. Рост модуля положительных чисел сопровождается ростом величины их кодов; рост модуля отрицательных чисел — уменьшением величины их кодов.

Принятый способ представления отрицательных чисел является следствием общей линии построения мини-машин, когда принимаются все меры для уменьшения аппаратной части машины. В частности, использование дополнительного кода для представления отрицательных чисел позволяет максимально упростить одну из основных операций — сложение. При сложении чисел, независимо от их знаков, производится сложение кодов.

Если, например, складывается положительное число с отрицательным, модуль результата меньше модуля любого из слагаемых. Однако, если результат отрицателен, его код должен быть больше 4000₈, а если положителен, то меньше 4000₈. Один из исходных кодов (соответствующий отрицательному числу) уже заранее больше 4000₈, и к этому коду добавляется ещё некоторый другой. Результат при этом, естественно, стремится переполнить разрядную сетку. Если переполнения не происходит, то в НР(СМ) в качестве результата накапливается большой по величине код, изображающий, таким образом, малое по модулю отрицательное число. Если же происходит переполнение разрядной сетки, то единица переполнения переходит из НР(СМ) в ДР(ДРС), а результат в виде малого по величине кода, оставшегося в НР(СМ), изображает малое положительное число.

Удобным mnemonicическим правилом образования дополнительного кода (а, следовательно, получения отрицательных чисел из положительных и наоборот) является следующее: дополнительный код — это такой, который дополняет исходный (при сложении) до переполнения разрядной сетки машины, т. е. до 10000₈. Это правило легко проверить на разобранном выше примере:

$$x + x_{\text{доп}} = \frac{2511_8 + 5267_8}{10000} = 1 \boxed{0000_8},$$

и в разрядной сетке машины результат равен нулю, что соответствует использованию дополнительного кода для представления отрицательного числа:

$$x + x_{\text{доп}} = x - x = 0.$$

Таким образом, для получения вручную² кода отрицательного числа при известном модуле достаточно вычесть этот модуль (по правилам восьмеричной арифметики) из кода 10000₈.

3. СИСТЕМА КОМАНД

Система команд ЭВМ является своеобразным сводом законов преобразования данных в машине на основе выполнения ею отдельных операций. Точное знание этих законов совершенно необходимо при работе программиста, а умелое использование их сочетаний позволяет строить эффективные программы.

Процесс овладения системой команд «Электроники-100» существенно упрощается благодаря малому числу основных команд. Таблица команд машины приведена в приложении 1 (вкладка). Для повышения эффективности последующего использования этой таблицы при практической работе рекомендуется закреплять с её помощью знания, получаемые при чтении последующего материала.

Общим в структуре записи всех команд является использование старшей триады 12-разрядного слова для кодирования номера команды. Всего, таким образом, имеется 8 основных команд с кодами от 0₈ до 7₈. Команды с кодами от 0₈ до 5₈ являются командами обращения к памяти, команды с кодом 6₈ образуют группу команд обращения к внешним устройствам и команды с кодом 7₈ — группу микрокоманд. Команды с кодами 6₈ и 7₈ являются расширяемой группой команд.

3.1. КОМАНДЫ ОБРАЩЕНИЯ К ПАМЯТИ

Команды обращения к памяти используются в тех случаях, когда необходимо выполнить некоторые операции, требующие использования данных, хранящихся в памяти, засылки результата в память или передачи управления в определенное место программы, характеризующее своим адресом.

Поэтому при записи таких команд должны быть указаны характер операции и адрес тех ячеек памяти, которые участвуют в операции. Характер (вид) операции однозначно определяется её кодом (номером), записываемым в старшей триаде командного слова. Способ записи адресной части команд обращения к памяти является более тонким (но общим для всех таких команд) — он будет разобран ниже, пока же адрес используемой ячейки памяти, выражаемый содержимым трех младших триад, будет условно обозначаться через y . Структура записи команды в этом параграфе, таким образом, имеет вид рис. 3.1.

Рассмотрим вначале сами команды обращения к памяти.

1. Логическое умножение (ЛУ).

Код операции 0₈. По этой команде машина выполняет поразрядное логическое умножение

¹ Операции с двоичными числами исчерпывающе описаны в книге А. И. Китова и Н. А. Криничко. Цифровые вычислительные машины и программирование. М., Физматгиз, 1961.

² Для машинной реализации этой операции предусмотрена специальная команда, см. гл. 3

содержимого накопительного регистра (сумматора) С содержимым ячейки у. Результат остается в накопительном регистре (сумматоре), содержимое ячейки у не изменяется. Поразрядное логическое умножение выполняется по обычным правилам:

$$0 \wedge 0 = 0$$

$$0 \wedge 1 = 0$$

$$1 \wedge 0 = 0$$

$$1 \wedge 1 = 1$$

(здесь $\wedge$ — символ логического умножения).

Пример. В накопительном регистре (сумматоре) содержится код 2276_8 , в ячейке с адресом у — код 7521_8 . После выполнения команды ЛУ в накопительном регистре (на сумматоре) получится код 2020_8 :

$$\begin{array}{r} |HP| \rightarrow 2276_8 \rightarrow 010 \ 010 \ 111 \ 110_2 \\ |y| \rightarrow 7521_8 \rightarrow 111 \ 101 \ 010 \ 001_2 \\ \hline 010 \ 000 \ 010 \ 000_2 = 2020_8 \end{array}$$

Символическое обозначение действий при ЛУ:

$$|HP| \wedge |y| \rightarrow HP.$$

Здесь и далее запись $|*|$ обозначает «содержимое (регистра или ячейки) *».



Рис. 3.1. Условное отображение и список команд обращения к памяти: 0у — логическое умножение содержимого НР (СМ) на содержимое ячейки по адресу у; 1у — (сложение) прибавление к содержимому НР (СМ) содержимого ячейки по адресу у (если возникает «1» переполнения, она прибавляется к содержимому ДР (ДРС)); 2у — увеличение содержимого ячейки по адресу у на 1 и пропуск очередной команды, если новое содержимое оказывается равным нулю; 3у — посылка содержимого НР (СМ) в ячейку по адресу у с (последующим) обнулением НР (СМ); 4у — обращение к подпрограмме с начальным адресом у; 5у — безусловная передача управления по адресу у.

Основное назначение операции ЛУ — выделение содержимого нужных разрядов машинного слова либо «стирание» ненужных разрядов. В первом случае константа ячейки у должна содержать единицы в нужных разрядах, а во втором случае — нули.

Пример. При подготовке кода к печати требуется выделить содержимое младшей триады. В этом случае константа в ячейке должна содержать код 0007_8 .

После выполнения команды ЛУ содержимое старших триад кода в НР(СМ) будет стерто, а содержимое младшей триады останется без изменения.

2. Сложение (СЛ).

Код операции 1_8 . По этой команде машина к содержимому НР (СМ) прибавляет по правилам двоичной арифметики содержимое ячейки у. В каждом разряде, начиная с младшего, сложение выполняется по правилам:

$$0 + 0 = 0$$

$$1 + 0 = 1$$

$$0 + 1 = 1$$

$$1 + 1 = 0 + 1 \text{ переполнения}$$

Как и в обычной десятичной арифметике, единица переполнения «переходит» в соседний старший разряд.

Результат выполнения операции остается на НР (СМ); единица переполнения 12-разрядной сетки складывается с содержимым дополнительного разряда (сумматора), единица переполнения ДР (ДРС) теряется.

Символическое обозначение действий при СЛ:

$$|HP| + |y| \rightarrow HP.$$

Пример. В НР (на СМ) содержится код 2771_8 , содержимое ДР(ДРС) равно 0, в ячейке у содержится код 5142_8 . После выполнения операции СЛ в НР (на СМ) получится код 0133_8 и содержимое ДР (ДРС) станет равным 1:

$$\begin{array}{r} \text{ДР} \\ HP \rightarrow 2771_8 \rightarrow 0 \ 010 \ 111 \ 111 \ 001_2 \\ y \rightarrow 5142_8 \rightarrow + \ 101 \ 001 \ 100 \ 010_2 \\ \hline 1 \ 111 \ 111 \ 111 \ 111_2 \leftarrow \text{Единицы переполнения} \\ \hline \underbrace{1}_{\text{ДР}} \ \underbrace{000 \ 001 \ 011 \ 011}_{\text{НР}} = 0133_8 \end{array}$$

(Штрихом и стрелками указан путь появления и учёта единиц переполнения).

Следует отметить, что тот же результат, естественно, получится и при использовании правил восьмеричной арифметики:

	$0 + 0 = 0$	$1 + 1 = 2$	$2 + 2 = 4$
	$0 + 1 = 1$	$1 + 2 = 3$	$2 + 3 = 5$
	$0 + 2 = 2$	$1 + 3 = 4$	$2 + 4 = 6$
	$0 + 3 = 3$	$1 + 4 = 5$	$2 + 5 = 7$
	$0 + 4 = 4$	$1 + 5 = 6$	$2 + 6 = 0 +$
1переполнения	$0 + 5 = 5$	$1 + 6 = 7$	$2 + 7 = 1 +$
1переполнения	$0 + 6 = 6$	$1 + 7 = 0 + 1$ переполнения	
	$0 + 7 = 7$	или 10_8	
	$3 + 3 = 6$		$4 + 4 = 0 + 1$ переполнения
	$3 + 4 = 7$		$4 + 5 = 1 + 1$ переполнения
	$3 + 5 = 0 + 1$ переполнения		$4 + 6 = 2 + 1$ переполнения
	$3 + 6 = 1 + 1$ переполнения		$4 + 7 = 3 + 1$ переполнения
	$3 + 7 = 2 + 1$ переполнения		
	$5 + 5 = 2 + 1$ переполнения	$6 + 6 = 4 + 1$ переполнения	$7 + 7 =$
6+1переполнения		$6 + 7 = 5 + 1$ переполнения	
	$5 + 6 = 3 + 1$ переполнения		
	$5 + 7 = 4 + 1$ переполнения		

(С учётом переместительного закона сложения начальные строки каждого из последующих столбцов приведенной таблицы могут быть дополнены на основании предыдущих).

Для приведенного примера результат сложения теперь можно подсчитать следующим образом:

$$\begin{array}{r}
 \text{НР} \rightarrow 27\overline{17}1_8 \\
 y \rightarrow 51\overline{14}2_8 \\
 \hline
 \overline{111} \leftarrow \text{Единицы переполнения} \\
 \hline
 \overline{101} \quad \overline{33}_8 \\
 \text{ДР} \quad \text{НР}
 \end{array}$$

Операция сложения является основой выполнения всех арифметических операций в «Электронике-100». Кроме того, она используется для вызова кодов в НР (на СМ) при различных пересылках кодов или для их анализа—для этого НР предварительно обнуляют, а затем выполняют операцию СЛ. При выполнении СЛ содержимое ячейки у не изменяется.

3. Увеличение и пропуск (УП).

Код операции 2_8 . Полное наименование команды должно было бы звучать так: «Увеличение и пропуск очередной команды (при определенных условиях)». Эта команда является одной из наиболее изящных в системе команд «Электронике-100».

По команде УП содержимое ячейки у увеличивается на 1 (младшего разряда), и если новое содержимое ячейки у равно нулю (единица переполнения разрядной сетки теряется), происходит пропуск следующей команды. При выполнении команды УП содержимое НР (СМ) и ДР (ДРС) не изменяется.

Символическое обозначение действий, выполняемых по команде УП, выглядит следующим образом:

$$\begin{aligned}
 |y| + 1 &\rightarrow y; \\
 |CK| + 1 &\rightarrow CK, \text{ если } |y| = 0,
 \end{aligned}$$

что означает, содержимое ячейки у, увеличенное на 1, засылается в ячейку у и при указанном условии содержимое счетчика команд, увеличенное на 1, засылается в счетчик команд. Запись отражает и последовательность действий (вначале содержимое ячейки у увеличивается на 1, а затем анализируется новое содержимое), и способ реализации пропуска команды (путем дополнительного увеличения на 1 содержимого счетчика команд — а именно, его содержимое определяет адрес ячейки, откуда будет выбираться следующая команда, см. гл. 2).

Основное назначение команды — простая реализация различных счетчиков (в частности, счетчиков циклов), переадресация, организация разветвлений (см. гл. 4).

4. Посылка и сброс (ПС).

Код операции 3_8 . По этой команде содержимое НР (СМ) посылается в ячейку по адресу у (старое содержимое ячейки теряется, а сам НР (СМ) очищается). Содержимое ДР (ДРС) не изменяется. Символическая запись действий при операции ПС:

$$|НР| \rightarrow y, 0 \rightarrow НР.$$

Основное назначение операции — засылка результатов действий, выполняемых с использованием накопительного регистра (сумматора), в оперативную память машины.

5. Обращение к подпрограмме (ОП).

Код операции 4_8 . При обращении к подпрограмме машине необходимо указать адрес начала подпрограммы и позаботиться о возвращении управления в следующую ячейку основной программы по окончании работы подпрограммы. Для этого в «Электронике -100» предусмотрена специальная структура

основной организации подпрограмм — первая ячейка поля подпрограмм отводится для запоминания адреса возврата из подпрограммы, а собственно подпрограмма начинается со следующей ячейки; предполагается также, что подпрограмма заканчивается безусловной передачей управления по адресу, запасенному в первой ячейке подпрограммы.

В соответствии с этим в самой команде оказывается возможным указать только начальный адрес y поля, где размещается подпрограмма. При выполнении команды ОП в ячейку с адресом y будет автоматически записан абсолютный адрес возврата (адрес ячейки, следующей за той, в которой располагается команда с кодом 4_8), а управление передается в ячейку с адресом $y + 1$; содержимое НР (СМ) и ДР (ДРС) не изменяется.

Символическая запись действия при выполнении операции ОП:

$$|СК| \rightarrow y, y + 1 \rightarrow СК.$$

Напомним (см. гл.1), что при выполнении любой команды в СК выбирается (формируется) абсолютный адрес следующей команды, т. е. адрес ячейки, откуда очередная команда будет выбрана. Таким образом, в момент выполнения команды ОП в СК уже содержится адрес следующей ячейки памяти, где располагается продолжение основной программы, к которому надо обратиться по возвращении из подпрограммы. Этот адрес и запасается в начальной ячейке памяти, отведенной под поле подпрограммы. Одновременно (в течение одного машинного такта) вместо старого содержимого в СК заносится адрес $y+1$, с которого и начинается собственно подпрограмма, а машина, формально использующая СК как путеводитель по памяти, передает управление в ячейку $y + 1$.

О выходе из подпрограммы уже упоминалось выше. Важно отметить, что в ячейке y запасается абсолютный адрес возврата и это обеспечивает выход из подпрограммы независимо от места её расположения в памяти. В частности, основная программа и подпрограмма могут располагаться на разных страницах МОЗУ.

Команда обращения к подпрограмме позволяет строить экономные программы в тех случаях, когда требуется одно или несколько повторений в различных местах основной программы одних и тех же действий (приемов, вычислений и т. п.). Повторяющиеся действия выполняются в виде подпрограммы, и в соответствующих местах основной программы записываются лишь команды ОП (см. гл. 4).

6. Безусловная передача управления (БП).

Код операции 5_8 . По этой команде производится изменение последовательности выполнения команд независимо от дополнительных условий. При выполнении команды БП управление всегда передается по адресу, указанному адресной частью команды; содержимое НР (СМ) и ДР (ДРС) не изменяется.

Формальная запись действий при БП выглядит следующим образом:

$$y \rightarrow СК.$$

Команда БП используется при организации разветвлений, циклов, обходов участков памяти, занятых константами, для выхода из подпрограмм и в ряде других случаев.

3.2. АДРЕСНАЯ ЧАСТЬ КОМАНД ОБРАЩЕНИЯ К ПАМЯТИ

Как уже упоминалось, существенную роль в структуре команд обращения к памяти играет её страничная организация. Вся память разбита на блоки по 128 ячеек; в каждом блоке ячейки пронумерованы от 000_8 до 177_8 . (Для этого требуется 7 двоичных разрядов. Легко заметить, что начала страниц имеют абсолютные номера ячеек, кратные 200_8 ; 0000 ; 0200 ; 0400 ; 0600 ; 1000 и т. д. Сквозная нумерация ячеек МОЗУ в своих младших семи разрядах периодически повторяет нумерацию ячеек внутри каждой страницы.) Из каждой ячейки возможно прямое адресование к любой из 200_8 ячеек памяти, расположенных на той же текущей странице, и ещё к 200_8 ячейкам памяти на нулевой странице. Кроме того, имеется возможность косвенного адресования к любой из 4096 ячеек МОЗУ. Для указания текущей или нулевой страницы, а также для указания прямого или косвенного способа адресования в структуре команд обращения к памяти отводится по одному разряду (см. рис. 2.3).

Таким образом, адресная часть команд обращения к памяти складывается из семи двоичных разрядов адреса на странице и двух признаков: признака нулевой или текущей страницы и признака прямого или косвенного адресования. Девять двоичных разрядов адресной части команды при её записи в восьмеричном виде после группировки по триадам образуют три (младших) цифры в общем коде команды (старшую цифру составляет код операции). При этом две младшие триады адреса на странице целиком переходят в две младшие триады адресной части команды. Третья же справа триада (и, следовательно, третья справа цифра в команде) определяется сразу тремя источниками — двумя признаками и значением старшего разряда адреса на странице. Её запись и использование является наиболее тонким моментом в определении адресной части команд обращения к памяти.

При записи каждой команды третью справа цифру приходится синтезировать по трем указанным источникам, принимая во внимание «вес» каждого из разрядов в триаде и следующие значения признаков:

3-й разряд	0 — прямое адресование,
(признак косвенного адресования)	1 — косвенное адресование,
4-й разряд	0 — нулевая страница,
(признак текущей страницы)	1 — текущая страница.

Так как 3-й разряд в рассматриваемой триаде имеет вес $2^2(=4)$, 4-й разряд $2^1(=2)$ и пятый разряд $2^0(=1)$, то результирующее значение этой триады складывается из значения старшего разряда адреса на странице с единичным весом, двойки, если используется адрес на текущей странице, и четверки, если адресование косвенное. В случае прямого адресования в рассматриваемой триаде равен нулю 3-й разряд и, следовательно, в её значении отсутствует четверка; при адресовании к нулевой странице равен нулю 4-й разряд — и в результирующем значении триады отсутствует двойка.

Общее выражение для определения третьей справа цифры команд обращения к памяти можно, таким образом, записать в следующем виде: $\Sigma = 4 X$ (признак косвенного адресования) + $2 X$ (признак текущей страницы) + (значение старшего разряда адреса на странице).

Примеры. Пусть в некотором месте программы требуется к содержимому НР(СМ) прибавить константу С. Это можно сделать несколькими способами, хотя везде используется операция сложения (код операции I_8):

а) константу С запасают на той же текущей странице памяти, где расположена и сама программа, например в ячейке 177_8 этой страницы (в самом конце страницы, чтобы оставить максимум свободного места для команд — «текста» программы):

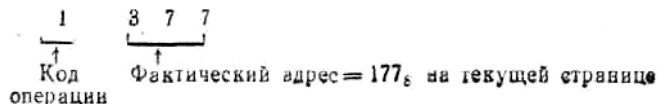
Адрес на странице	Содержимое
177_8	С

В адресной части команды достаточно прямо адресоваться к этой ячейке на текущей странице, т. е. признак косвенного адресования = 0 (адресование прямое),

признак текущей страницы = 1

старший разряд адреса на странице = 1 (от используемого адреса 177_8).

Таким образом, $\Sigma = 4 X 0 + 2 X 1 + 1 = 3$, и команда, реализующая требуемую операцию, выглядит



как

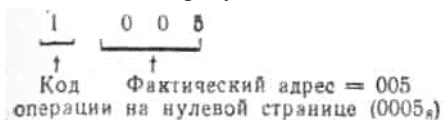
б) константа С имеется на нулевой странице, например, записана в ячейке 0005_8 (адрес на странице— 005_8)¹. При этом можно использовать прямое адресование на нулевую страницу, и в таком случае

признак косвенного адресования = 0 (адресование прямое),

признак текущей страницы = 0 (адресование на нулевую страницу),

старший разряд адреса на странице = 0 (от используемого адреса на странице 005_8).

Таким образом, $\Sigma = 4 \times 0 + 2 \times 0 + 0 = 0$, и требуемая команда имеет вид



а) если константа С имеется в некоторой ячейке не на нулевой и не на текущей странице, может использоваться косвенное адресование. Пусть, например, константа С запасена в ячейке с адресом 2000_8 . В этом случае адрес 2000_8 заносят в некоторую ячейку на текущей или нулевой странице и затем осуществляют косвенное адресование к этой ячейке.

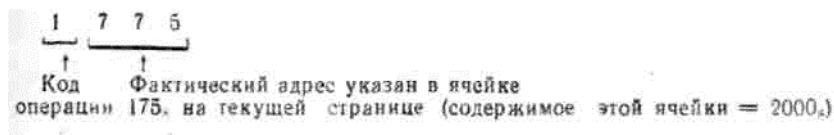
Например, адрес 2000_8 занесен в ячейку 175_8 на текущей странице; тогда

признак косвенного адресования = 1,

признак текущей страницы = 1,

старший разряд адреса на странице = 1.

При этом $\Sigma = 4 \times 1 + 2 \times 1 + 1 = 7$, и команда имеет вид



Замечания.

1. Из сравнения п.п. а) и в) следует, что в последнем случае требуется дополнительная ячейка для указания фактического (абсолютного) адреса константы С. Можно с тем же эффектом использовать эту ячейку для дополнительного хранения константы и далее воспользоваться приемом п. а).

2. Косвенное адресование весьма эффективно и для других команд обращения к памяти (УП, ПС, ОП, БП) при их отдельном употреблении и совершенно необходимо при организации циклических программ (см.

¹ Адрес на странице определяет 7 младших двоичных разрядов абсолютного адреса ячейки, и при восьмеричной записи старшая триада адреса на странице является неполной. Дополняя полученный 7-разрядный двоичный код двумя незначащими нулями слева, можно считать формально правильной приведенную восьмеричную запись адреса на странице, однако следует помнить, что фактически этот адрес всегда содержит лишь 7 двоичных разрядов.

гл. 4).

3. Так как к нулевой странице возможно прямое адресование из любого места памяти, то она имеет особую ценность. На нее обычно выносят наиболее употребительные константы, адреса подпрограмм и т. п. Кроме того, на нулевой странице каждого куба МОЗУ располагаются 10_8 индексных ячеек и на нулевую страницу 0-го куба МОЗУ передается управление при прерываниях (см. ниже).

Особенности использования индексных ячеек

Ячейки памяти с номерами 0010_8 — 0017_8 в «Электронике-100» являются индексными. Имеется определенная специфика их использования. При прямом адресовании эти ячейки работают как обычные ячейки памяти. При каждом косвенном адресовании к индексной ячейке её содержимое *вначале увеличивается* на единицу, а затем используется в качестве адреса. Такие свойства индексных ячеек позволяют строить эффективные циклические программы (см. гл. 4).

3.3. КОМАНДЫ ОБРАЩЕНИЯ К ВНЕШНИМ И ПЕРИФЕРИЙНЫМ УСТРОЙСТВАМ

Код операции 6_8 . В структуре этих команд предусмотрено указание номера (восьмеричного кода) внешнего устройства и вида его взаимодействия с процессором (см. рис 2.4). Из 12 бит (двоичных разрядов) старшая триада отводится под код операции, две следующие — для кода внешнего устройства и младшая триада — для указания вида взаимодействия данного внешнего устройства с процессором. Таким образом обеспечивается принципиальная возможность программного управления работой 64 внешних устройств при семи видах взаимодействия с каждым из них. Использование в качестве части внешних устройств разного рода коммутаторов позволяет дополнительно расширить эти пределы.

Принципы построения внешних и периферийных устройств, их назначение и обращение к ним настолько разнообразны, что не удастся иметь даже полностью идентичной системы команд в части указания вида взаимодействия. Можно выделить, однако, несколько групп внешних устройств с идентичными командами обращения к ним: устройства ввода — вывода, преобразователи аналог — код и код — аналог, коммутаторы и блоки управления индикацией, шаговыми двигателями и т. п. Некоторые периферийные устройства имеют дополнительные особенности обращения.

Ниже рассматриваются наиболее типовые команды (см. также табл. 3.1).

Устройства ввода — вывода

В командах обращения к этим устройствам разряды младшей триады «закреплены» за определенными действиями: единица в одиннадцатом разряде вызывает опрос флага внешнего устройства и пропуск очередной команды, если флаг находится в единичном состоянии; единица в десятом разряде (имеющая в триаде вес, равный двойке) соответствует гашению флага; единица в девятом разряде (вес в младшей триаде равен четырем) — обмен между НР (СМ) и буфером внешнего устройства. Возможно применение микропрограммирования¹; особенно часто совмещают в одной команде обмен информацией и гашение флага (в младшей триаде при этом оказывается код 6_8) — таким путем очередной обмен сопровождается подготовительной операцией для опознания готовности внешнего устройства к последующему обмену (после отработки каждого обмена и ввода-вывода очередного кода флаг внешнего устройства всегда устанавливается в единичное состояние).

Так как основным устройством ввода-вывода минимального комплекса является пишущая машинка, а долговременным носителем информации — 8-дорожечная перфолента, имеется ряд особенностей, связанных с унификацией канала ввода-вывода. Все устройства ввода как внешние устройства имеют общий номер 03_8 (аналогично номер выводных устройств 04_8); буферные регистры, как ввода, так и вывода имеют меньшее, чем 12, число разрядов, и обмен между буфером и младшими разрядами НР (СМ) идет неполными словами (по 8 бит).

Дополнительными особенностями канала ввода является обнуление не только флага, но и НР (СМ) по команде 6032 (при работе со считывателем эта же команда инициирует ещё и начало движения перфоленты), и логическое сложение содержимого буфера ввода с содержимым НР (СМ) по команде 6034² (результат остается на НР).

Пример использования в программе команд ввода см. в § 4.11

Таблица 3.1

Код команды	Наименование команды	Пояснения
6001	Разрешить прерывание	Выполняется ещё одна команда и разрешается прерывание
6002	Запретить прерывание	

¹ При объединении нескольких операций обращения к внешнему устройству в рамках одной команды вначале выполняется операция, соответствующая команде $6nm1$, затем $6nm2$ и лишь потом $6nm4$ (при наличии единицы в 11-, 10- и 9-м разрядах соответственно).

² Следует отметить, что команда 6036 работает так, как формально и должна была бы работать команда обмена, смикропрограммированная с командой 6032: $0 \rightarrow \text{НР (СМ)}$, $0 \rightarrow \text{флаг ВВ}$, $|\text{буфер ВВ}| \rightarrow \text{НР (СМ)}$.

6031	Опрос флага ввода	Если флаг ввода в «1», пропускается очередная команда
6032 ¹	Очистка флага ввода и НР	Флаг ввода устанавливается в «0» и обнуляется НР (СМ)
6034	Считывание 1	К восьми младшим разрядам НР (СМ) по правилам логического сложения прибавляется содержимое буферного регистра ввода: разряды 0—3 НР (СМ) содержимого не изменяют
6036 ¹	Считывание 2	Считывание содержимого буферного регистра ввода в НР (СМ) с предварительной очисткой НР (СМ)
6041	Опрос флага ввода	Если флаг ввода в «1», пропускается очередная команда
6042	Очистка флага вывода	
6046	Вывод на печать или перфорацию одного символа (кода) с предварительной очисткой флага вывода	Очистка флага вывода, пересылка содержимого восьми младших разрядов НР (СМ) в буферный регистр вывода и начало собственно вывода; по окончании вывода флаг вывода установится в «1»

При расширении комплектности машины:

6004	Запуск аналого-цифрового преобразователя (АЦП)	Очистка НР (СМ) и запуск АЦП; по окончании работы АЦП код преобразованного сигнала заносится в разряды 0—9 НР (СМ)
6133	Опрос и очистка флага таймера	Если флаг таймера в «1», пропускается очередная команда; флаг таймера устанавливается в «0»
6137	Запуск таймера на очередной цикл	Содержимое НР (СМ) заносится в буферный регистр таймера — запуск таймера; при досчете до 7777 ₈ флаг установится в «1»
62N1	Установка поля данных	N засылается в регистр поля данных и команды ЛУ, СЛ, УП и ПС с косвенным адресованием оперируют с данными из куба с номером N (но адрес операнда выбирается из того же поля, где расположена программа)
62N2	Установка поля команд	Подготавливается пересылка N в регистр поля команд, сама пересылка (и переход в куб с номером N) происходит при выполнении первой (последующей) команды ОП или БП; до выполнения команд ОП или БП разрешенное ранее прерывание блокируется
6214	Считывание поля данных	Содержимое РПД считывается по правилам логического сложения в разряды 6—8 НР (СМ) ²
6224	Считывание поля команд	Содержимое РПК считывается по правилам логического сложения в разряды 6—8 НР (СМ) ²
6234	Считывание регистра запоминания поля	Содержимое РПК записывается в разряды 6—8 НР (СМ), а содержимое РПД — в разряды 9—11 НР (СМ)
6244	Восстановление полей памяти	Восстанавливается содержимое РПД и РПК, запасенное ранее (при прерывании) в РЗП

Команды управления прерыванием и преобразования аналог — код

Для управления прерыванием требуется обеспечить два режима — разрешение прерывания и его запрещение. Такая задача в принципе решается использованием триггера, что внешне очень сходно с использованием триггера готовности внешних устройств ввода — вывода (флага). Аналогичны (внешне) и команды управления этими триггерами: единица в младшей триаде команды означает разрешение прерывания (при этом как бы опрашивается флаг, — подробнее о работе системы прерывания см. § 3.4), запоминаемое в данном случае на триггере; двойка в младшей цифре команды (единица в 10-м разряде) означает запрещение прерывания — сброс триггера разрешения прерывания полностью аналогичен сбросу флага.

¹ И начало движения перфоленты при работе со считывателем; после считывания очередной строки перфоленты флаг ввода установится в «1».

² Содержимое разрядов 0-5 и 9-11 НР (СМ) не изменяется

Блок прерывания в машине рассматривается как внешнее устройство с номером 00_8 , и команды управления прерыванием, таким образом, имеют вид:

6001 — включить прерывание,

6002 — выключить прерывание.

Нетрудно видеть, что в младшей триаде эти команды используют лишь два разряда из трех. Вместе с этим, для управления работой преобразователя аналог — код требуется единственная команда, инициирующая начало его работы. По-видимому, в целях экономии адресов внешних устройств преобразователю аналог — код как внешнему устройству также присвоен номер 00_8 , и команда управления им выглядит так:

6004.

По этой команде (внешне напоминающей команду обмена с внешним устройством с номером 00_8) производится обнуление НР (СМ) и последующее преобразование аналогового сигнала, подведенного ко входу преобразователя, в код.

Аналоговый сигнал должен представлять собой напряжение в диапазоне $0 \text{—} +10 \text{ В}$; этот сигнал линейно преобразуется в 10-разрядный двоичный код. Результат преобразования получается в НР (на СМ) в разрядах $0 \text{—} 9$ (в 0-м разряде НР находится старший разряд кода, в 9-м — младший); в 10-м разряде НР (СМ) после работы преобразователя аналог — код находится единица. Затем полученный после преобразования код обычно нормализуют сдвигом вправо на 2 разряда с предварительной очисткой ДР (ДРС).

Использование дополнительных кубов памяти

Как видно из рассмотрения команд обращения к памяти (см. § 3.1, 3.2), разрядная сетка машины и система команд позволяют относительно просто строить программы, когда программа и все данные уместятся в одном кубе памяти. Однако возможны случаи, когда 4096 12-разрядных слов оказывается недостаточным, чтобы запасти необходимые сведения из внешнего мира или описать все необходимые действия с ними («не хватает слов!»).

Использование дополнительных кубов оперативной памяти часто устраняет такие затруднения. Адресование же в этих случаях принимает двухступенчатую структуру: вначале указывается так называемое «поле команд» и (или) «поле данных» (т. е. номер куба, оперативная память которого используется для размещения последующих команд или данных), а затем обычным путем записывается программа. Для программного управления переходами из куба в куб занята часть команд с кодом операции 6_8 : $62N1$ — установка поля данных с номером N ($N = 0 \text{—} 7$);

$62N2$ — установка поля команд с номером N ;

$62M4$ — различные операции по считыванию поля данных ($M = 1$), поля команд ($M = 2$) и регистра запоминания полей ($M = 3$) в НР (на СМ), а также операция восстановления Полей в соответствии с содержимым регистра запоминания полей ($M = 4$).

Для данного раздела важно лишь, что схемы управления расширенной памятью формально рассматриваются как 10_8 внешних устройств, занимая фактически все команды для устройств с номерами от 20_8 до 27_8 . Подробно работа каждой из этих команд рассматривается в гл. 4.

Команды обращения к коммутаторам, блокам управления шаговыми двигателями, цифровой индикацией и т. п.

Во всех случаях обращения к этим внешним устройствам они должны получить некоторую цифровую информацию (номер канала — при работе с коммутатором, управляющий код или управляющее слово — при использовании шаговых двигателей или цифровой индикации), а затем — отдельную команду на исполнение (включение канала, обработку управляющего слова и т.п.). В соответствии с этим имеется общность и в структуре цепочки команд, управляющих работой таких устройств. Вначале в НР(СМ) вызывается цифровая информация, которая по одной из команд обращения к данному внешнему устройству¹ переписывается в его буферный регистр; в конце цепочки команд (после обнуления НР, если оно необходимо) другая команда обращения к этому же внешнему устройству производит его запуск.

Для коммутаторов команда, по которой в буферный регистр блока управления коммутатором переписывается номер канала, имеет в младшей триаде код четверки, а подключение, передача или запись полезной информации осуществляется по команде обращения к (данному) внешнему устройству с кодом двойки в младшей триаде (для выходных коммутаторов этой команде предшествует обнуление НР и вызов на него полезной информации).

Для блоков управления шаговыми двигателями, цифровой индикацией и т. п. передаваемая цифровая информация уже сама по себе является с самого начала полезной информацией (в отличие от служебной информации о номере канала для коммутаторов). Команда обмена полезной информацией и здесь выглядит как команда обращения к внешнему устройству с кодом двойки в младшей триаде, запуск устройств осуществляется по команде обращения к внешнему устройству с кодом четверки в младшей триаде.

Когда разрядной сетки машины оказывается недостаточно для обмена полезной информацией, используют многократное повторение операции обмена (несколькими словами), применяя при необходимости коммутаторы дискретных сигналов.

1. Ввод сигналов через коммутатор аналоговых сигналов (и АЦП) осуществляется с использованием следующей последовательности команд:

Код номера поста и номера датчика → НР (СМ)

¹ См. выше описание общей структуры команд обращения к внешним устройствам

6076 (обнуление управляющих регистров коммутатора и засылка в них кода из НР)
6004 (включение АЦП)
7112 (нормализация полученного кода путем сдвига его на 2 разряда вправо с предварительной очисткой ДР)
6072 (закрытие ключей опрошенного канала)

Формат расположения кода номеров поста и датчика имеет вид:

0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	Код номера поста			Код номера датчика					

2. Ввод кодов через коммутатор дискретных сигналов выполняется с помощью таких команд:

Номер канала → в НР (СМ)
6314 (открывание ключей указанного канала)
7200
6312 (запись информации в НР и отключение канала)

3. Вывод кодов из ЭВМ через выходной коммутатор дискретных сигналов:

Номер канала → в НР (СМ)
6324 (открывание ключей от выходного буферного регистра к указанному потребителю)
Выводимый код → в НР (СМ)
6322 (запись информации из НР в буферный информационный регистр коммутатора)

4. Управление шаговыми двигателями:

Номер соответствующего канала выходного коммутатора дискретных сигналов → в НР (СМ)
6324
Информационное слово → в НР (СМ)
6672 (запись информации в счетчик импульсов досчета)
6674 (запуск счетчика досчета)

Информационное слово при управлении шаговыми двигателями имеет следующий формат:

0	1	2	3	4	5	6	7	8	9	10	11
						Обратный код числа импульсов					

Направление движения:
0 – вперед, 1 – назад

5. Управление цифровой индикацией — пример одной из более сложных структур управления внешними устройствами и соответственно более сложной цепочки команд.

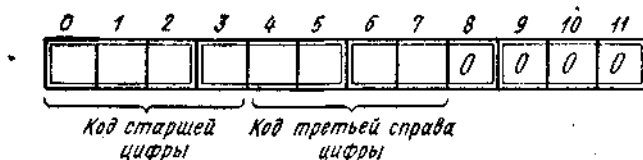
Сама цифровая индикация выполнена на газоразрядных трубках по четыре десятичных цифры на каждом индикаторном устройстве, причем, после второй справа, третьей или четвертой (старшей) цифры может высвечиваться запятая. Таким образом, количество бит информации, необходимое для управления одним устройством, превышает 12, т.е. не может быть передано одним машинным словом. Поэтому буферный регистр устройства управления цифровой индикацией получает информацию через два канала выходного коммутатора дискретных сигналов.

Устройство управления цифровой информацией рассчитано на обслуживание нескольких (до 6) индикаторных устройств, так что получается своеобразная двухступенчатая структура: через выходной коммутатор дискретных сигналов полезная информация «перекачивается» в буферный регистр устройства управления цифровой индикацией, а какому из шести подключенных индикаторных устройств эта информация предназначена, сообщается из машины дополнительно, при выполнении исполнительной команды. Это вспомогательное служебное слово, стробирующее (избирающее) одно из индикаторных устройств, заодно переносит информацию и о положении запятой в высвечиваемом числе. Дополнительной особенностью схемы управления здесь является объединение в одном слове информации о положении запятой в двух соседних индикаторных устройствах (см. ниже).

Таким образом, цепочка команд для случая, когда буферный регистр устройств управления цифровой индикацией получает информацию по каналам «а» и «в» выходного коммутатора дискретных сигналов, имеет вид:

Код «а» → в НР (СМ)
6324
Старшее информационное слово → в НР (СМ)
6322
6754 (пересылка информации из буферного регистра выходного коммутатора в буферный регистр управления цифровой индикацией)
Код «в» → в НР (СМ)
6324
Младшее информационное слово → в НР (СМ)
6322
6754
Служебное слово (строб + запятые) → в НР (СМ)
6752 (пересылка информация о запятых и включение избранного индикаторного устройства)

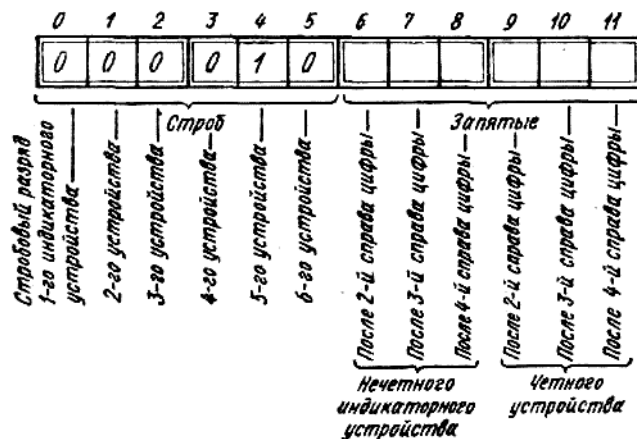
Формат старшего информационного слова:



Формат младшего информационного слова:



Формат служебного слова «строб + запятые»:



(Все индикаторные устройства при занесении запятой работают парами — 1 и 2, 3 и 4, 5 и 6, так что для указанного выше строга разряды 6—8 несут информацию о положении запятой в 4-разрядном десятичном числе, которое будет заноситься в пятое индикаторное устройство, а разряды 9—11 должны нести информацию, подтверждающую положение запятой в шестом индикаторном устройстве.)

Аналогичную структуру и состав команд управления имеет световое табло — единственное отличие состоит в том, что если блок управления цифровой индикацией рассматривается как внешнее устройство с номером 75₈, то под световое табло отведен номер 74₈ (и соответственно команды 6744 и 6742).

Ввод частотных сигналов осуществляется с помощью команд их коммутации (через входной коммутатор аналоговых сигналов), очистки и запуска счетчика в блоке частотно-кодowego преобразователя (БЧКП) и считывания кода в НР(СМ) по окончании работы преобразователя. Однако из-за относительно большой длительности преобразования способ обращения к БЧКП окончательно пока не определился. Важно лишь отметить, что здесь предполагается задействовать два номера для внешних устройств: 12₈ — для счетчика и 73₈ — для схемы управления запуском БЧКП.

Команды обращения к таймеру

Номер таймера, как внешнего устройства, 13₈. Для управления работой таймера в автоматическом режиме используются следующие команды:

6133 — опрос флага таймера, его обнуление и пропуск одной команды, если флаг был в единичном состоянии;

6137 — запись исходного кода из НР(СМ) в регистр таймера, запуск схемы досчета и схемы организации прерывания, когда содержимое регистра таймера станет равным 7777₈.

Стандартные импульсы досчета отстоят друг от друга по времени на 40 мс (с точностью, определяемой точностью поддержания частоты промышленной сети — около 1%; имеется возможность использования в таймере и высокостабильных кварцевых генераторов); число импульсов, задаваемых до очередного прерывания, должно быть указано (занесено на НР) в обратном коде. Одновременно с запросом на прерывание таймер устанавливает свой флаг в единичное состояние.

Диапазон задержек (период прерывания), реализуемый с помощью таймера, лежит в диапазоне 80 мс — 2,73 мин.

Некоторые экземпляры машин могут иметь таймер с расширением диапазона задержек в два и в четыре раза. Для указания диапазона перед командой 6137 должна использоваться специальная команда:

6771 — исходный диапазон (цена импульса 40 мс);

6772 — цена импульса увеличена вдвое;

6774 — цена импульса увеличена в четыре раза.

Как и в других случаях, возможности машины могут быть расширены за счет программирования — для получения больших задержек в программе организуется дополнительный счетчик импульсов прерывания таймера.

3.4. ПРОГРАММНОЕ ПРЕРЫВАНИЕ

Важным свойством «Электроники-100» является возможность прерывания решения задач по сигналам внешних устройств. Любое из внешних устройств может подавать импульсный или потенциальный сигнал на так называемую шину прерывания, имеющуюся в машине, сигнализируя этим о появлении нового запроса на использование процессора. Если необходимые условия выполнены (см. ниже), происходит прерывание решаемой задачи путем имитации выполнения команды 4000₈, т. е. обращения к подпрограмме, поле которой начинается с ячейки 0000₈ (0-го куба МОЗУ).

Необходимые условия прерывания состоят в том, чтобы было разрешено прерывание (где-то по программе была бы выполнена команда 6001 и далее не встречалось команды 6002 и не было прерываний), чтобы появился запрос на прерывание хотя бы от одного из внешних устройств и чтобы после выполнения первых двух условий закончилось выполнение одной (очередной) команды.

Таким образом, если разрешено прерывание (команда 6001 выполнялась) и затем внутри цикла выполнения какой-либо команды происходит запрос на прерывание от внешнего устройства, то выполнение этой команды заканчивается, а затем в ячейку 0000₈ заносится содержимое СК (адрес следующей команды, которая выполнялась бы в отсутствие прерывания) и управление передается в ячейку 0001₈, что

¹ Имеется возможность ручного однократного запуска таймера, а также возможность работы с фиксированными временными интервалами 15, 30, 60, 120 с.

имитирует выполнение команды ОП с кодом 4000₈ между командой, при выполнении которой пришел запрос от внешнего устройства, и следующей командой.

Если же вначале появился (и ещё сохраняется) запрос на прерывание от внешнего устройства, а затем по программе выполняется команда с кодом 6001₈, то вначале выполняется одна команда, непосредственно следующая за 6001₈, а затем имитируется выполнение команды ОП по адресу 0000₈.

Программа обслуживания прерываний всегда начинается в ячейке 0001₈ и может иметь самый различный характер в зависимости от набора внешних устройств, входящих в систему, и принципов построения общей программы. Чаще всего, однако, эта программа требует использования НР (СМ), и поэтому одним из первых действий при обслуживании прерываний является запоминание содержимого НР (СМ) и ДР (ДРС), имевших место в момент прерывания и сохранившихся там к моменту передачи управления в ячейку 0001₈. Для этого требуется минимум три команды: запись | НР | в некоторой буферной ячейке, циклический сдвиг¹, обычно влево для того, чтобы содержимое ДР (ДРС) попало в НР (СМ), и запись (в другую буферную ячейку) этого содержимого. Таким образом, необходимый минимум информации о прерванной задаче оказывается обеспеченным — в ячейке 0000₈ имеется адрес продолжения задачи, в буферных ячейках запасены состояния двух других основных регистров машины в момент прерывания.

Далее обычно следует цепочка команд, обеспечивающих опрос флагов всех устройств, которые могли вызвать прерывание, с выходом на соответствующие программы обслуживания. Заканчивается программа обслуживания прерываний восстановлением содержимого НР (СМ) и ДР (ДРС) в момент прерывания, разрешением прерывания и безусловной передачей управления с косвенным адресованием к ячейке 0000₈ (на продолжение прерванной задачи). Подробнее об этом см. в гл. 4 и 8.

Важно отметить, что на время отработки очередного прерывания (до нового выполнения команды 6001₈) обслуживание любых других прерываний блокируется. Для того чтобы не потерять запросы на прерывание, желательно иметь все сигналы, выходящие на шину прерывания, не импульсными, а потенциальными. Вместе с этим внешнее устройство, запрашивающее прерывание, должно установить свой флаг в единичное состояние — для опознания источника прерывания. Поэтому рационально строить (электрические) схемы системы таким образом, чтобы сигнал прерывания заводился на шину прерывания именно от флага внешнего устройства — триггера, имеющего потенциальный выход. Именно так (с использованием флагов) в машине построено, в частности, взаимодействие схемы прерывания с устройствами ввода—вывода.

3.5. ГРУППЫ МИКРОКОМАНД

Микрокоманды объединяют большую группу операций, связанных с разнообразными логическими действиями над содержимым НР (СМ) и ДР (ДРС), а также с анализом содержимого этих регистров. В отличие от остальных команд, микрокоманды (код операции 7₈) используют для указания требуемых действий фактически все девять младших разрядов машинного слова. Структура микрокоманд показана на рис. 2.5 и описывалась в § 2.2.

Все микрокоманды разбиты на две группы (табл. 3.2, 3.3). Микрокоманды первой группы (табл. 3.2) всегда имеют 0 в 3-м разряде (старшем разряде второй слева триады), микрокоманды второй группы (табл. 3.3) имеют в этом разряде 1. Внутри первой группы микрокоманд каждый разряд «закреплен» за определенной операцией (эта операция выполняется, если в данном разряде записана единица); внутри микрокоманд второй группы дополнительное увеличение числа операций происходит за счет придания различного смысла содержимому 5-, 6- и 7-го разрядов в зависимости от содержимого 8-го разряда.

Таблица 3.2

Код команды	Наименование команды	Пояснения
7000	Нет операции	Задержка на один машинный цикл
7001	Увеличение содержимого НР (СМ) на 1	
7004	Циклический сдвиг влево на 1 разряд	13-разрядный код, содержащийся в ДР (ДРС) и НР (СМ), циклически сдвигается влево на 1 разряд; перемещение происходит в соседний разряд, а из ДРС — в младший разряд НР (СМ).
7006	Циклический сдвиг влево на 2 разряда	13-разрядный код в ДРСМ ¹ циклически сдвигается влево на 2 разряда
7010	Циклический сдвиг вправо на 1 разряд	13-разрядный код в ДРСМ ¹ циклически сдвигается вправо на 1 разряд
7012'	Циклический сдвиг вправо на 2 разряда	13-разрядный код в ДРСМ ¹ циклически сдвигается вправо на 2 разряда
7020	Инвертирование содержимого ДР (ДРС)	

¹ О работе команд циклического сдвига см. следующий параграф

7021	Инвертирование содержимого ДР и увеличение содержимого НР (СМ) на 1
7040	Инвертирование содержимого НР (СМ)

Продолжение

Код команды	Наименование команды	Пояснения
7041	Образование дополнительного кода в НР (СМ)	Обнуление НР (СМ)
7060	Инвертирование содержимого НР (СМ) и ДР (ДРС)	
7100	Очистка ДР (ДРС)	
7101	Очистка ДР (ДРС) и увеличение содержимого НР (СМ) на 1	
7104	Циклический сдвиг влево на 1 разряд с предварительной очисткой ДР (ДРС)	
7106	Циклический сдвиг влево на 2 разряда с предварительной очисткой ДР (ДРС)	
7110	Циклический сдвиг вправо на 1 разряд с предварительной очисткой ДР (ДРС)	
7112	Циклический сдвиг вправо на 2 разряда с предварительной очисткой ДР (ДРС)	
7120	Запись «1» в ЛР (ЛРС)	
7121	Запись «1» в ДР (ДРС) и увеличение содержимого НР (СМ) на 1	
7140	Очистка ДР (ДРС) и инвертирование содержимого НР (СМ)	
7200	Очистка НР (СМ)	
7201	Засыпка кода 0001 в НР (СМ)	
7204	Засыпка содержимого ДР (ДРС) в младший разряд НР (СМ)	
7206	Засыпка содержимого ДР (ДРС) в 10-й разряд НР (СМ)	
7210	Засыпка содержимого ДР (ДРС) в старший (0-й) разряд НР (СМ)	
7212	Засыпка содержимого ЛР (ЛРС) в 1-й разряд НР (СМ)	
7240	Засыпка «—1» (кода 7777) в НР (СМ)	
7300	Очистка НР (СМ) в ЛР (ЛРС)	
7320	Очистка НР (СМ) и запись «1» в ДР (ДРС)	
7340	Очистка ДР (ДРС) и запись «—1» в НР (СМ)	
' ДРСМ — объединенный регистр ДР (ДРС) и НР (СМ)		

Дополнительную гибкость и производительность микрокоманды получают за счет возможностей микропрограммирования, точнее, совмещения в одной команде нескольких не противоречащих друг другу команд¹. Вместе с этим следует отметить, что выполнение отдельных действий внутри микрокоманды производится в определенной последовательности (см. табл. 3.4), обусловленной организацией электронных схем ЭВМ. Поэтому

Таблица 3.3

Код команды	Назначение команды
7402	Останов
7404	Логическое сложение содержимого НР (СМ) с кодом, набранным на клавишном регистре
7410	Безусловный пропуск (очередной команды)
7420	Пропуск, если содержимое ЛР (ЛРС) равно 1
7430	Пропуск, если содержимое ЛР (ЛРС) равно 0
7440	Пропуск, если содержимое НР (СМ) равно 0
7450	Пропуск, если содержимое НР (СМ) не равно 0
7460	Пропуск, если содержимое НР (СМ) равно 0 и (или) содержимое ДР (ДРС) равно 1
7470	Пропуск, если содержимое НР (М) и ДР (ДРС)

¹ Нельзя, конечно, заставить машину, например, одновременно осуществлять сдвиг содержимого НР (СМ) вправо и влево, однако вполне допустимо одновременное обнуление НР и ДР

Код команды	Назначение команды
7500	Пропуск, если содержимое НР (СМ) меньше 0 (если 0-й разряд НР содержит 1)
7510	Пропуск, если содержимое НР (СМ) больше или равно 0
7520	Пропуск, если содержимое НР (СМ) меньше 0 и (или) содержимое ДР (ДРС) равно 1
7530	Пропуск, если содержимое НР (СМ) больше или содержимое ДР (ДРС) равно 0
7540	Пропуск, если содержимое НР (СМ) меньше или равно 0
7550	Пропуск, если содержимое НР (СМ) больше 0
7600	Очистка НР (СМ)
7604	Занесение кода с клавишного регистра в НР (СМ)
7610	Безусловный пропуск и очистка НР (СМ)
7620	Пропуск, если содержимое ДР (ДРС) равно 1 и очистка НР (СМ)
7630	Пропуск, если содержимое ДР (ДРС) равно 0 и очистка НР (СМ)
7640	Пропуск, если содержимое НР (СМ) равно 0 и (безусловная) очистка НР (СМ)
7650	Пропуск, если содержимое НР (СМ) не равно 0 и (безусловная) очистка НР (СМ)
7700	Пропуск, если содержимое НР (СМ) меньше 0 и (безусловная) очистка НР (СМ)
7710	Пропуск, если содержимое НР (СМ) больше или (безусловная) очистка НР (СМ)
7750	Пропуск, если содержимое НР (СМ) больше 0 и (безусловная) очистка НР (СМ)

Таблица 3.4

Группы микрокоманд	Операция	Примечания
1-я группа	1. Очистка НР (СМ) и ДР (ДРС) 2. Образование обратного кода 3. Увеличение $ НР $ на 1 4. Сдвиг	Для «Электроники-100» выполняется лишь одна из них
2-я группа	1. Пропуски по проверкам $ НР < 0, НР > 0,$ $ НР = 0, ДР = 1,$ $ НР \neq 0$ и т.п. 2. 0 - НР (СМ) 3. Логическое сложение $ КР \vee НР $ 4. Останов	

эффект произвольного микропрограммирования может оказаться самым неожиданным. Программисту рекомендуется пользоваться таблицами проверенных микрокоманд (см. табл. 3.2 и 3.3, а также вкладку), а в сомнительных случаях прибегать к консультации инженеров-электриков, обслуживающих машину, и обязательно проверять работу сомнительной команды непосредственно на машине—на каком-либо тестовом примере.

Основу всех микрокоманд первой группы составляют элементарные операции очистки НР (СМ) и (или) ДР (ДРС), инвертирование содержимого любого из этих регистров, циклические сдвиги вправо и влево на 1 или 2 разряда и увеличение содержимого НР на единицу. Микропрограммированием этих операций удастся реализовать такие широко применяемые операции, как загрузка фиксированных кодов $+1_8$ и -1_8 (в машинном представлении чисел, принятых для «Электроники-100») в НР (СМ), образование дополнительного и обратного кодов, вызванных ранее в НР, умножение и деление на 2 и 4, холостую команду (не изменяющую содержимого НР, ДР, регистров внешних устройств и каких-либо ячеек МОЗУ).

Основное количество микрокоманд второй группы составляют различные операции анализа содержимого НР (СМ) и ДР (ДРС) на равенство или неравенство одного или обоих этих регистров нулю, на знак числа в НР — с пропуском очередной команды, если соответствующее условие выполнено. Во

второй группе микрокоманд, как и в первой, имеется возможность обнуления НР (СМ), что часто используется для расширения возможностей микрокоманд второй группы при различных операциях анализа (независимо от исходных условий при этом после анализа $|НР| = 0$). Отдельное место во второй группе микрокоманд занимают операция логического сложения содержимого НР с кодом, набранным на клавишном регистре, и останов.

Команды с кодом операции 7_8 используются также для управления работой блоков расширенной арифметики, схемным путем реализующих операции умножения и деления, нормализации и сдвигов. В структуре этих команд обязательно наличие «1» в 3- и 11-м разрядах. Список команд работы с ДАБ приведен в табл. 3.5 и приложении 1

Таблица 3.5

Код команды	Назначение команды	Пояснения
7405	Умножение	Содержимое РМЧ умножается на код, который находится в ячейке, следующей за командой обращения; старшие разряды произведения засылаются в НР (СМ), младшие — в РМЧ
7407	Деление	Содержимое НРРМЧ ¹ делится на код, который находится в ячейке, следующей за командой обращения; частное засылается в РМЧ, остаток — в НР; при переполнении в ДР засылается 1
7411	Нормализация	Содержимое НРРМЧ ¹ сдвигается влево, пока содержимое 0- и 1-го разрядов НР (СМ) не станет различным; содержимое СЧШ увеличивается на число сдвигов, содержимое ДР (ДРС) теряется, освободившиеся разряды РМЧ ² заполняются нулями
7413	Арифметический сдвиг влево	Содержимое ДРНРРМЧ ² сдвигается влево на число разрядов, код которого указан в ячейке, следующей за командой обращения, плюс один разряд; освобождающиеся разряды заполняются нулями, содержимое ДР (ДРС) теряется
7415	Арифметический сдвиг вправо	Содержимое НРРМЧ ¹ сдвигается вправо на число разрядов, код которого указан в ячейке, следующей за командой обращения плюс один разряд; в освободившиеся разряды заносится 0 — для положительного числа и 1 — для отрицательного
7417	Логический сдвиг вправо	Содержимое НРРМЧ ¹ сдвигается вправо на число разрядов, код которого указан в ячейке, следующей за командой обращения, плюс один разряд; освободившиеся разряды заполняются нулями, ДР (ДРС) не задействован
7421	Запись кода в РМЧ	Содержимое НР (СМ) переписывается в РМЧ, после чего НР (СМ) обнуляется
7441	Засылка содержимого СЧШ в НР (СМ)	
7501	Запись содержимого РМЧ в НР (СМ)	
7601	Очистка НР (СМ)	
7621	Очистка НР (СМ) и РМЧ	
7403	Запись содержимого НР (СМ) в СЧШ	Содержимое НР (СМ) переписывается в СЧШ с последующим обнулением НР (СМ) — только для «Электроники-100И»

¹ НРРМЧ — объединенный регистр НР (СМ) и РМЧ.

² Д.РНРРМЧ — объединенный регистр ДР, НР в РМЧ.

4. ПРИЕМЫ ПРОГРАММИРОВАНИЯ

4.1. ОБЩИЕ СВЕДЕНИЯ

Ознакомившись с представлением чисел в машине и системой её команд, можно приступить к составлению программ для данной конкретной машины. Однако есть определенные принципы и приемы написания программ, общие для любых вычислительных машин. Они подробно рассмотрены и обоснованы в ряде учебников и монографий (см., например, [2, 3]), поэтому далее будут кратко упоминаться и рассматриваться на примерах лишь те из них, которые наиболее часто используются при программировании или имеют определенную специфику в своей реализации на «Электронике-100».

Составлению собственно программы, т. е. написанию её в кодах машины или на одном из входных языков, всегда предшествуют этапы выбора алгоритма и численного метода решения конкретной задачи (если это расчётная задача) и составления структурной схемы (или блок-схемы) будущей программы. Вопросы выбора алгоритма рассматриваются в гл. 7 — 9; для данной главы более существенны вопросы составления структурной схемы программ.

Структурная схема программы позволяет в достаточно общем виде представить последовательность операций, приводящих к реализации выбранного алгоритма. Вместе с этим структурная схема предопределяет все дальнейшее построение программы с учётом специфики системы команд данной машины. Таким образом, структурная схема программы должна быть и достаточно полной, и достаточно подробной. Только для очень простых задач удается достичь этого в одной блок-схеме, сохранив её наглядность. Обычно же составление структурной схемы проходит ряд стадий, которые коротко можно охарактеризовать выражением «от сложного к простому» или «от общего к частному».

Вначале намечаются крупные блоки, целые функциональные узлы, последовательная работа которых (в соответствии и по аналогии с последовательным принципом выполнения операций в ЭВМ) приводит к решению поставленной задачи в соответствии с выбранным алгоритмом. Простота связей отдельных блоков между собой сразу делает обзримым весь путь решения и его узловые моменты. Затем для каждого из крупных блоков составляется все более подробная блок-схема с указанием всех связей, циклов обращений к подпрограммам и в другие блоки. (Для начинающих программистов рекомендуется доводить структурную схему отдельных блоков до такой степени подробности, при которой конкретизация каждого из мелких блоков при окончательном написании программы сводится к использованию 1—3 команд.) Следует отметить, что составление структурных схем не только облегчает работу самого программиста, но и намного упрощает взаимопонимание программистов (последнее особенно важно при разработке взаимосвязанных участков сложных программ несколькими специалистами). Кроме того, структурные схемы оказываются совершенно необходимыми при составлении отчетов (когда программа обычно уже работает и остается лишь объяснить другим, как это удалось сделать), и если программист с пользой для себя не позаботится об их составлении заранее, такая работа становится для него только неприятной тратой времени.

Структурные схемы получают наибольшую наглядность при использовании графической формы их представления, в связи с чем возникает вопрос об условных обозначениях.¹ Обычно каждый отдельный блок снабжается пояснительной надписью, заключаемой в рамку. Связи между блоками указываются стрелками. Линейные блоки (т. е. блоки, имеющие один вход и один выход и выстраивающиеся таким образом в линию) обычно имеют прямоугольную рамку. Блоки, в которых происходят разветвления в результате проверки некоторых логических или арифметических условий, имеют рамку другой формы — овальную, ромбовидную или многогранную; рядом с каждой из выходящих стрелок дополнительно указываются условия передачи управления в данном направлении.

Пояснительные надписи внутри блоков получают наибольшую эффективность, если выполнены на смысловом уровне — с полным и точным смысловым наименованием используемых ячеек, счетчиков, циклов и т. п. В отличие от смысловых формализованные надписи требуют обычно дополнительного списка обозначений, а их чтение неизбежно проходит при этом через дополнительный этап «вживания». Наряду с этим формализованная надпись рядом со смысловой часто упрощает дальнейшие надписи (особенно при разветвлениях), отчего повышается наглядность всей структурной схемы.

Кроме пояснительных надписей, блоки структурных схем полезно снабжать номерами. На первоначальной структурной схеме (с крупными блоками) все блоки снабжаются сквозной нумерацией; более детальные блоки, раскрывающие структуру крупных блоков общей схемы, в своей нумерации имеют двойной номер — вначале номер крупного блока и после точки — собственный порядковый номер мелкого блока (например, 2.13 означает 13-й мелкий блок 2-го крупного блока). Эти номера затем указывают на бланках программы рядом с первой командой очередного блока, за счет чего достигается простота отыскания нужных участков программы при её чтении и отладках. В этих же целях полезно проставить и на структурной схеме фактические адреса расположения блоков в МОЗУ.

Весь процесс составления программы можно рассматривать как синтез некоторой последовательности операций, приводящих к необходимому результату. Однако программисту необходимо владеть и искусством анализа уже написанных программ (участков программ). Подробно эти вопросы рассматриваются в конце данной главы.

4.2. ОСОБЕННОСТИ, СВЯЗАННЫЕ С ОДНОАДРЕСНОСТЬЮ МАШИНЫ И СТРАНИЧНОЙ ОРГАНИЗАЦИЕЙ ПАМЯТИ

Одноадресность машины приводит к значительной нагрузке на накопительный регистр (сумматор), с участием которого осуществляется большинство операций в машине. В связи с этим важное значение приобретает рациональная организация последовательности операций и использования НР (СМ) при реализации отдельных блоков программы.

¹ Решение этого вопроса для выходных (отчётных) документов узаконено ГОСТ 19427—74 и ГОСТ 19428-74.

Необходимо каждый раз стремиться к максимальному использованию данных, вызванных в НР (на СМ). Полезно помнить, что целый ряд команд (УП, ОП, БП, часть команд обращения к внешним устройствам и микрокоманд) не изменяет содержимого НР (СМ) и ДР (ДРС).

Часто для продолжения программы необходимо иметь обнуленными НР (СМ) и (или) ДР (ДРС). Для этой цели служат специальные команды (7200, 7300, 7100), однако заметную экономию может дать применение микропрограммирования — совмещение операций обнуления НР (СМ) и ДР (ДРС) с различными операциями анализа их содержимого, сдвига и др. (рекомендуется сравнить микрокоманды внутри каждой из групп в верхней и нижней части табл. 3.2. и 3.3).

Указанные приемы отчетливо можно проследить на разбираемых ниже программах («Ввод 1» и программе сдвига таблиц).

Определенную специфику в программирование в кодах «Электроники-100» вносит страничная организация памяти. Если программа вместе с константами и исходными данными занимает не более 128 (200₈) ячеек, т. е. уместается в пределах одной страницы, эта специфика не доставляет программисту никаких неудобств и может вообще остаться незамеченной, как если бы с самого начала предположить размер МОЗУ ограниченным таким объемом. В иных случаях для перехода со страницы на страницу, для использования массива данных, расположенных на других страницах, для обращения к подпрограммам и в ряде других случаев используют косвенное адресование и особенности нулевой страницы — возможность непосредственного адресования к ней (и прямого, и косвенного) из любого места МОЗУ. При этом наиболее общий прием разбиения программы на отдельные, по возможности минимально связанные, блоки приобретает дополнительный смысл экономии памяти.

Обычно написание программ начинают с ячейки 0200₈ (с начала первой страницы), оставляя нулевую страницу для размещения наиболее употребительных констант и адресов подпрограмм. (Следует отметить, что абсолютная нумерация страниц памяти — а их в МОЗУ содержится с 0 до 37₈ — практически никакого значения для программиста не имеет — важно лишь нулевая эта страница или нет.) Собственно команды удобно располагать с начала страницы — от меньших номеров к большим, в соответствии с тем, как их потом будет использовать управляющая часть ЭВМ. С самого конца страницы (от адреса 177₈ на текущей странице и далее к меньшим адресам) по мере необходимости размещают константы и адреса, используемые командами, которые располагаются на данной странице. Такая структура использования памяти внутри страницы не требует предопределения границы команд и констант, что уменьшает потери ячеек памяти при первых «прикидках» программы. По мере написания программы количество неиспользованных ячеек на странице сокращается, и если программа достаточно велика, возникает необходимость перехода (передачи управления) на следующую страницу, обычно в её начало. В этом случае используют один из следующих приёмов¹.

1. В одной из ячеек (поля констант) данной страницы запасают адрес начала следующей страницы, а последней командой осуществляют безусловную передачу управления с косвенным адресованием к этой ячейке.

2. Если места на текущей странице нет и некуда записать адрес, можно «пожертвовать» для этой цели одну из ячеек на нулевой странице. Затем, как и в предыдущем случае, используют команду БП с косвенным адресованием.

Структура команд обращения к памяти и страничная организация МОЗУ определяют и рациональную форму бланков для программирования. Для каждой команды и константы удобно иметь её адрес на текущей странице и абсолютный адрес. Кроме того, каждая запись или, по крайней мере, группа кодов обычно нуждается в пояснениях на смысловом уровне и в «привязке» к блок-схеме программы. И наконец, важное значение имеет наглядность и удобство чтения записей, особенно в части обозначения различных переходов и передач управления. Все названные требования были учтены при разработке формы бланка, представленной на рис. 4.7.

Колонки для абсолютных адресов и адресов на странице частично заполнены типографским способом и сразу выделяются среди других, в то же время, отличаясь друг от друга шириной колонки (на четыре и на три цифры соответственно). Колонка для кодов команд (или констант) непосредственно примыкает к широкому полю для пояснений на смысловом уровне, а соседняя (с другой стороны) колонка с адресом на странице имеет меньшую ширину, благодаря чему как бы отступает на второй план. Зрительную разрядку создает узкое пустое (нелинованное) поле между колонками для абсолютного адреса и адреса на странице; в то же время это поле эффективно используется для указания номеров блоков программы в соответствии с её структурной схемой.

Слева оставлено поле для указания передач управления. Как известно, именно на такие команды приходится наибольшее число ошибок. Поэтому на указанном поле рекомендуется в процессе программирования обозначать стрелками все передачи управления ещё до написания самой команды; пропуски очередной команды, которые совершаются при определенных условиях, удобно отличать штриховыми стрелками и пояснительными надписями.

Стандартный лист разграфлен на три октавы, что обеспечивает достаточную высоту для цифр адресов и кодов. Внизу листа остается достаточно места для размещения в случае необходимости нескольких дополнительных команд. В верхней части листа отведено место для наименования задачи и сквозной нумерации бланков программы.

Ниже приводятся примеры использования таких бланков при написании программы.

¹. См. так же § 5.5

4.3. ОРГАНИЗАЦИЯ РАЗВЕТВЛЕНИИ

Практически в каждой программе встречаются места, где в зависимости от тех или иных условий должны выполняться различные действия. На структурной схеме программы таким участкам соответствуют различные ветви (рис. 4.1).

Типичная структура организации разветвления, обусловленная одноадресностью «Электроники-100», представлена на рис. 4.2. Запасая команду БП на начало программы обслуживания одной из ветвей (ветви 1 на рис. 4.1), перед ней располагают команду, которая организует пропуск (т.е. обход) этой команды при выполнении определенных условий; таким образом, управление передается в следующую ячейку памяти, начиная с которой располагают программу обслуживания другой ветви (ветви 2 на рис. 4.1). Все условия разветвления обычно сводят в две большие группы — арифметические (например, достигнуто ли нужное число обращений в счетчик) и логические (обнулен ли некоторый признак¹, передано ли управление в определенное

место программы, готово ли внешнее устройство к общению с процессором и др.)

¹ Точнее, по-видимому, сказать: «находится ли значение признака в пределах логического нуля».

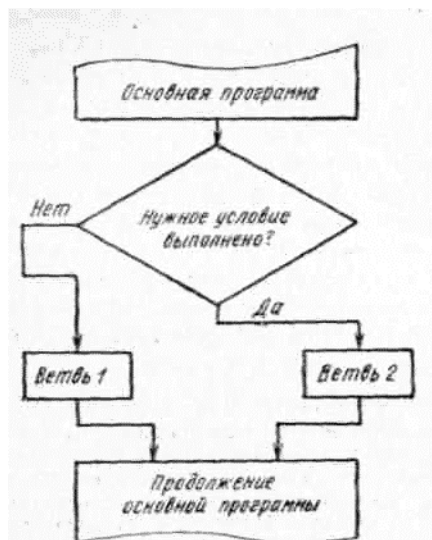


Рис. 4.1. Структурная схема разветвляющейся программы.

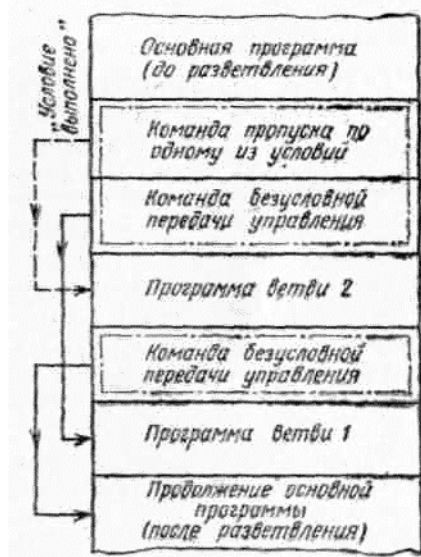


Рис. 4.2. Типичная структура программной реализации разветвления. Команды, организующие собственно разветвление выделены штрихпунктиром; команда безусловной передачи управления в конце программы ветви 2 обеспечивает обход программы ветви 1, если работала ветвь 2.

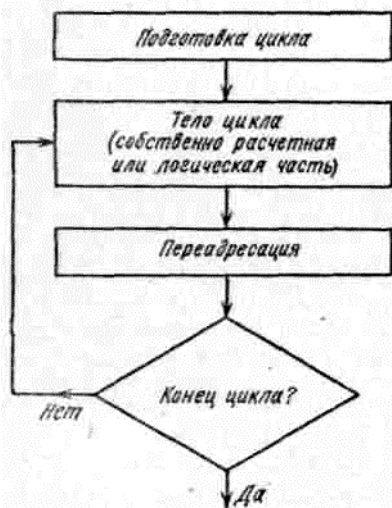


Рис. 4.3. Типичная структурная схема циклической программы.

Однако в системе команд «Электроники-100» нельзя выделить группы команд, однозначно реализующие пропуск только по всем арифметическим условиям. Для пропуска очередной команды по одному из условий используются следующие команды: УП (код операции 2₈) реализует пропуск по одному из арифметических условий; опрос флага (структура команды *бтп1*, где *тп* — восьмеричный код, т. е. номер внешнего устройства) реализует пропуск по одному из логических условий и большая группа микрокоманд второй группы, среди которых есть команды для реализации и тех и других условий.

Следует отметить, что первые две группы команд не затрагивают содержимого НР(СМ) и ДР(ДРС). При использовании микрокоманд для организации разветвления машина производит анализ содержимого ДР(ДРС) и (или) некоторых разрядов НР(СМ), вследствие чего разветвлению должны предшествовать операции по формированию (или вызову) соответствующих кодов в эти регистры; если содержимое НР(СМ) не используется для дальнейших операций, можно совместить в одной команде (за счет применения микропрограммирования) анализ и последующее обнуление НР (СМ).

Примеры организации разветвлений будут даны в § 4.4, 4.9, 4.10.

4.4. ЦИКЛИЧЕСКИЕ ПРОГРАММЫ

В тех случаях, когда требуется несколько раз выполнить одинаковые действия над некоторым изменяющимся составом исходных данных, экономным способом организации программ является использование циклов. Общий вид структурной схемы циклической программы приведен на рис. 4.3.

Основная группа команд, с помощью которых и осуществляются необходимые логические и вычислительные операции, составляет *тело цикла*. Эти команды в цикле выполняются многократно, за счет чего обычно и происходит основная экономия команд. Структура тела цикла определяется спецификой конкретной задачи.

Для того чтобы одни и те же команды тела цикла могли оперировать каждый раз с новыми данными, необходимо после каждого прохода изменять адреса исходных данных. Эту работу выполняет блок *переадресации* (рис. 4.3). Исходные данные предварительно запасают в последовательных ячейках памяти (в виде таблицы), а в командах тела цикла используют косвенное адресование к некоторой рабочей ячейке, куда при работе блока подготовки цикла (рис. 4.3) заносится начальный адрес таблицы.

После каждого прохода тела цикла блок переадресации производит увеличение или уменьшение адреса, изменяя содержимое рабочей ячейки на необходимую величину (шаг). Если требуется увеличивать каждый раз адрес на 1, это может быть сделано одной командой УП, однако более эффективным в таком случае является использование индексных ячеек. (Достаточно записать в индексной ячейке начальный адрес таблицы, уменьшенный на 1, и применять её далее в качестве рабочей ячейки адреса в теле цикла—при каждом косвенном обращении содержимое индексной ячейки сначала увеличивается на 1, а затем используется в качестве адреса).

Важно отметить, что при переадресациях с помощью индексных ячеек и команд УП блок переадресации не затрагивает содержимого НР (СМ) и ДР (ДРС). В более общем случае переадресация происходит с использованием этих регистров,

и приходится запоминать их содержимое по выходе из тела цикла в специально отводимых рабочих ячейках для последующего восстановления по выходе из блока переадресации.

Блок окончания цикла является типичным разветвлением. Используются два способа выхода из цикла. Если число повторений известно заранее, то выход из цикла организуется по переполнении счетчика, с использованием команды УП. В других случаях выход из цикла организуют с помощью микрокоманд второй группы по достижении нужных условий в НР (СМ) или ДР (ДРС).

Блок подготовки цикла организует необходимые начальные состояния рабочих ячеек счетчика цикла, адресов исходных данных, используемых констант и т. п. Следует отметить, что расположение блока подготовки цикла до начала самого цикла, как и более общий прием вынесения всех настроечных операций в начало программ, создает большие удобства при отладках — для повторения работы какого-то участка программы с самого начала достаточно передать управление в его начало независимо от того, работала программа раньше или нет¹

В качестве примера можно разобрать процесс составления программы для подсчета суммы массива кодов. Для большей конкретности будет полагаться, что массив располагается, начиная с адреса 1000₈, и содержит 90₁₀ кодов. Программа должна занимать место с адреса 0400₈ (т. е. с начала второй страницы МОЗУ). Результат должен быть получен в ячейке с адресом 0150₈.

Разработка программы начинается с определения точного алгоритма и составления подробной структурной схемы. В данном случае основное расчётное соотношение можно представить в виде

$$S = \sum_{i=i_{нач}}^{i_{конеч}} x_i$$

где S — искомая сумма; i — текущий адрес; $i_{нач}$ ($= 1000_8$) — начальный адрес массива суммируемых кодов; x_i — текущий код (содержимое текущей, i -й, ячейки).

Таким образом, расчётный алгоритм здесь сводится к единственной операции сложения (других, собственно расчётных действий не требуется). Необходимо лишь позаботиться, чтобы указанная операция тела цикла могла оперировать с нужными данными. Кроме того, важно определить, достаточно ли для всех вычислений в теле цикла и переадресаций только НР(СМ) или для промежуточных результатов требуются дополнительные рабочие ячейки. Вновь рассматривая с этой точки зрения тело цикла, можно заметить, что собственно суммирование вполне можно осуществлять на НР(СМ), добавляя к уже накопленному на нем результату очередное слагаемое. (Естественно, надо будет проследить, чтобы перед входом программы в тело цикла $|НР| = 0$, иначе результат будет искажен.)

Так как текущий адрес принимает подряд все значения от $i_{нач}$ до $i_{конеч}$, меняясь каждый раз на 1, переадресацию легко осуществлять с помощью команды УП либо с использованием индексных ячеек — в обоих случаях содержимое НР(СМ) при этом не затрагивается. (Для большей наглядности самого процесса переадресации далее будет использована команда УП, хотя использование индексной ячейки сэкономило бы одну команду.)

Теперь почти определились структура и содержание двух блоков — тела цикла и переадресации: в теле цикла к содержимому НР(СМ) достаточно прибавлять каждый раз значение переменной x_i очередной, i -й, ячейки, а в блоке переадресации сам адрес i каждый раз увеличивать на 1. Последний шаг здесь — связать эти два блока между собой. Вновь обращаясь к телу цикла, необходимо проследить возможность реализации нужных действий с использованием системы команд «Электроники-100».

Среди основных команд машины в данном случае подходит операция сложения (код операции 1₈):

$$|НР| + |y| \rightarrow НР,$$

остается уточнить лишь способ задания адреса y , откуда выбирается очередное слагаемое. Вариантов всего два: использовать в адресной части команды СЛ прямое или косвенное адресование. При прямом адресовании одна и та же неизменяемая команда из тела цикла обращалась бы за данными в одну и ту же ячейку. При косвенном адресовании та же команда обращается в некоторую ячейку памяти лишь за адресом, по которому расположены данные: меняя содержимое этой ячейки в блоке переадресации, можно использовать все новые данные, расположенные в соседних ячейках.

Таким образом, окончательно определилось, что тело цикла использует команду СЛ с косвенным адресованием к некоторой (рабочей) ячейке, а блок переадресации каждый раз увеличивает содержимое этой ячейки (т. е. фактический адрес, используемый в теле цикла) на единицу. Одновременно выяснилось, что требуется специальная рабочая ячейка для текущего адреса и в эту ячейку в блоке подготовки цикла необходимо занести $i_{нач}$ (в данном случае код 1000₈). Очевидно, соответствующая константа должна иметься отдельно на текущей или нулевой странице МОЗУ.

Так как в данном случае заранее известно, сколько раз должны повториться вычисления внутри цикла, выход из цикла организуется по переполнении счетчика, т. е. блок конца цикла содержит две команды: УП и БП. Для счетчика требуется отдельная рабочая ячейка, куда в блоке подготовки цикла требуется занести дополнительный код необходимого числа повторений цикла.

По выходе из цикла результат, накопленный на НР (СМ), надо заслать в ячейку результата и остановить машину. Таким образом, полная структурная схема программы приобретает вид, показанный на рис. 4.4. Все блоки пронумерованы сверху вниз, в порядке их действия в программе (в сильно ветвящихся программах целесообразно пронумеровать все блоки вдоль одной ветви, затем, вернувшись к разветвлению, первому от конца этой ветви, продолжить нумерацию до конца другой ветви, вернуться к предыдущему разветвлению и т. д.). Блок 2а обведен штрихом, так как он может не понадобиться, если при выходе из блока 2 окажется, что $|НР| = 0$.

Следующий шаг состоит в распределении памяти, определении и запасании необходимых констант.

уже определена:

$$i_{нач} = 1000_8.$$

опр
изв
пол
пер

$$\begin{array}{r} 90 \overline{) 8} \\ \underline{8} \\ 10 \\ \underline{8} \\ 2 \end{array} \quad \begin{array}{r} 8 \\ \underline{11} \\ 8 \\ \underline{3} \\ 2 \end{array} \quad \begin{array}{r} 8 \\ \underline{1} \\ 1 \end{array}$$

$90_{10} = 132_8$

вязанную с необходимым числом повторения цикла, требуется рассчитать, код числа и образовав от него дополнительный код. Для этого в соответствии с евода кодов к новому основанию исходное число делится на это основание до стного и остатка, частное вновь делится на основание и т. д., а затем все остатки к началу;

$$\begin{array}{r} 90 \overline{) 8} \\ \underline{8} \\ 10 \\ \underline{8} \\ 2 \end{array} \quad \begin{array}{r} 8 \\ \underline{11} \\ 8 \\ \underline{3} \\ 2 \end{array} \quad \begin{array}{r} 8 \\ \underline{1} \\ 1 \end{array}$$

$90_{10} = 132_8$

¹ Менее удобным в этом отношении является прием во

него участка программы вслед за его прохождением.

Для получения дополнительного кода вычитаем полученный восьмеричный код из 10.000_8

$$\begin{array}{r} - 10000 \\ 0132 \\ \hline 7646 \end{array}$$

Требуемая константа равняется, таким образом, 7646_8 .

При распределении памяти константы и рабочие ячейки обычно располагают в конце страницы, а команды программы — в начале. В данном случае таблица распределения памяти имеет вид, представленный на рис. 4.5¹

Теперь уже можно писать сам текст программы (собственно команды). Эта работа начинается с заполнения абсолютных адресов и адресов на странице на бланках для программирования. Первоначально берут два бланка — один нумеруют с начала страницы (с номера на странице 000), второй — с конца (чтобы последняя ячейка имела номер на странице 177_8).

В соответствии с принятым распределением памяти на второй бланк заносятся необходимые константы и отмечаются рабочие ячейки (чтобы рабочие ячейки зрительно выделялись, их лучше штриховать, а при подготовке программы заполнять нулями² рис. 4.6).

С начала первого бланка начинают писать собственно команды. Рекомендуется в начале программы и между блоками оставлять по 1—3 запасных

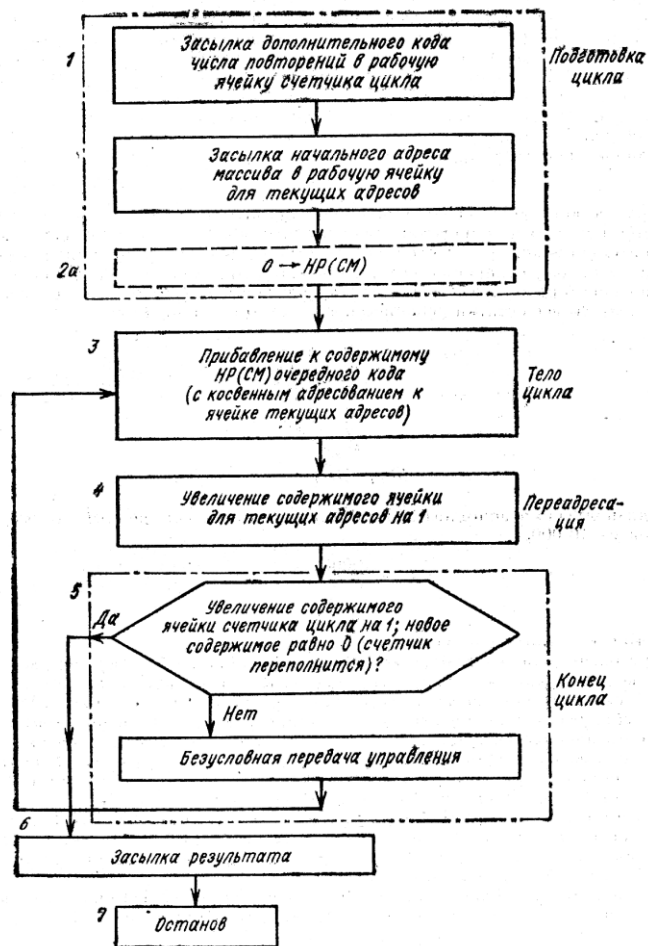


Рис. 4.4. Полная структурная схема циклической программы для подсчета суммы кодов

Адрес	Содержимое	Начало страницы
0400	Команды программы	
...	...	
0574	Рабочая ячейка текущего адреса	
0575	Рабочая ячейка счетчика цикла	
0576	Начальный адрес массива	
0577	Дополнительный код числа повторений	Конец страницы

Рис. 4.5. Распределение памяти для программы подсчета суммы кодов.

6	6	
7	7	
0570	170	
1	1	
2	2	
3	3	
4	4	Раб. ячейка текущего адреса*
5	5	Раб. ячейка счетчика цикла
6	6	1000 - Начальный адрес массива
7	7	7646 = -132 ₈ - Константа числа повтор.

Рис. 4.6. Заполнение ячеек для констант и рабочих ячеек программы. Б8

¹ Для большей компактности программы константы и рабочие ячейки могут быть приближены к командам после того, как определится конечный адрес цепочки команд, но сам «текст» программы должен быть после этого дополнительно скорректирован в адресной части команд обращения к памяти в соответствии с новыми адресами рабочих ячеек и констант.

² При работе программы содержимое рабочих ячеек изменяется независимо от их исходного состояния. Нулевое (фиксированное) исходное содержимое облегчает анализ (поиск ошибок) проработавших участков программы

ячейки для последующих возможных изменений и модификаций программы, упрощения отладочных операций и т. д. Обычно такие ячейки заполняют командами 7000 («нет операции»), никак не влияющими на содержимое НР(СМ) и ДР (ДРС) и исполняющимися за один машинный цикл.

Первым блоком в структурной схеме стоит пересылка дополнительного кода числа повторений в рабочую ячейку счетчика цикла. Собственно засылка может быть осуществлена командой ПС, однако необходимый код предварительно должен быть вызван в НР(СМ).

Для вызова кода можно использовать операцию СЛ, но НР(СМ) предварительно должен быть обнулен (так как при передаче управления на начало программы | НР | в общем случае произвольно). Таким образом, первый блок разворачивается в следующую цепочку команд:

7000 Резерв
7200 0 → НР
1377 Содержимое ячейки 177 на текущей странице (требуемый код) → НР 3375 Код из НР → ячейку 175 на текущей странице; 0 → НР.

Структура второго блока совершенно аналогична — не требуется лишь обнуления НР(СМ), так как это автоматически выполнилось последней командой предыдущего блока. По этой же причине отпадает надобность в блоке 2а.

Следует обратить внимание на то, что при формировании второй триады команд обращения к памяти используются правила, изложенные в гл. 3, при этом заметную помощь оказывают адреса ячеек на странице, выделенные на бланке для программирования в отдельную графу.

Конечный вид текста программы в записи на бланке для программирования показан на рис. 4.7.

Замечание 1. В силу использования косвенного адресования к данным (в теле цикла) их место расположения в МОЗУ безразлично. Поэтому иногда оказывается рациональным выносить даже небольшие массивы данных, участвующих в цикле, на отдельную страницу, сохраняя максимум места на текущей странице для команд программы.

Замечание 2. Нетрудно видеть, что при организации циклической программы тело цикла «обрастает» дополнительными командами для подготовки, окончания цикла и переадресации; кроме того, ряд ячеек занимает для хранения констант и под рабочие ячейки. Поэтому организовывать цикл обычно рационально при числе повторений цикла, большем 3—4, либо в случае относительно сложной структуры тела цикла.

Замечание 3. Эффективность программы резко повышается, если вычисления удастся организовать в виде двойных (многократных) циклов. В этом случае весь внутренний цикл, вместе с блоками подготовки и окончания цикла, входит в тело внешнего цикла. Часть ячеек, где ранее хранились константы внутреннего цикла, используются в качестве рабочих ячеек внешнего цикла. При подготовке внешнего цикла в них заносятся начальные адреса таблиц, в которые помещены сведения о местоположении нужных данных, а при подготовке внутреннего цикла к этим ячейкам происходит теперь косвенное адресование. Пример двойного цикла приводится в § 4.9.

Суммирование кодов

Наименование (код) задачи

Лист № 1

Передача управления	Абсолютный адрес	№ блока	Адрес на стр.	Команда (содержимое)	Пояснение
	0400		000	7000	Резерв
	1	1	1	7200	
	2		2	1377	Засылка-132 в счетчик цикла
	3		3	3375	
	4	2	4	1376	Засылка $i_{нач}$ в рабочую
	5		5	3374	ячейку
	6		6	7000	Резерв
	7		7	7000	
	0410	3	010	1774	Суммирование (с косвенным адресованием)
	1		1	7000	
	2	4	2	2374	Переадресация
Да	3	5	3	2375	Конец цикла?
Нет	4	6	4	5210	
	5		5	7000	
	6	7	6	3150	Засылка результата
	7	8	7	7402	Останов
	0		0		

0573	173		
4	4		Рабочая ячейка текущего адреса
5	5		Рабочая ячейка счетчика цикла
6	6	1000	$i_{нач}$
7	7	7646	Дополнительный код числа повторений

Рис. 4.7. Полный текст программы для суммирования кодов.

4.5. ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ

В тех случаях, когда одни и те же вычисления должны выполняться в разных местах программы, собственно расчётные операции выделяют в подпрограмму, а в соответствующие места программы заносят лишь команды обращения к этой подпрограмме. Как указывалось в гл. 3, первая ячейка поля, отводимого для подпрограммы, используется для (автоматической) записи адреса продолжения основной программы и таким образом является рабочей (команды подпрограммы располагаются со следующей ячейки).

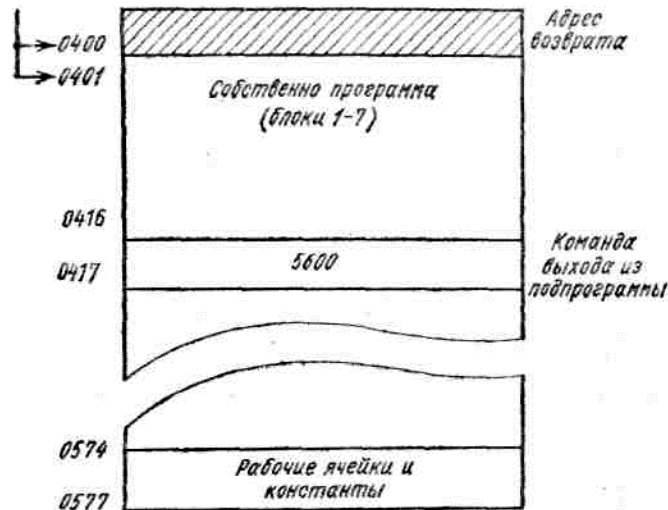


Рис. 4.8. Оформление программы рис. 4.7 в виде подпрограммы.

К первой ячейке подпрограммы обращаются косвенно по команде БП при выходе из подпрограммы. Таким образом, для того чтобы участок программы превратить в подпрограмму, необходимо одну ячейку перед началом этого участка выделить для адреса возврата (адрес этой ячейки будет в дальнейшем являться адресом подпрограммы), а в конце участка добавить команду БП с косвенным адресованием по начальному адресу.

Так, программа подсчета суммы кодов, рассмотренная в предыдущем параграфе, может быть обращена в подпрограмму, если первую ячейку использовать как начальную ячейку поля подпрограммы¹, а команду 7402 (останов) заменить командой выхода из подпрограммы 5600 (БП с косвенным адресованием к 0 ячейке на текущей странице, рис. 4.8). Адрес подпрограммы будет 0400. Если код 0400₈ занесен на нулевой странице, например в ячейке 0176, то к этой подпрограмме можно обращаться из любого места МОЗУ по команде 4576.

Так как подпрограмма используется в нескольких местах основной программы, важным является вопрос о состоянии НР (СМ) при входе в подпрограмму и при выходе из нее, а также вопрос о размещении исходных данных и результатов.

Когда НР(СМ) не используется для передачи некоторой дополнительной информации при входе в подпрограмму и при выходе из нее, рационально обеспечить независимость работоспособности подпрограммы при произвольном содержимом НР (СМ) в момент входа в подпрограмму и обнуление НР(СМ) перед выходом из нее (а в некоторых случаях сохранение содержимого НР(СМ) таким же, как при входе в подпрограмму). В более общих случаях содержимое НР(СМ) в момент входа в подпрограмму может использоваться в качестве исходных данных, настроечных параметров и т. п. Обычно первой операцией в достаточно сложной подпрограмме при этом бывает запись содержимого НР(СМ) в рабочую ячейку внутри поля подпрограммы. В ряде случаев результат работы подпрограммы в виде единственного кода вызывается (остается) в НР (на СМ) перед возвращением в основную программу. Иногда в качестве рабочих ячеек для засылки исходных данных и хранения результатов используют ячейки вне основного поля подпрограммы (обычно на нулевой странице).

Наконец, для передачи исходных данных в подпрограмму вместе с НР(СМ) может употребляться и дополнительная ячейка памяти, следующая непосредственно за командой обращения к подпрограмме. Так как адрес этой ячейки автоматически заносится в первую ячейку поля подпрограммы (для обычного использования в качестве адреса возврата в основную программу), подпрограмма сразу получает в своё поле адрес расположения дополнительных исходных данных. Для того чтобы возвратиться к продолжению основной программы (обойдя ячейку с дополнительными данными), содержимое первой ячейки поля подпрограммы в конце её работы увеличивают на единицу (по команде УП), а затем осуществляют обычный выход из подпрограммы по команде БП с косвенным адресованием к этой ячейке. Таким образом, в рассматриваемом случае первая ячейка поля подпрограммы используется дважды: вначале для хранения адреса дополнительных данных, а затем для формирования адреса возврата и возвращения в основную программу. (Такой прием, обусловленный одноадресностью машины, применяется, например, при организации стандартной подпрограммы умножения двух чисел.)

Иногда для удобства организации основной программы в дополнительной ячейке памяти, следующей непосредственно за

¹ Резерв в начале участка программы здесь оказался очень кстати.

командой ОП, размещают не сами дополнительные данные, а лишь указывают их адрес. В этом случае до самих исходных данных в подпрограмме приходится добираться с помощью двойного косвенного адресования, используя дополнительную рабочую ячейку. Обратившись первый раз косвенно по «адресу возврата», извлекают адрес исходных данных и переписывают его в рабочую ячейку, и лишь при косвенном адресовании к этой рабочей ячейке дополнительные исходные данные уже могут быть доступны подпрограмме.

Специальный прием приходится также применять, если подпрограмма переходит на следующую страницу МОЗУ. При этом адрес возврата в основную программу остается (вместе с началом подпрограммы) на предыдущей странице — известен лишь адрес ячейки (начальный адрес подпрограммы), где нужный адрес записан. Как и в предыдущем случае, должно быть организовано двойное косвенное адресование — теперь перед выходом из подпрограммы. Вначале в дополнительную рабочую ячейку с косвенным обращением по начальному адресу подпрограммы с предыдущей страницы МОЗУ вызывается (переписывается) адрес возврата в основную программу, а затем уже осуществляется выход из подпрограммы по команде БП с косвенным адресованием к этой рабочей ячейке.

Другой способ состоит в том, что последние команды подпрограммы, включая команду возвращения в основную программу, располагают на той же странице МОЗУ, что и начало подпрограммы, вынося на другие страницы все неуместившиеся команды «средней» части подпрограммы и команды перехода к исходной странице МОЗУ.

Наиболее общие расчётные операции, встречающиеся в инженерных приложениях «Электроники-100», разработаны до уровня стандартных подпрограмм. Сюда относятся подпрограммы умножения и деления с обычной и удвоенной точностью, вычисление тригонометрических, показательных и логарифмических функций и т. п. Описанием этих подпрограмм комплектуется документация на каждую ЭВМ.

Обычно все подпрограммы написаны в адресах, начинающихся с адреса 0200₈. Вместе с этим многие подпрограммы укладываются в пределах одной страницы МОЗУ, так что все рабочие ячейки и константы находятся на той же странице, что и команды подпрограммы, а адресование за пределы текущей страницы при работе самой подпрограммы осуществляется только через адрес возврата, автоматически заносимый по начальному адресу при обращении к подпрограмме. Поэтому такие подпрограммы могут без всяких изменений размещаться со сдвигом на любое целое число страниц МОЗУ.

Очевидный путь размещения и использования стандартных подпрограмм без переделки состоит в следующем. При компоновке общей программы для необходимых стандартных подпрограмм выделяются отдельные страницы МОЗУ. Начальные адреса стандартных подпрограмм выносятся на нулевую страницу. Основная программа пишется на свободных страницах памяти, а при необходимости может занимать и ячейки, оставшиеся неиспользованными на страницах, которые заняты стандартными подпрограммами. Слабая сторона этого приема состоит в том, что перестает соблюдаться общая система размещения команд, констант и рабочих ячеек на каждой отдельной странице МОЗУ. При достаточно больших и сложных программах, когда дело доходит до экономии ячеек памяти и использования всех оставшихся «окон», такое нарушение системы построения программы может существенно затруднять её отладку.

Когда программист свободен в выборе места для размещения (разрабатываемых) подпрограмм, их рационально располагать в конце страниц перед константами и рабочими ячейками, короткие подпрограммы — в начале страниц либо занимать под группу примыкающих друг к другу подпрограмм целые страницы памяти. Последний способ размещения подпрограмм оказывается особенно удобным, если общая логика построения всей программы сводится к последовательному использованию набора подпрограмм. Тогда в программе выделяется некоторая управляющая часть, состоящая из набора обращений к подпрограммам и ряда вспомогательных действий (типа пересылки исходных данных и результатов, засылки и проверки признаков, организации разветвлений и т. п.). В управляющей части, которая фактически и реализует весь основной алгоритм работы программы, обычно обращаются к подпрограммам косвенно, и после разработки и размещения подпрограмм остается только заполнить их фактическими адресами заранее отведенные для этого ячейки.

4.6. ОБСЛУЖИВАНИЕ ПРЕРЫВАНИЙ

«Электроника-100», как и большинство универсальных ЦВМ, является машиной последовательного действия, т. е. каждая отдельная операция в машине начинается только после завершения предыдущей операции. Так же последовательно, команда за командой, происходит работа каждой отдельной программы. Пока в машине выполняются логические операции и расчеты, связанные с решением каждой отдельной задачи, такая организация работы ЭВМ не вызывает особых неудобств и даже является достаточно эффективной в плане использования основных электронных узлов машины.

Эффективность работы ЭВМ последовательного действия начинает заметно снижаться, когда по ходу решения необходимо произвести операции ввода (и/или вывода) данных, обычно с использованием относительно медленных электромеханических устройств. Разрыв в быстродействии процессора и устройств ввода—вывода может достигать нескольких десятичных порядков (для «Электроники-100» см. табл. 1.1). Если при этом сохранить последовательный принцип работы, то на время выполнения операций ввода—вывода вся вычислительная система снизит скорость работы до нескольких десятков операций в секунду (вместо сотен тысяч в секунду при работе процессора). Для преодоления этого «узкого» места в современных машинах принимают специальные меры. Так, например, в больших машинах устройства ввода—вывода снабжаются индивидуальными буферными блоками памяти значительной емкости — иногда в сотни машинных слов. Такой буфер может быстро обмениваться очередной порцией информации с процессором, а затем работать со своим устройством ввода или вывода в нужном ему темпе.

Для повышения оперативности работы «Электроники-100» используется система прерывания производимых вычислений. Любая программа в принципе может быть приостановлена (прервана) по сигналу извне для обмена данными с устройством, готовым в данный момент участвовать в таком обмене. При этом удачно сочетается общий принцип минимума аппаратуры в мини-машине (внешнее устройство снабжается триггером-сигнализатором готовности и буферным регистром всего на одно машинное слово) с максимальным повышением эффективности его использования (на обмен с внешним устройством затрачивается лишь фактически необходимое время, все остальное время процессор может быть занят полезными вычислениями до нового запроса от внешнего устройства на обмен очередным словом). Однако, как и в других подобных случаях, повышение эффективности работы оборудования мини-машины достигается за счет определенных программных приемов¹.

¹ Можно также считать, что за счёт квалификации программиста либо дополнительных программных ограничений — в соответствии с общим мировоззрением или проводимой технической политикой. Следует сразу отметить, что применение этих приемов, как и любых универсальных программных блоков, становится целесообразным лишь при достаточно широком круге обслуживаемых задач — в ином

Использование системы прерывания может дать очевидные преимущества, когда в ведении вычислительной системы находится (или могут находиться) сразу несколько задач, решаемых по очереди или по мере их появления (по запросу извне). Для упорядочения работы в такой системе устанавливается та или иная дисциплина обслуживания заявок и, по необходимости, часть памяти машины занимается программой-диспетчером.

Основные дисциплинарные ограничения, связанные с появлением диспетчера, будут рассмотрены в гл. 8, здесь же важно отметить следующее. Так как любая задача находится на обслуживании у диспетчера и решается при разрешенном прерывании, а момент появления очередной заявки на решение другой задачи может быть случайным, каждой задаче в общем случае приписывается тот или иной приоритет для установления очередности прохождения задачи в конфликтной ситуации. Далее, так как в ведении диспетчера находится несколько задач и любая из них при своем появлении может прерывать любую уже начавшуюся, должна быть предусмотрена достаточно большая глубина прерывания

Как уже упоминалось, «Электроника-100» имеет два режима прерывания — программный и информационный (режим «разрыва данными»). Большая часть диспетчерских функций осуществляется с использованием программного прерывания. Типовые действия при обслуживании программного прерывания сводятся к запоминанию состояния основных регистров машины, обнаружению источника прерывания и идентификации запрашиваемой задачи.

1. Запоминание состояния основных регистров машины частично автоматизировано — в момент прерывания (см. § 3.4) имитируется выполнение команды 4000_8 , т. е. в ячейке 0000 запоминается адрес продолжения прерванной задачи, а управление передается в ячейку 0001 . С ячейки 0001 должна начинаться программа, реализующая остальные необходимые действия. Таким образом, к моменту получения управления диспетчером состояние счетчика команд (СК.) в момент прерывания оказывается запомненным автоматически, состояние регистра адреса (РА) и буферного регистра (РБ) роли не играет, так как прерывание происходит только после завершения очередной команды. Остается запомнить состояние НР(СМ) и ДР(ДРС), так как эти регистры могут использоваться (и, следовательно, изменять своё состояние) при работе программ диспетчера. Для хранения | НР | и | ДР | выделяются две ячейки памяти (в общем поле, предоставляемом для диспетчерских функций), куда и переписывается содержимое НР(СМ) и ДР(ДРС) по следующей схеме, реализуемой тремя командами:

Запись НР в ячейку «а» с очисткой НР
Циклический сдвиг влево на один разряд
Запись НР в ячейку «в»

Первой командой запоминается состояние НР(СМ) в ячейке «а» и подготавливается возможность выделения и запоминания |ДР| — при циклическом сдвиге содержимое ДР(ДРС) переходит в младший разряд очищенного предыдущей командой НР(СМ) и следующей командой запоминается в ячейке «в».

2. Обнаружение источника прерывания основано на анализе состояния флагов внешних устройств. Обычно все внешние устройства в момент готовности устанавливают свой триггер готовности (флаг) в единичное состояние. Любой из флагов имеет выход на общую шину прерывания, инициируя появление последнего, если ранее выполнялась команда 6001 и других прерываний ещё не происходило (в ином случае запрос на прерывание «запасается» на флаге внешнего устройства). Программа состоит из последовательного опроса флагов всех задействованных внешних устройств и счета числа пройденных позиций. Сама команда опроса флага (типа $6mn1$, где mn — восьмеричный номер внешнего устройства), обеспечивающая пропуск очередной команды, если флаг установлен в единичное состояние, предопределяет необходимое разветвление программы.

Когда в составе системы имеется входной коммутатор дискретных сигналов (ВКДС), запросы внешних устройств на прерывание могут доводиться до процессора «в централизованном порядке». Для этого на шину прерывания получает выход только флаг ВКДС и схемным путем обеспечивается возможность установки его в единичное состояние, если перешел в единичное состояние любой из флагов внешних устройств, подключенных через ВКДС. Сами флаги объединяются в слова по 12 разрядов, которые могут вводиться в машину по командам обращения к ВКДС. Программа обнаружения источника прерывания в этом случае состоит в определении номера разряда, отличного от нуля.

3. Идентификация запрашиваемой задачи фактически состоит в выборе из таблиц её исходных данных (типа начального адреса и приоритета) в соответствии с состоянием счетчика позиций, пройденных при анализе цепочки флагов.

Структурная схема программы, реализующей указанные функции, приведена на рис. 4.9.

Дополнительные программные приемы применяются при выходе из прерывания. Особенности их построения и применения определяются в первую очередь числом возможных источников прерывания и логикой построения программы-диспетчера.

Если имеется единственный источник прерывания или при обслуживании очередного прерывания все остальные блокируются (т. е. в тех случаях, когда глубина прерывания равна единице), диспетчерские функции предельно упрощены и при выходе из прерывания необходимо лишь восстановить содержимое ДР (ДРС) и НР(СМ), разрешить прерывание и вернуть управление прерванной задаче. Эти функции осуществляются в следующей последовательности:

$0 \rightarrow \text{НР (СМ)}$
Вызов в НР (СМ) содержимого ячейки «в»
$0 \rightarrow \text{ДР (ДРС)}$ и циклический сдвиг вправо на один разряд (восстановление ДР)
Вызов в НР (СМ) содержимого ячейки «а» (восстановление НР)

случае плата за универсальность (в виде дополнительных затрат памяти и усложнения программы) оказывается неоправданной.

Вызов содержимого некоторой ячейки в очищенный НР (СМ) осуществляется по команде СЛ, при восстановлении ДР(ДРС) используется МК первой группы, возврат управления прерванной задачи осуществляется с косвенным адресованием к ячейке 0000, куда при прерывании был автоматически занесен адрес возврата (предполагается, что при обслуживании прерывания содержимое этой ячейки не было испорчено).

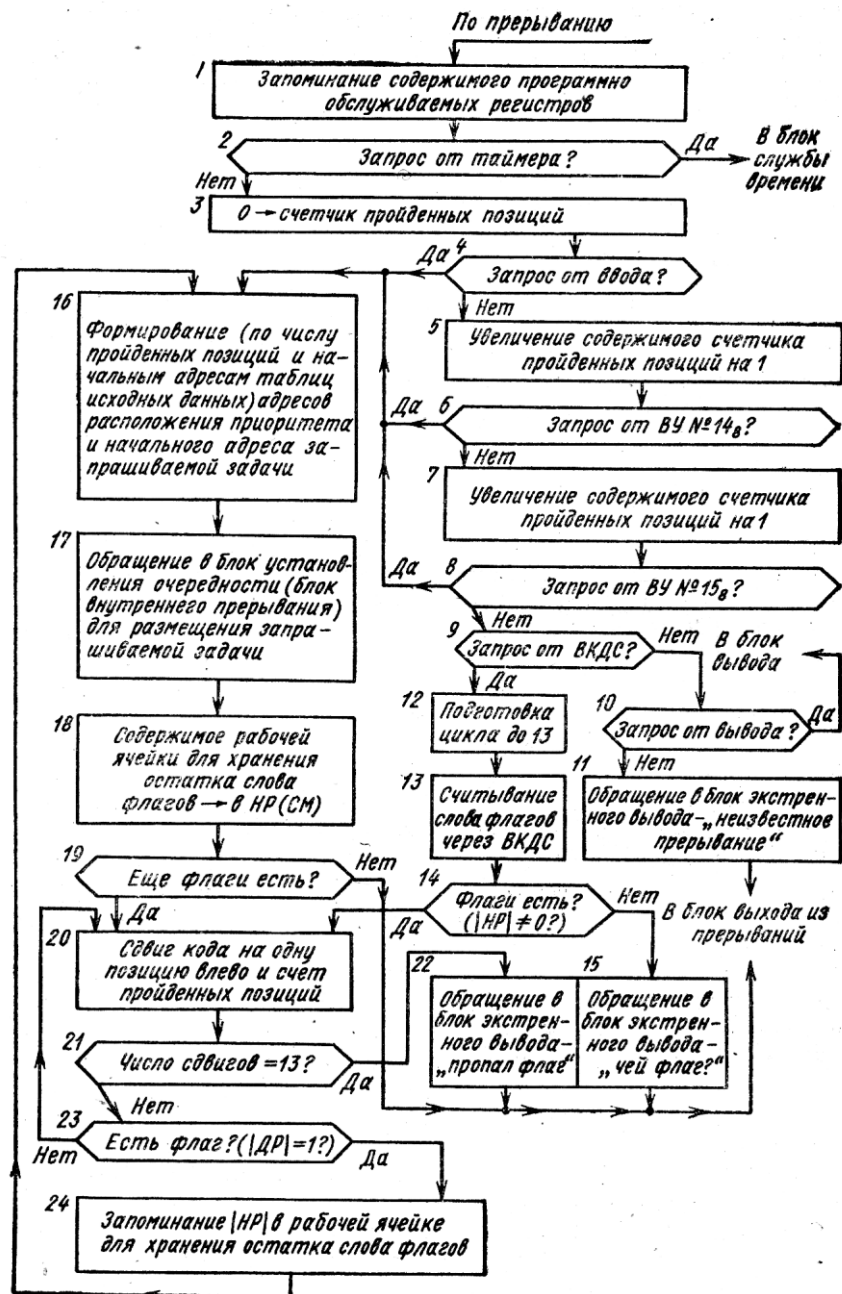


Рис. 4.9. Структурная схема блока обнаружения источника прерывания и идентификации запрашиваемой задачи.

Если запрос обладает низким приоритетом, он попадает в очередь не начавшихся задач; содержимое ячеек 0000, «а» и «в» при этом не затрагивается и тут же используется блок выхода из прерывания для продолжения прерванной задачи. Если же запрос обладает высоким приоритетом и должно начаться его немедленное обслуживание, в буферную зону очереди прерванных задач вместе с другими сведениями попадает (переписывается) и содержимое ячеек 0000, «а» и «в». Чтобы не усложнять логику построения блока выхода из прерывания, в ячейку 0000 тут же заносится начальный адрес запроса, а ячейки «а» и «в» обнуляются.

Дополнительные логические блоки появляются у диспетчера для восстановления содержимого ячеек 0000, «а» и «в» по мере решения очередных задач. Эти вопросы, более специфичные для диспетчера, рассматриваются в гл. 8.

Следует отметить, что команда 5400 должна обязательно располагаться непосредственно вслед за командой разрешения прерывания. Это делается на тот случай, если за время обслуживания очередного прерывания успело накопиться ещё хотя бы одно требование на прерывание (оно запасено на каком-либо флаге, но не может быть реализовано до выполнения команды 6001 и ещё одной, следующей за ней). В этом случае только пара команд 6001, 5400 обеспечивает выход из диспетчерской части программы до наступления нового прерывания. По команде 5400 в счетчике команд (СК) восстанавливается адрес продолжения прерванной ранее задачи и тут же происходит новое прерывание (в прерванной задаче не успевают выполниться ни одна команда). В ячейку 0000 попадает код того же адреса, и все необходимые предпосылки продолжения задачи после обслуживания нового прерывания сохраняются. Если же между командами 6001 и 5400 поместить хотя бы одну команду, блок выхода из прерывания в рассматриваемом случае не будет работать, так как до выполнения команды 65400 наступит новое прерывание, содержимое ячейки 0000 заменится новым адресом возврата (причем это будет внутренний адрес диспетчерской части программы) и, таким образом, путь возвращения к прерванной задаче разрушится.

Логика обслуживания прерывания (как и всего диспетчера) заметно усложняется, если глубина прерывания должна быть более единицы. В этом случае допускается многократное прерывание одних задач другими в любом месте их рабочих программ. Поэтому все задачи идут с разрешенным прерыванием, а у блока обслуживания прерывания появляется новая сторона взаимодействия с диспетчером. Так как все прерывания автоматически начинаются с занесения адреса прерывания в ячейку 0000 и требуют немедленного запоминания содержимого НР(СМ) и ДР(ДРС), до разрешения нового прерывания необходимо позаботиться о полном использовании или о сохранении данных, временно запомненных в ячейках 0000, «а» и «в». Диспетчер решает эту задачу с использованием системы приоритетов и двух очередей (одной — на решение не начавшихся задач, другой — очереди прерванных задач), имеющих свои буферные зоны в МОЗУ для хранения необходимых данных.

Таким образом, программное прерывание создает необходимую основу для построения на базе «Электроники-100» системы, способной одновременно обслуживать несколько задач с использованием различных внешних устройств при времени реакции на очередное прерывание, исчисляемом десятками микросекунд. В «Электронике-100» предусмотрен ещё один канал обмена информацией, быстродействие которого существенно выше и определяется только быстродействием машины. Это канал информационного прерывания, обеспечивающий в режиме «разрыва данными» максимальную скорость обмена в один машинный цикл (2 мкс для «Электроники-100» в обычном исполнении и 1,6 мкс при интегральном исполнении).

При информационном прерывании машина может осуществлять обмен с внешним устройством одиночными словами (одноцикловый разрыв) или массивом слов (трехцикловый разрыв). Особенностью такого режима работы машины является почти полное его осуществление чисто аппаратными (схемными) средствами. Внешнее устройство при этом подает в машину семь видов сигналов:

- запрос на разрыв;
- сигнал выбора цикла (одно- или трехцикловый разрыв, т. е. обмен отдельным словом или массивом);
- запрет на увеличение текущего адреса при обмене (при одноцикловом обмене);
- (12-разрядный) адрес внешнего устройства (т. е. адрес в МОЗУ, начиная с которого осуществляется обмен с внешним устройством) и (3-разрядный) номер куба;
- сигнал «данные в ЭВМ» (определяющий направление обмена: запись или чтение);
- (12-разрядные) данные ВУ — передаваемая информация;
- сигнал «входные данные» (для записи информации от внешнего устройства в буферный регистр).

При появлении запроса на «разрыв данными» машина заканчивает выполнение очередной команды, после чего работа программы приостанавливается и происходит необходимый обмен данными с внешним устройством в соответствии с его сигналами. По окончании обмена продолжается выполнение программы.

При одноцикловом разрыве единственный код «данные ВУ» записывается (или считывается) по адресу, указанному 12-разрядным сигналом от внешнего устройства.

При трехцикловом разрыве адрес внешнего устройства, указываемый этим сигналом, определяет начальный адрес поля обмена, однако первая ячейка этого поля отводится для указания адреса счетчика «отсчет слов» (ОС). Собственно массив для обмена начинается со второй ячейки поля и далее, ко все большим адресам. Объем массива определяется обратным кодом счетчика ОС, так что возможно программным путем устанавливать счетчик ОС на необходимую длину массива. Обмен начинается по сигналу внешнего устройства и заканчивается при переполнении счетчика ОС — соответствующий импульс выдается из машины для сброса всех семи сигналов обмена.

Информационное прерывание как очень быстрый канал обмена обычно используется для ввода (или вывода) данных от малоинерционных объектов, для обмена данными между ЭВМ, а также между машиной и буферными накопителями. В частности, с использованием канала «разрыв данными» организован обмен кодами между «Электроникой-100» и накопителем на магнитной ленте (типа НМЛ-67).

Важно отметить, что информационное прерывание программа никак не ощущает, поэтому окончание обмена, если на него необходимо реагировать по программе, должно сопровождаться обычным программным прерыванием. Вместе с этим при использовании информационного прерывания затраты времени на реакцию по программе происходят уже после обмена, благодаря чему оперативность системы в целом заметно возрастает.

4.7. ОРГАНИЗАЦИЯ АРИФМЕТИЧЕСКИХ ВЫЧИСЛЕНИЙ

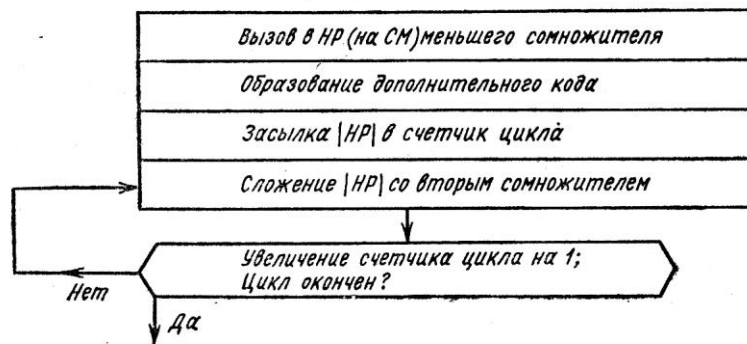
Специфика малой машины отчетливо проявляется при решении задач вычислительного характера. Наряду с общими проблемами выбора численного метода и разработки алгоритма здесь возникают дополнительные вопросы, связанные с масштабированием переменных, выбором длины машинного слова и (даже) с реализацией необходимых вычислительных операций. В системе команд «Электроники-100» имеется лишь одна чисто арифметическая операция — сложение, все остальные операции приходится организовывать программным путем. Основой для этого служат другие команды (большой частью СЛ, УП и МК) и принятая система представления отрицательных чисел в дополнительном коде.

Наиболее просто реализуется операция *вычитания*:

Вызов вычитаемого в НР (на СМ)
Образование дополнительного кода
Сложение содержимого НР (СМ) с уменьшаемым

Вся операция требует использования двух команд СЛ и одной или двух (если в начале $|\text{НР}| \neq 0$) микрокоманд. Результат вычитания, оставшийся в НР (на СМ), может быть заслан по необходимому адресу с использованием команды ПС.

Общий принцип реализации операции *умножения* состоит в последовательном суммировании одного из сомножителей необходимое число раз (в соответствии с величиной второго сомножителя). Дополнительными особенностями при этом являются возможность превышения (у результата) разрядной сетки машины и учёт способа представления отрицательных чисел в машине. Только в простейшем случае положительных сомножителей, когда заранее известно, что результат укладывается в разрядную сетку машины и один из сомножителей достаточно мал, целесообразно организовать цикл прямого сложения второго сомножителя:



В более общем случае используют специальные *подпрограммы умножения*, входящие в общее математическое обеспечение и поставляемые вместе с ЭВМ. В этих подпрограммах результат имеет удвоенную длину слова: старшие разряды по выходе из подпрограммы располагаются в НР (СМ), младшие находятся в стандартной ячейке. Два старших разряда занимают под знак и имеют код 00, если результат положителен, и 11, если он отрицателен (результат при этом представляется в дополнительном коде в остальных 22_{10} разрядах)¹. Само умножение ведется после приведения сомножителей к виду положительных чисел и со сдвигом сомножителей. При обращении в подпрограмму один из сомножителей вызывается в НР (на СМ), а другой занимает ячейку вслед за командой обращения к подпрограмме.

Операция *деления* реализуется путем последовательного вычитания делителя из делимого. Стандартная подпрограмма рассчитана на случай, когда делимое записано в двух ячейках, а делитель в одной ячейке.

Разработаны и также входят в состав математического обеспечения машины подпрограммы для вычислительных операций над числами, занимающими по две ячейки каждое.

Следует указать, что благодаря большому быстродействию машины подпрограммы арифметических действий укладываются в сотни микросекунд². Более того, по отдельному заказу «Электроника-100» комплектуется добавляемым арифметическим блоком (ДАБ), аппаратным путем реализующим операции умножения и деления за 35 мкс (для «Электроники-100И» — за 10—11 мкс). Команды обращения к ДАБ занимают часть микрокоманд второй группы и в отличие от них имеют единицу в младшем разряде (см. табл. 3.5).

Важным этапом решения задач вычислительного характера на «Электронике-100» является масштабирование физических переменных и определение необходимой длины машинного слова.

Исходными предпосылками здесь служат следующие соображения. В физических задачах константы, коэффициенты и переменные могут быть (а чаще всего и бывают) размерными величинами (длина измеряется в метрах, масса — в килограммах и т. п.), машина же при вычислениях оперирует с безразмерными кодами. Физические переменные чаще всего могут принимать непрерывный континуум значений в некотором диапазоне их изменения; при машинной реализации на ЦВМ все константы и переменные могут принимать лишь некоторое конечное число «разрешенных» значений. В частности, 12 бит машинного слова в «Электронике-100» позволяют иметь $2^{12} - 1 = 4095$ ступеней (или уровней) разрешения, причем при желании можно принять, что вся эта «лестница» нарастает от 0 до 4095 либо что половина ступеней спускается ниже нуля (т. е. вся шкала охватывает диапазон от -2048 до $+2047$)³. Физические величины могут принимать в некоторых задачах как очень большие, так и очень малые значения, причем эти значения чаще бывают дробными, чем целыми. В «Электронике-100» основная система команд и способ представления чисел рассчитаны на использование чисел с фиксированной запятой.

В первую очередь выбирают безразмерные целочисленные переменные (типа счетчиков, признаков состояния и т. п.). Для этих переменных единица младшего разряда (т. е. машинная единица при нормировании ступени разрешения) отождествляется с единицей физической переменной. После этого вопрос состоит только в том, достаточна ли длина машинного слова для перекрытия диапазона изменения переменной: если достаточна, то на переменную распространяются все обычные правила хранения чисел и операций над ними, принятые в «Электронике-100», если нет, то для хранения переменной отводят две или более ячеек и используют в дальнейшем специальные подпрограммы для вычислений, в которых эта переменная участвует.

Для переменных, которые могут принимать как целые, так и дробные значения, производят масштабирование и оценку разрешающей способности машинного представления переменной, исходя из очевидной пропорции:

если всегда $x \geq 0$, то

$x_{\max}$ — 4095 ступеней,

Δx — 1 ступень;

для знакопеременных величин:

$|x|_{\max}$ — 2047 ступеней,

Δx — 1 ступень.

Из этих соотношений непосредственно следует, что разрешающая способность представления переменной в одной ячейке составляет для положительных величин

¹ Дублирование содержимого знакового разряда (с автоматическим заполнением, таким образом, одного старшего разряда, всегда пустующего при перемножении двух чисел с мантиссами по 11 разрядов) обеспечивает совершенно такой же принцип использования разрядной сетки для результата (теперь уже в пределах 24 разрядов), что и для исходных 12-разрядных сомножителей

² Среднее время умножения и деления чисел с обычной точностью составляет 460 и 690 мкс соответственно.

³ В некоторых случаях нормируют не величину ступени, а диапазон изменения переменной: $-1 \leq x \leq 1 - 2^{-11}$; в этом случае величина ступени принимает значение 2^{-11} .

$$\Delta x_{\min} = \frac{|x|_{\max}}{4095} \text{ (физ. ед./ступень)}$$

и для знакопеременных величин

$$\Delta x_{\min} = \frac{|x|_{\max}}{2047} \text{ (физ. ед./ступень)}$$

Это означает, что при исходном масштабировании рассматриваемых однородных величин и в дальнейших расчетах они не могут быть представлены точнее, чем до $\Delta x_{\min}$, а те из них, модуль которых окажется меньше $\Delta x_{\min}$, будут заменены нулями. Если рассчитанная разрешающая способность оказывается приемлемой, то вычисляется масштаб представления переменной в машине:

$$M_x = \frac{1}{\Delta x_{\min}} \text{ (ступеней/физ. ед.)},$$

после чего все рассматриваемые переменные перед вводом в машину умножаются на этот масштабирующий множитель. Для каждой переменной результат округляется до ближайшего целого числа, переводится в двоичный (восьмеричный) код и может быть занесен в одну ячейку, отведенную для соответствующей переменной. Все дальнейшие вычислительные операции осуществляются над полученным кодом по общим правилам действия над переменными в «Электронике-100», а конечный результат может быть приведен к исходному масштабу и размерности делением на M_x .

Если разрешающая способность представления переменной в одной ячейке оказывается недостаточной, прибегают к использованию двух или более ячеек для хранения переменной. При этом число ступеней возрастает примерно в $(2^4)^{m-1}$ раз, где m — число ячеек для хранения одного числа; во столько же раз улучшается разрешающая способность представления переменной. Нетрудно заметить, что эта зависимость весьма резкая — каждая последующая ячейка улучшает разрешение более чем на 3,5 десятичных порядка, так что в инженерных расчетах для размещения переменной, редко требуется более трех ячеек. В таком случае вычислительные операции осуществляются с помощью специальных подпрограмм. Фактическая разрешающая способность определяется как

$$\Delta x_{\min} = \frac{|x|_{\max}}{2^{12m-1} - 1} \text{ (физ. ед./ступень)}.$$

Помимо необходимой разрешающей способности, требуемая длина машинного слова для хранения переменных определяется ещё числом и характером вычислений, в которых эти переменные участвуют. Нетрудно убедиться, что при сложении N положительных двоичных чисел, у которых одинаковое число значащих разрядов (или при умножении некоторого числа на N), величина разрядной сетки результата увеличивается на $\log_2 N$ разрядов. Наряду с этим при вычитании двух чисел требуемая величина разрядной сетки может и уменьшиться. Поэтому целесообразно проследить за всеми операциями, в которых участвует переменная, и оценить возможное увеличение её разрядности. В труднообозримых случаях можно воспользоваться соотношением, полученным в одной из фундаментальных работ [4]:

$$p \geq \frac{1}{2} \log_a \frac{Q}{3} \quad (4.1)$$

Здесь p — число дополнительных разрядов для сохранения с вероятностью 0,95 точности в единицу младшего разряда; Q — число операций; a — основание системы счисления.

При исходном масштабировании переменной необходимо сократить разрядную сетку на величину p , соответствующую ожидаемому ухудшению точности¹. После этого уточняют результирующую разрешающую способность машинного представления переменной и её масштаб и производят окончательный выбор числа ячеек для хранения переменной.

В особо ответственных случаях при вычислениях может быть введен программный контроль переполнения разрядной сетки. Логический анализ в «Электронике-100» относительно просто строится с использованием микрокоманд второй группы, организующих пропуск очередной команды в зависимости от состояния ДР(ДРС) и 0-го разряда НР(СМ). С его помощью могут выявиться такие эффекты, связанные с переполнением сетки и способом представления отрицательных чисел в машине, как превращение суммы положительных чисел в отрицательное число и наоборот при суммировании знакопеременных чисел; превышение одним из заведомо положительных слагаемых результата (суммы) и т. п. В простейшем случае такой анализ в результате обнаружения эффектов переполнения может заканчиваться остановом. С помощью специальных программ, использующих приемы такого рода, и широких возможностей микрокоманд второй группы по проверке сразу нескольких логических условий можно организовать динамическое распределение памяти для хранения переменных и результатов действия над ними (здесь эти вопросы не рассматриваются).

Необходимо отметить ещё одну особенность исходного масштабирования переменных и коэффициентов. Если какие-либо величины по условиям задачи связаны некоторыми соотношениями, то число независимо масштабируемых переменных сокращается на число таких соотношений. Масштаб остальных переменных (и разрешающая способность их представления) определяется в соответствии с выбранными масштабами независимых переменных и с учётом уравнений связи.

Пример.

В простейшей системе регулирования расчет управляющего воздействия выполняется в соответствии с соотношением

¹ Здесь недостаточно просто отвести для результата соответствующее число ячеек, так как возможное переполнение разрядной сетки требует заранее заготовленной программной реакции, т. е. использования специальных подпрограмм.

$$\Delta y = K \Delta x, \quad (4.2)$$

где Δy — величина управляющего воздействия на приводной двигатель (время его включения, с); Δx — величина рассогласования регулируемой величины относительно заданного значения (физ. ед.); K — коэффициент пропорциональности (с/физ. ед.).

Масштабирование Δy и Δx произведено заранее, исходя из диапазона изменения переменных и требуемой точности их представления. При этом масштаб представления K в машине, очевидно, должен определяться с учётом уравнения связи, которое в соответствии с изменением масштаба теперь записывается в виде

$$\Delta y_{\text{маш}} = K_{\text{маш}} \Delta x_{\text{маш}} \quad (4.3)$$

или

$$\Delta y M_y = K M_x \Delta x M_x.$$

Из соотношений (4.2) и (4.3) следует, что

$$M_K = M_y M_x^{-1} (\text{физ. ед./с}),$$

а разрешающая способность машинного представления коэффициента K

$$\Delta K_{\min} = \frac{1}{M_K} = \frac{M_x}{M_y} \quad (\text{с/физ. ед.})$$

Если полученная разрешающая способность $\Delta K_{\min}$ оказывается недостаточной, приходится корректировать масштабы M_x или M_y . В ином случае рассчитывается величина машинной константы.

$$K_{\text{маш}} = K M_K$$

с округлением до ближайшего целого числа¹ и переводом полученного результата в двоичную систему для занесения константы в память.

Приведенные приемы масштабирования достаточно просты и формализованы, чтобы их можно было автоматизировать. Однако чаще всего масштабирование переменных производится вручную при подготовке задачи к постановке её решения на ЭВМ. Положительной стороной при этом является дополнительное, достаточно глубокое осмысливание условий задачи, применяемых соотношений и ограничений, часть которых ускользает от внимания при использовании автоматизированных алгоритмов и приемов

Для облегчения арифметических вычислений в «Электронике-100» может использоваться *программный блок плавающей запятой* (БПЗ), входящий в состав математического обеспечения ЭВМ. С помощью специально разработанных подпрограмм оказывается возможным организовать вычислительные операции и вспомогательные сервисные действия над числами, представленными в нормализованном виде. Для записи (хранения) числа используются три ячейки: в первой хранится двоичный код порядка числа, две следующие ячейки отводятся под нормализованную мантиссу; старшие разряды в первой и во второй ячейках отводятся соответственно под знак порядка и знак числа:



Запись порядка практически ничем не отличается от записи обычного числа в «Электронике-100» — в одиннадцати младших разрядах может быть записан код любого целого числа от 0 до 2047 (от 0000₈ до 3777₈); отрицательные числа записываются в дополнительном коде. В сетке из 24 двоичных разрядов для мантиссы 23 разряда используются для хранения собственно разрядов мантиссы; отрицательные числа записываются в дополнительном коде, охватывающем все 24 разряда. Мантисса всегда нормализована, т. е. старший разряд её отличен от нуля (для положительных чисел в разряде 1 ячейки $a+1$ всегда находится 1). За адрес числа принимается адрес размещения порядка, т. е. a .

При такой форме записи числа мантисса хранит его значащие разряды (в виде 23-разрядного двоичного кода), а порядок указывает, на сколько двоичных разрядов необходимо сдвинуть (вправо, если порядок положителен, и влево, если он отрицателен) запятую, в исходном состоянии помещаемую перед старшим разрядом мантиссы. Если обозначить мантиссу некоторого числа A через M , а порядок через $P/7$, то само число описывается соотношением:

$$A = (\text{sign } A) M 2^{P/7}.$$

Соответствующие правила действий над числами, представленными в такой форме, реализуются в подпрограммах БПЗ. Для описания необходимого набора команд в БПЗ используется внутренняя интерпретирующая система — после обращения в БПЗ все команды получают дополнительный смысл, отличный от их обычного значения в основной системе команд «Электроники-100». Смысл интерпретации кодов операции в БПЗ понятен из табл. 4.1.

Для работы БПЗ² используют ячейки 0005, 0007, 0040 — 0064 и 4700—7611, причем три ячейки на 0-й странице (0044 — 0046) применяются для организации дополнительного программного регистра — так называемого плавающего накопительного регистра (ПНР). С его помощью могут производиться вычислительные и сервисные операции, приведенные в табл. 4.1.

Нетрудно видеть, что БПЗ существенно облегчает организацию вычислений. Однако программист и в этом случае должен помнить, что разрешающая способность машинного представления переменных ограничена 23 двоичными разрядами (которые с учётом порядка могут занимать произвольное положение на логарифмической оси от 2^{-2047} до 2^{2047}). Результаты вычислений нормализуются и округляются, так что мантисса результата занимает также 23 двоичных разряда. При сложении двух чисел,

¹ Очевидно, используемая в машине константа при этом может отличаться от желаемой, однако не более чем на $0,5 \Delta K_{\min}$. Изменять константу K , играющую роль коэффициента усиления в цепи обратной связи системы управления, при машинной реализации можно лишь на величину, кратную $\Delta K_{\min}$, путем изменения $K_{\text{маш}}$ на соответствующее число машинных единиц.

² Один из наиболее полных вариантов — 3-й.

порядки которых различаются более чем на 23 двоичных разряда, меньшее (по модулю) число просто теряется, и БПЗ здесь бессильен чем-либо помочь. Поэтому и при использовании БПЗ оценка разрешающей способности машинного представления переменных является обязательной.

Таблица 4.1

(Программный) блок плавающей запятой	
Диапазон представления от $-0,999999 \cdot 10^{-615}$ до $0,999999 \cdot 10^{615}$	
Команды обращения к памяти (структура аналогична изображенной на рис. 3.1):	
1у—сложение ПНР с у , 2у—вычитание у из ПНР , 3у—умножение ПНР на у , 4у—деление ПНР на у , 5у—запись у в ПНР, 6у—пересылка ПНР по адресу у, 7—нормализация ПНР , Код 0000—команда выхода из БПЗ.	} команды, имеющиеся в любом варианте БПЗ
Дополнительные команды (полный восьмеричный код):	
0001—возведение ПНР в квадрат, 0002—извлечение квадратного корня из ПНР , 0013—обращение к программе ввода, 0014—обращение к программе вывода в стандартном формате	} только в 3-м варианте
0003 — вычисление $\sin x$,	
0004—вычисление $\cos x$,	
0003 — вычисление $\sin x$,	
0004—вычисление $\cos x$,	
0007—вычисление $\ln x$.	
П р и м е ч а н и я : 1. Состав дополнительных команд можно расширить, присвоив новой команде код $00pn$ и записав начальный адрес соответствующей подпрограммы в ячейке $654+00pn$; p, n — восьмеричные числа. 2. В новой подпрограмме нельзя использовать дополнительные команды. 3. Указанная интерпретация команд выполняется по соответствующим подпрограммам — от момента обращения в БПЗ по команде 4407 до выхода из БПЗ по коду 0000. 4. Время выполнения операций возрастает (по сравнению с командами я подпрограммами работы с числами обычной точности) примерно на два десятичных порядка.	

4.8 РАБОТА С ТАБЛИЦАМИ

Во многих задачах оказывается необходимой или рациональной организация некоторых массивов данных, коэффициентов или констант в виде таблиц. Независимо от вида хранимой в них информации каждая таблица обычно занимает в МОЗУ определенный ряд последовательно расположенных ячеек, так что любой массив табличных данных трансформируется к одномерному представлению.

Когда исходный массив многомерен, заполнение таблицы производится в соответствии с некоторым правилом (расчётным соотношением), связывающим индексы переменной в массиве с её адресом в таблице¹. При заданном правиле формирования таблицы она полностью характеризуется своим начальным адресом $A_{\text{нач}}$ и длиной L (числом хранимых машинных слов).

Наиболее распространенными операциями при работе с таблицами являются выборка (занесение) данных из (в) таблицы и сдвиг (перемещение) таблиц. Для выборки единственного слова из таблицы достаточно определить его фактический адрес и затем косвенно обратиться по этому адресу. Если нумерация позиций в таблице начинается с 1 (1, 2, ..., L), то фактический адрес i -й позиции A_i определяется соотношением

$$A_i = A_{\text{нач}} + i - 1. \quad ^2$$

Гораздо чаще выборка данных из таблицы осуществляется не однократно, а входит в некоторый расчётный цикл, внутри которого требуются те или иные табличные величины. В этом случае процесс выборки из таблицы, очевидно, должен быть подчинен общей логике построения цикла и учитывать его возможности и закономерности.

При программировании на «Электронике-100» выборка из таблиц получается наиболее простой в том случае, когда шаг выборки (увеличение адреса позиции при очередном обращении к таблице) является равномерным и равен 1. Тогда табличная переменная ничем не отличается от переменных цикла (соответствующие правила обращения с ними рассмотрены в § 4.4).

¹ Практически это правило формализует обычно такой способ хранения информации, при котором в начале таблицы располагают первую строку многомерного массива (т. е. фиксируют все индексы, кроме первого), затем вторую (увеличивают на 1 второй индекс и вновь дают первому индексу пробежать все дозволённые значения) и т. д., пока не будут перебраны все значения второго символа, а затем, при необходимости, третьего и т. д.

² Необходимость вычитания 1, очевидно, связана с тем, что нумерация позиций начинается не с 0, а с 1, т. е. по адресу $A_{\text{нач}}$ уже хранится первая позиция, определяемая фактическим адресом A_1 . При ручном программировании подобные тонкости встречаются постоянно, и они легко поддаются учёту, если размерность массива мысленно снизить, например, до нескольких единиц.

Если шаг равномерной выборки равен $\Delta A \neq 1$, то текущий адрес может формироваться внутри цикла, однако в отличие от предыдущего случая переадресация осуществляется уже не одной командой УП, а тремя и с использованием НР(СМ):

Вызов текущего адреса в НР (на СМ)
Сложение с ΔA
Засылка результата в рабочую ячейку текущего адреса

Таким способом может производиться выборка не только от начала таблицы к её концу, но и в обратном порядке (о уменьшением адресов) — достаточно иметь $\Delta A < 0$.

Когда шаг выборки из таблицы является неравномерным, целесообразно использование вспомогательной таблицы, куда заранее (или при работе предыдущих участков программы) заносятся фактические адреса табличных Данных, требуемых на каждом шаге расчета. При этом выборка осуществляется с использованием простейшего переадресования вспомогательной таблицы и двойного косвенного адресования — в одной рабочей ячейке цикла

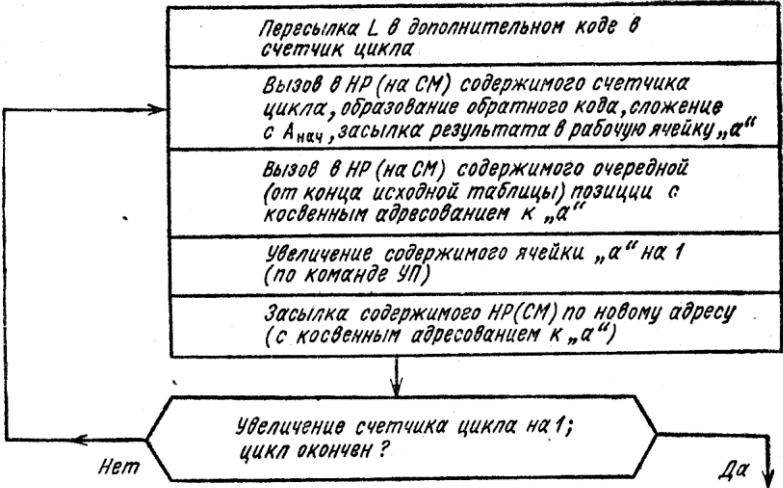


Рис. 4.10. Структурная схема сдвига таблицы на одну позицию назад (к большим адресам); L — длина таблицы; $A_{нач}$ — её начальный адрес.

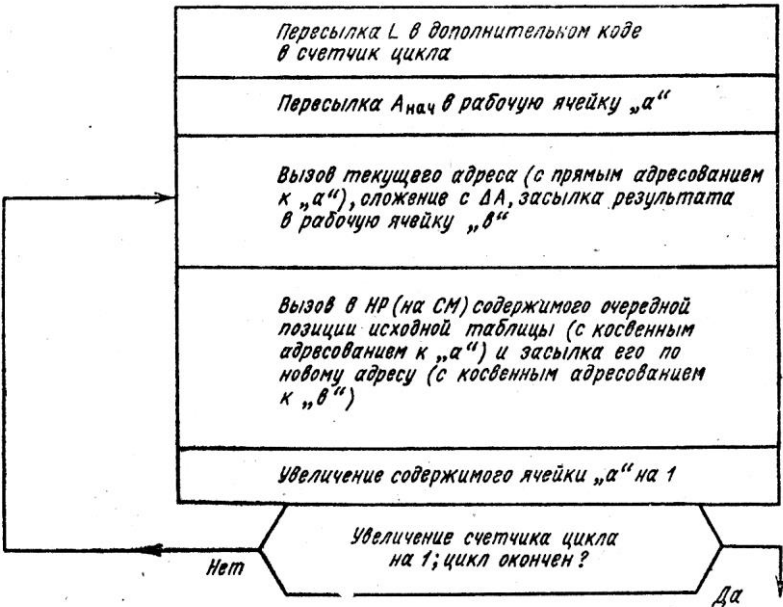


Рис. 4.11. Структурная схема сдвига таблицы вперед (к меньшим адресам); $\Delta A < 0$ — величина сдвига; L — длина таблицы; $A_{нач}$ — её начальный адрес, хранится текущий адрес вспомогательной таблицы и при (первом) косвенном адресовании к этой ячейке вызывается фактически необходимый текущий адрес табличных данных, который пересылается в другую рабочую ячейку цикла: при косвенном адресовании к этой последней ячейке (второе косвенное адресование) выбирается требуемая табличная величина.

В операциях сдвига таблиц внутри цикла необходимо формировать два текущих адреса табличных данных (откуда исходная величина берется и куда она засылается). Нетрудно заметить, что в общем случае, когда новая и старая таблицы могут перекрываться, при сдвиге таблицы вперед (к меньшим адресам) сдвиг должен начинаться с её первой ячейки, а при сдвиге назад (к большим адресам) — с последней; в обоих случаях цикл должен иметь L повторений (L — длина таблицы). С учётом

особенностей программного формирования (подготовки) цикла в «Электронике-100» рационально один текущий адрес подготавливать при организации цикла, а второй формировать внутри цикла, на основе жесткой связи текущих адресов при сдвиге таблицы. Структурные схемы программ сдвига таблиц приведены на рис. 4.10, 4.11.

Важной особенностью схемы сдвига таблицы на одну позицию назад является дополнительное использование содержимого счетчика цикла в операции формирования текущих адресов — при очередном увеличении счетчика цикла на 1 происходит подготовка автоматического переадресования одного из текущих адресов. Другой особенностью этой схемы является использование одной и той же рабочей ячейки для формирования обоих текущих адресов при экономном обращении к НР (СМ) — переадресование (увеличение текущего адреса на 1 по команде УП) производится после вызова (с косвенным адресованием) данных из очередной позиции таблицы в НР (на СМ) с немедленной их пересылкой по новому адресу (с косвенным адресованием к той же рабочей ячейке). Особенность адресования позиций исходной таблицы с 1, а не с 0 учтена использованием операции образования обратного кода, отличающегося от дополнительного как раз на 1.

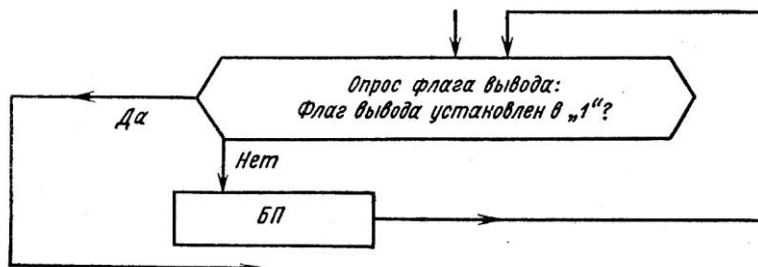
4.9. ПРИНЦИПЫ ОРГАНИЗАЦИИ ПЕЧАТИ

Большинство задач, решаемых на ЭВМ, заканчиваются выдачей результатов на печать. В машинах, комплектуемых электрифицированными пишущими машинками (ЭПМ), вывод информации может осуществляться только посимвольно, причем аппаратные возможности минимального комплекта оборудования этим и ограничены. Понятно, что в таких условиях программное обеспечение процесса вывода всей требуемой информации является важной частью общего математического обеспечения.

В системе команд «Электронике-100» вывод очередного кода из НР(СМ) в буфер вывода с последующей печатью¹ символа и сигнализацией (с помощью флага вывода) о готовности к выводу очередного кода обеспечивает команда 6046. Этой команде, очевидно, должен предшествовать вызов кода в НР(на СМ). До отработки собственно команды вывода электромеханическим устройством пишущей машинки команду вывода другого символа подавать нельзя. Таким образом, элементарная функция вывода отдельного символа реализуется по схеме:

Вызов кода очередного символа в НР (на СМ)
6046
Ожидание флага вывода

Последняя операция может осуществляться путем заикливания на команде опроса флага вывода (при работе в так называемом «монопольном режиме»):



либо с использованием системы прерывания (см. § 4.6) — при работе с диспетчером.

Любые другие (более сложные) функции вывода требуют дополнительных программных затрат, которые сводятся, как правило, к преобразованию исходной информации к виду, обеспечивающему посимвольный вызов кодов в НР (на СМ), и заикливанию операций вывода отдельного кода. В простых программах эти процессы обычно разделяются, хотя и являются тесно связанными через формат представления информации, подготовленной для вывода.

Наиболее просто собственно вывод организуется в том случае, когда набор подпрограмм подготовки данных для вывода обеспечивает получение отредактированной информации (с кодами пропусков, запятых, перевода каретки и т. п.) в таком виде, что все коды выводимых (печатаемых и служебных) символов располагаются подряд в отдельных последовательных ячейках памяти. Для настройки подпрограммы вывода достаточно указать лишь начальный адрес таблицы заготовленных кодов и её длину; сама подпрограмма реализуется в виде цикла рис. 4.12.

Более универсальный блок вывода требуется, когда сообщение располагается в МОЗУ отдельными группами символов и каждая группа представляет собой подготовленную к выводу таблицу кодов символов (по одному коду в каждой последовательной ячейке). Платой за универсальность программы является её усложнение и усложнение самого обращения к подпрограмме.

В основной программе помимо таблиц сформированных кодов символов ещё должны содержаться две вспомогательные таблицы: таблица количества символов в каждой группе и таблица начальных адресов для каждой группы кодов, подготовленных к выводу. Адреса этих вспомогательных таблиц, а также сведения о количестве выводимых групп должны быть указаны при обращении к блоку вывода. В программе вывода (рис. 4.13) организуется двойной цикл: внутренний — для вывода каждой очередной таблицы группы символов /практически совпадающий по схеме с рис. 4.12) и внешний — для подготовки исходных данных (начального адреса и количества символов в очередной группе) для работы внутреннего цикла. Удобно при этом внутренний цикл сохранить в виде самостоятельной подпрограммы.

Нетрудно видеть, что употребление более универсальной подпрограммы вывода (вместо последовательного многократного использования более простой подпрограммы) является рациональным, когда число выводимых групп превышает 10 — 12. Однако и при меньшем числе групп дополнительные абсолютные затраты памяти составляют лишь около 20 ячеек МОЗУ но общая организация программ выигрывает за счет сохранения управления внутри подпрограммы на

¹ Или перфорацией, в зависимости от положения тумблеров на панели блока УВВУ (см. гл. 5).

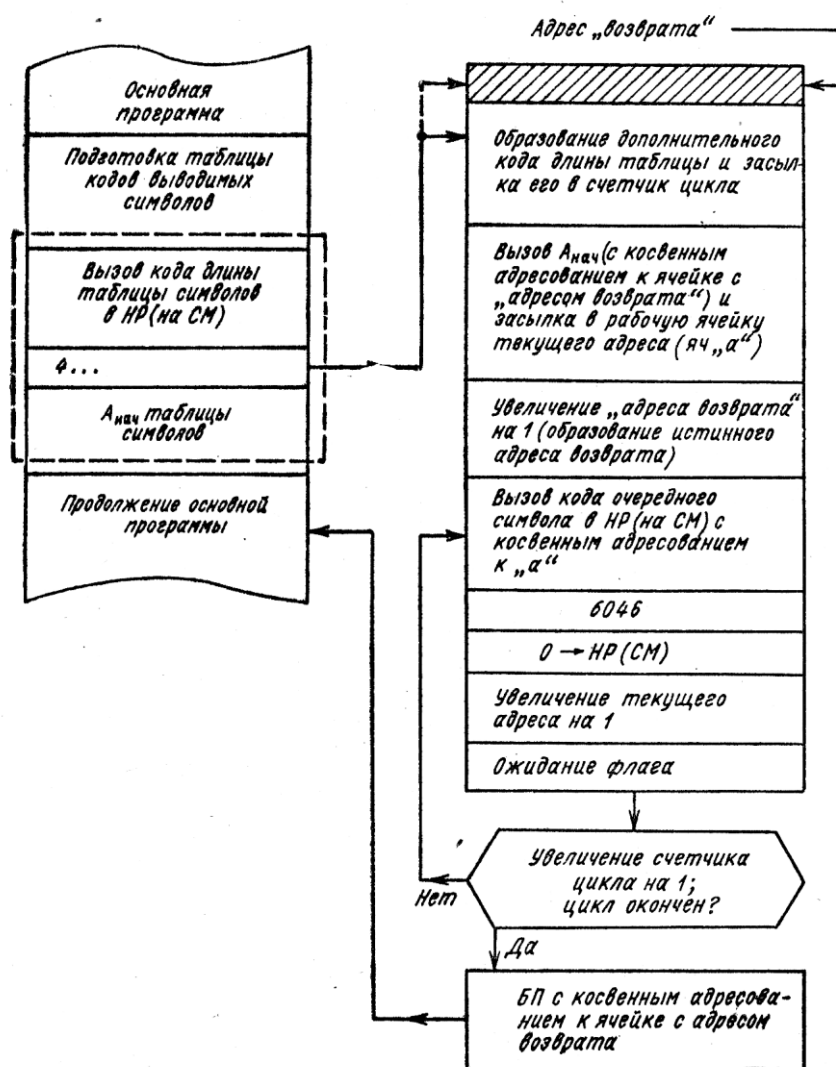


Рис. 4.12. Структурная схема вывода таблицы символов (по одному коду в ячейке); штрихом указан участок основной программы, организующий обращение к подпрограмме вывода.

Дальнейший путь совершенствования подпрограмм вывода предполагает использование приемов уплотнения информации, подготовленной к выводу. Для кодирования информационных и служебных символов клавиатуры ЭПМ при выводе оказывается достаточным 6 двоичных разрядов¹, так что в принципе в каждой ячейке МОЗУ могут храниться одновременно коды двух символов. При этом очевидно, достигается существенная экономия объема памяти, отводимого для хранения постоянной (неизменяемой) части выводимой информации, и одновременно усложняются подпрограммы подготовки переменной части сообщения и собственно подпрограммы вывода.

Дополнительные возможности открывает применение кодов, не используемых для кодирования символов на «Консуле-254», в качестве командных кодов для подпрограммы вывода (коды 15₈ — 17₈, 32₈ — 37₈ и 77₈). В частности, рационально использовать один из кодов в качестве признака конца сообщения, а другой — для организации цикла вывода одного и того же символа². Таким образом, общая последовательность кодов в таблице, подготовленной для вывода, содержит в общем случае не только информационные и служебные выводимые коды, но и команды для подпрограммы вывода; такая последовательность в дальнейшем будет называться *кодограммой*.

Структурная схема подпрограммы вывода кодограммы с кодированным указанием конца кодограммы и внутренних циклов (по выводу повторяющегося несколько раз одного и того же символа) приведена на рис. 4.14.

Развитие программ вывода обеспечивает включение в кодограмму кодированных команд обращения к подпрограммам подготовки информации непосредственно по ходу вывода информации, переход к другим фрагментам кодограммы, редактирование выводимой информации и т. д. При этом отпадает необходимость в выделении специального поля в МОЗУ для сбора всего сообщения целиком в одном месте. Результат преобразования информации выбирается на печать непосредственно из рабочих ячеек подпрограмм, тут же освобождая их для нового обращения. Такая универсальная подпрограмма вывода является весьма сложной и содержит помимо собственно программы вывода все подпрограммы подготовки (преобразования) и редактирования данных и значительную управляющую часть, обеспечивающую интерпретацию команд кодограммы и

¹ С помощью ЭПМ «Консул-254», которой комплектуется «Электрони-ка-100», может быть напечатано более 64 символов, однако переход с регистра на регистр может выполняться как вывод отдельного кода и, таким образом, разнообразие выводимых кодов не превышает числа клавиш ЭПМ (55), укладываемого при кодировании в 6 двоичных разрядов.

² Идеи этого и дальнейшего усовершенствования программ вывода и их первые реализации принадлежат А. С. Жданову.

организацию прохождения отдельных её фрагментов.

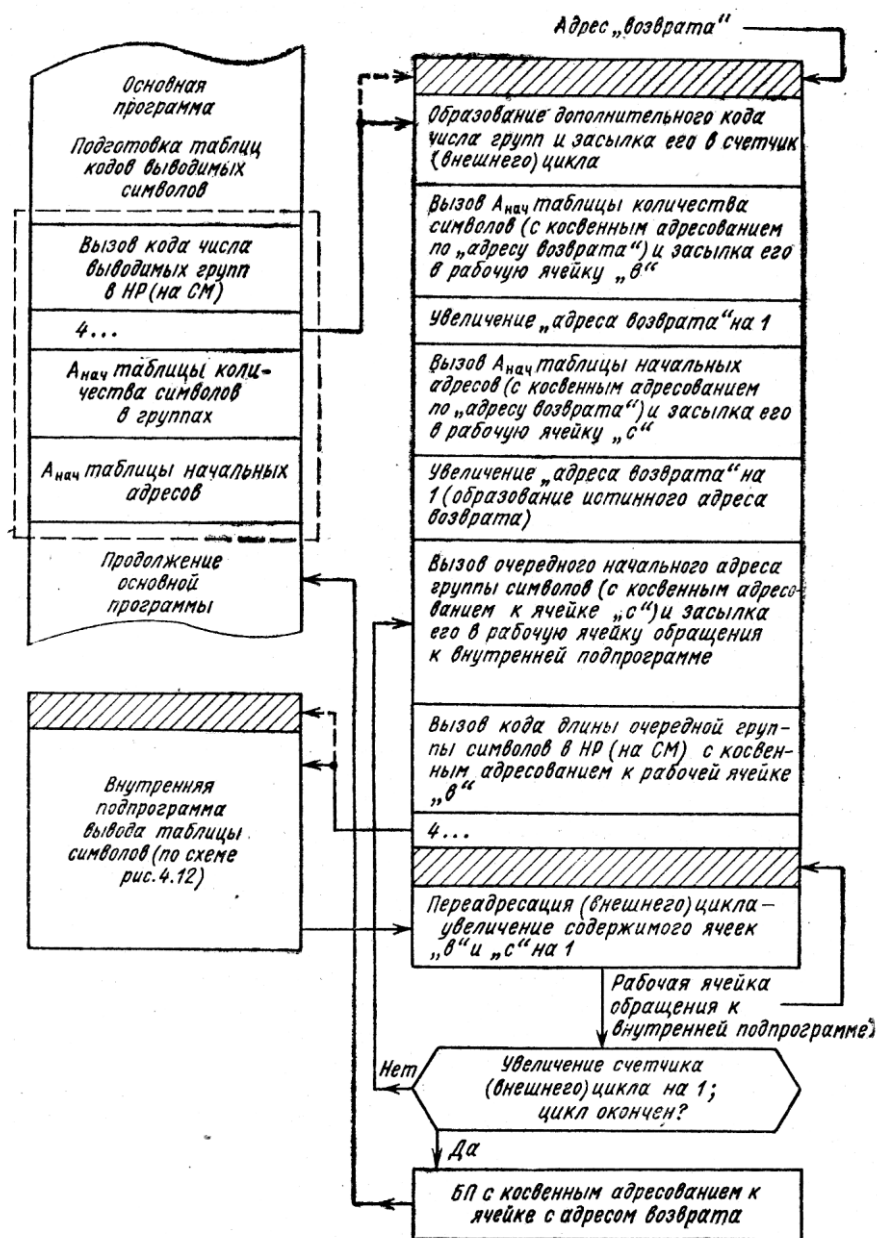


Рис. 4.13. Структурная схема вывода нескольких групп символов из разных участков МОЗУ; штрихом указан участок основной программы, организующий обращение к подпрограмме вывода.

4.10. ОСОБЕННОСТИ РАБОТЫ С РАСШИРЕННЫМ МОЗУ

В тех случаях, когда 4096 машинных слов оперативной памяти оказывается недостаточным для хранения команд программы и всех необходимых данных, эффективным средством увеличения возможностей ЭВМ является подключение дополнительных блоков оперативной памяти (расширение МОЗУ). В «Электронике-100» возможно наращивание оперативной памяти кубами по 4К слов до 8 кубов — до 32 768 машинных слов (конструктивное исполнение «Электроники-100И» обеспечивает возможность размещения в корпусе процессора 2-х кубов памяти по 4К слов каждый и наращивание памяти блоками по 8К слов). Вместе с этим основная система команд (в первую очередь, команды обращения к памяти) рассчитана на работу с одним кубом памяти. Поэтому дополнительные блоки МОЗУ рассматриваются как (дополнительные) внешние устройства со своими специальными командами обращения (см. табл. 3.1). Аппаратурно в машине вместе с дополнительными кубами памяти появляются три дополнительных регистра: регистр поля данных, регистр поля команд и регистр запоминания полей. - Итак, в общем случае машина может содержать до 8 кубов памяти по 4096 слов каждый. Каждый куб памяти рассматривается как некоторое поле, в котором могут располагаться и команды и данные. Поля нумеруются с 0 до 7 (очевидно, при этом десятичная и восьмеричная нумерации полей совпадают); команды обращения к расширенному МОЗУ имеют вид $62mn$, где m и n — восьмеричные числа.

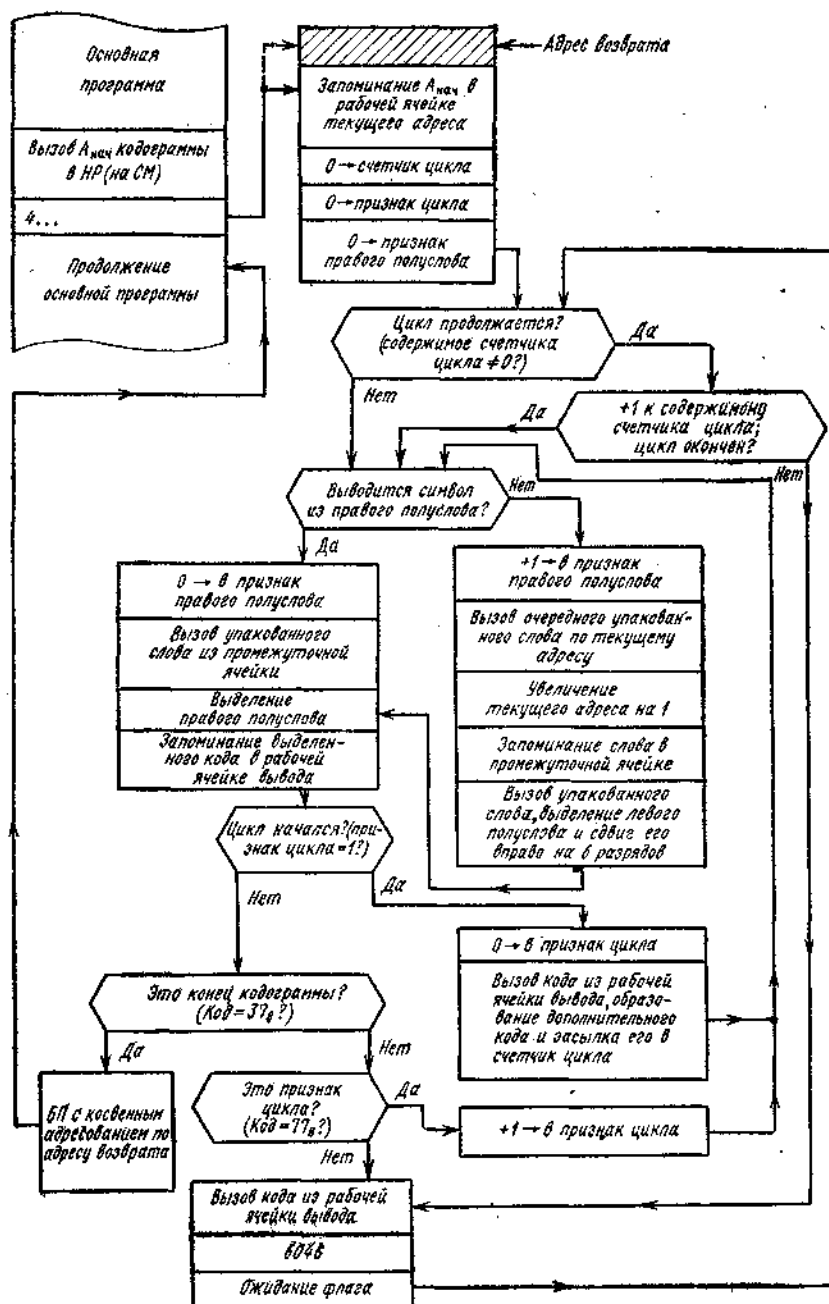


Рис. 4.14. Структурная схема блока вывода упакованной информации с циклами вывода одиночного символа и признаком конца кодограммы,

***Команды установки поля данных и поля команд обеспечивают, таким образом, доступ программы к любым данным в расширенной памяти и использование любого её участка для расположения программы. Вместе с использованием расширенной памяти программисту приходится иметь дело и с двумя фактически расширенными основными регистрами машины: счётчик команд при работе с расширенной памятью расширяется за счёт регистра поля команд, а регистр адреса — за счёт регистра поля данных. Поскольку в расширенной памяти по-прежнему хранятся 12-разрядные машинные слова (команды, константы, числа), не требуют расширения буферный регистр, куда эти слова вызываются, и накопительный регистр (сумматор), где производятся все действия над исходными данными согласно (прежней) основной системе команд.

Когда все команды программы уместятся в одном поле, а исходные данные и результаты расчёта могут располагаться в другом поле (или других полях), возникает необходимость указать, из какого поля берутся (или в какое поле засылаются) данные. Наряду с этим при работе самой программы часть констант и промежуточных результатов обычно хранятся вместе с программой (в том же поле). Чтобы предоставить возможность оперировать обеими группами данных после исполнения команды установки N -го поля данных 62N1, все команды обращения к памяти с прямым адресованием выбирают данные из того же поля, в котором находится программа, а при косвенном адресовании команды ЛУ, СЛ, УП и ПС начинают работать с данными из N -го поля. Расширенный адрес данных, используемых при косвенном адресовании, определяется содержимым 3-разрядного регистра поля данных и тем адресом, который выбирается в регистр адреса при исполнении указанных команд (предварительно этот адрес, следовательно, должен быть запасен или сформирован в одной из ячеек того поля, в котором находится программа; к этой ячейке и происходит косвенное адресование).

Пример. Пусть имеется программа:

	Ячейка	Содержимое	Пояснения		Ячейка	Содержимое	Пояснения
0-й куб	0424	7200	Вызов в НР	1-й куб	0430	0700	Константа
	-5	1232	(на СМ)		1		
			кода из		2		
	6	6211	ячейки 0432				
	7	1631	Сложение НР				
			с содержимым				
			ячейки 1-го				
			куба, адрес				
			которой указан				
			в ячейке 031				
			текущей				
			страницы				
	0430	7402	Останов				
	1	0430	Адрес				
	2	6000	Константа				

Если пустить машину из ячейки 0424 0-го куба МОЗУ, то после останова в ячейке 0430 НР (СМ) будет содержать код 6700.

Так как только относительно простые программы могут обходиться прямым адресованием (действие которого распространяется лишь на текущую страницу и на 0-ю страницу того же поля, в котором находится программа), рекомендуется устанавливать поле данных другого куба МОЗУ только непосредственно перед использованием данных этого куба и тотчас после этого восстанавливать поле данных того куба, в котором находится и программа.

Другую команду обращения к расширенной памяти приходится применять в тех случаях, когда сама программа занимает более куба МОЗУ. Команда установки N -го поля команд 62N2 заносит N в 3-разрядный регистр поля команд и подготавливает переход в этот куб памяти. Однако сам переход в новое поле происходит лишь тогда, когда после исполнения команды 62N2 встречается команда ОП или БП. Расширенный адрес передачи управления определяется содержимым регистра поля команд и тем адресом, который команды ОП или БП выбирают в счётчик команд. От момента выполнения команды 62N2 до окончания выполнения команды ОП или БП прерывание блокируется (если оно ранее было разрешено).

Пример. Имеется программа:

	Ячейка	Содержимое	Пояснения		Ячейка	Содержимое	Пояснения
0-й куб	6250	7200	Вызов кода из ячейки признака	1-й куб	6250	3252	Очистка рабочей ячейки
	1	1256					
	2	6212	Установка 1-го поля команд		1	7402	Останов
					2	////	Рабочая ячейка
	3	7440	Проверка на ноль НР				
	4	7402	Останов				
	5	5657	БП с учётом установленного поля команд				
	6	////	Ячейка признака				
	7	6250	Адрес				

При пуске программы из ячейки 6250 0-го куба МОЗУ останов может произойти в 0-м или 1-м кубе МОЗУ в зависимости от значения признака, занесенного (ранее) в ячейку 6256 0-го куба МОЗУ. Если признак не равен нулю, то останов происходит в ячейке 6254 0-го куба несмотря на то, что командой 6212 подготовлены условия перехода в 1-й куб. Если признак равен нулю, то программа переходит в 1-й куб и останавливается в ячейке 6251, предварительно очистив рабочую ячейку 6252 1-го куба. Тот же результат достигается, если в ячейке 6255 вместо команды 5657 использовать команду 6250.

Пример. Подпрограмму 0-го куба используют только программы 1-го куба.

	Ячейка	Содержимое	Пояснения		Ячейка	Содержимое	Пояснения
0-й куб	0200	////	Адрес возврата	1-й куб	1153	7200	Вызов исходных данных
	1	3222	Запоминание НР		4	1360	
	2		Собственно команды подпрограммы		5	6202	Обращение к подпрограмме
	.				6	4761	
	.						
	.				7	7402	Останов
	0220	6212	Выход из	-	1160	////	Ячейка исходных данных
	1	5600	подпрограммы				
	2	////	Рабочая ячейка		1	0200	Адрес

Если пустить программу из ячейки 1153 1-го куба МОЗУ, то происходит вызов исходных данных в НР (на СМ);

обращение в подпрограмму, начинающуюся в ячейке 0200 0-го куба МОЗУ, и по возвращении из подпрограммы останов в ячейке 1157 1-го куба МОЗУ.

Команды считывания содержимого дополнительных регистров, участвующих в управлении расширенной памятью, используются обычно для организации универсальных подпрограмм восстановления старого поля данных либо старого поля команд. Команды считывания поля данных 6214 и считывания поля команд 6224 заносят содержимое соответствующего регистра в 6—8 разряды¹ НР(СМ). Считывание в эти разряды производится по правилам логического сложения, так что если перед ним в НР(СМ) вызвана «заготовка» команды установки поля данных 6201 (или установки поля команд 6202), то после считывания в накопительном регистре (в сумматоре) будет содержаться код команды восстановления соответствующего регистра, который сразу может быть заслан в нужное место программы.

Особенности применения рассмотренных команд отчетливо проявляются в типичной ситуации обращения некоторой программы к подпрограммам, расположенным в другом кубе МОЗУ². В отличие от случая минимального объема памяти, здесь необходимо дополнительно позаботиться о передаче управления в другой куб памяти и, что более существенно, о восстановлении полей основной программы при выходе из подпрограммы. Остальные функции обращения к подпрограмме выполняет, как обычно (см. гл. 3), команда ОП — при входе в подпрограмму и БП с косвенным адресованием — при выходе из нее. Состав необходимых дополнительных команд становится очевидным, как это часто бывает, при рассмотрении всего процесса от конца к началу.

Так как в общем случае оба поля основной программы и используемой подпрограммы различны, перед возвращением из подпрограммы в основную программу, т. е. до команды БП с косвенным адресованием по адресу возврата, необходимо выполнить команды $62N_{ст1}$ и $62M_{ст2}$ (где $N_{ст}$ и $M_{ст}$ — номера (старых) поля данных и поля команд соответственно). Команды $62N_{ст1}$ и $62M_{ст2}$, очевидно, должны каждый раз формироваться при работе подпрограммы, так как до этого неизвестно, куда (в какой куб памяти) надо будет вернуть управление и какое поле данных при этом установить. Для возможности формирования поля команд в подпрограмме основная программа должна произвести считывание и сообщить (с помощью некоторой стандартной рабочей ячейки) свой номер поля команд $M_{ст}$, так как после установки поля команд подпрограммы (перед передачей ей управления) старое содержимое регистра поля команд будет затерто. Старый номер поля данных $N_{ст}$ при передаче управления в подпрограмму может быть сохранен в регистре поля данных, однако формирование команд $62N_{ст1}$ и $62M_{ст2}$ при этом должно осуществляться в самом начале работы подпрограммы (с засылкой их — для исполнения — в конец подпрограммы), после чего можно установить нужное подпрограмме поле данных.

Поскольку указанные стандартные действия должны выполняться каждый раз при обращении к подпрограмме, расположенной в произвольном кубе МОЗУ, их удобно выделить в стандартные подпрограммы межкубового обмена. Очевидно, в кубах с основной программой должна быть подпрограмма засылки своего $M_{ст}$ в стандартную ячейку куба МОЗУ с подпрограммами, а в этом кубе — подпрограмма формирования команд $62N_{ст1}$, $62M_{ст2}$ и установки своего поля данных. Более того, рациональным оказывается сведение всех употребляемых подпрограмм в библиотеку с выделением в соответствующем кубе МОЗУ общего участка входа в библиотечные подпрограммы, характеризуемые лишь своим номером. В основных программах при этом приходится вслед за командой обращения отводить ячейки для исходных данных, которые используются в подпрограммах, а НР (СМ) при обращении к подпрограмме занимать под её номер.

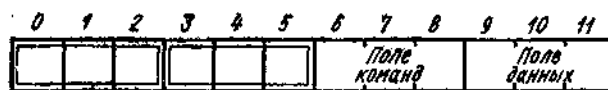
Пример. Пусть все наиболее употребительные подпрограммы собраны в библиотеку, которая расположена в j -м кубе МОЗУ. Ячейка 0170 этого куба выделена для запоминания $M_{ст}$ — номера поля команд основных программ, обратившихся к подпрограмме; входной участок библиотеки подпрограмм имеет адрес $A_{нач}$. Структурные схемы достаточно универсальных подпрограмм межкубового обмена в этом случае могут выглядеть, как показано на рис. 4.15.

Очевидно, подпрограмма межкубового обмена должна находиться в каждом кубе, из которого основные программы обращаются к подпрограммам. (Для случая $M_{ст} = N_{ст}$ эта подпрограмма несколько упрощается.) В полях $M_{ст}$ и j должны иметься «заготовки» команд восстановления поля данных и восстановления поля команд (6201 и 6202), доступные для прямого адресования из подпрограмм межкубового обмена. Следует отметить, что единый порядок обращения к библиотеке стандартных подпрограмм требует заполнения ячейки 0170 в j -м кубе и выделения места для исходных данных вслед за обращением к библиотеке и тогда, когда основная программа располагается в том же j -м поле (такова в данном случае своеобразная плата за универсальность). Наряду с этим организация всех подпрограмм в библиотеку позволяет в данном случае использовать всегда одни и те же стандартные рабочие ячейки для хранения исходных данных и результатов и, таким образом, не тратить дополнительных команд для настройки каждой подпрограммы перед началом её работы. Три рабочих ячейки для исходных данных позволяют удовлетворить достаточно широкий круг требований по передаче настроечных параметров в подпрограмму³.

Последняя особенность использования команд, управляющих работой расширенной памяти, связана с режимом программного прерывания. При расширении МОЗУ появляется дополнительный регистр запоминания полей (РЗП) емкостью 6 бит. Программное прерывание в ЭВМ с расширенной памятью при выполнении обычных (см. выше) условий его наступления происходит в следующей последовательности: вначале заканчивается выполнение очередной команды, затем содержимое регистров поля данных и поля команд переписывается в РЗП и автоматически устанавливаются 0-е поле данных и 0-е поле команд; лишь после этого имитируется выполнение команды 4000₈. Таким образом, поля прерванной программы запоминаются в РЗП, адрес продолжения в ячейке 0000₈ 0-го куба и управление передается в ячейку 0001₈ 0-го куба МОЗУ.

Для организации программ с глубиной прерывания, равной 1, предусмотрена команда восстановления полей 6244. По этой команде из РЗП старое содержимое поля данных и поля команд восстанавливается в соответствующих регистрах.

В более общем случае глубина прерывания может быть увеличена программным путем (см. гл. 8); при этом содержимое полей в момент прерывания необходимо хранить до выхода именно из данного прерывания, обеспечив в то же время возможность использования РЗП при новых прерываниях. Для этих целей содержимое РЗП можно считывать в НР(СМ) для последующей пересылки в рабочую ячейку. По команде 6234 содержимое РЗП считывается (по правилам логического сложения) в младшие разряды НР(СМ):



Перед выходом из прерывания путем логического умножения выделяют содержимое поля данных и поля команд, формируют команды восстановления полей и засылают их для непосредственного выполнения перед командами 6001, 5400.

¹ Т. е. во вторую триаду справа, где номер поля и находится в командах установки полей.

² Это соответствует, например, естественному сосредоточению всех стандартных подпрограмм в одном месте памяти.

³ Это число, впрочем, может быть пересмотрено и уменьшено за счёт использования вспомогательных таблиц и усложнения подпрограмм.

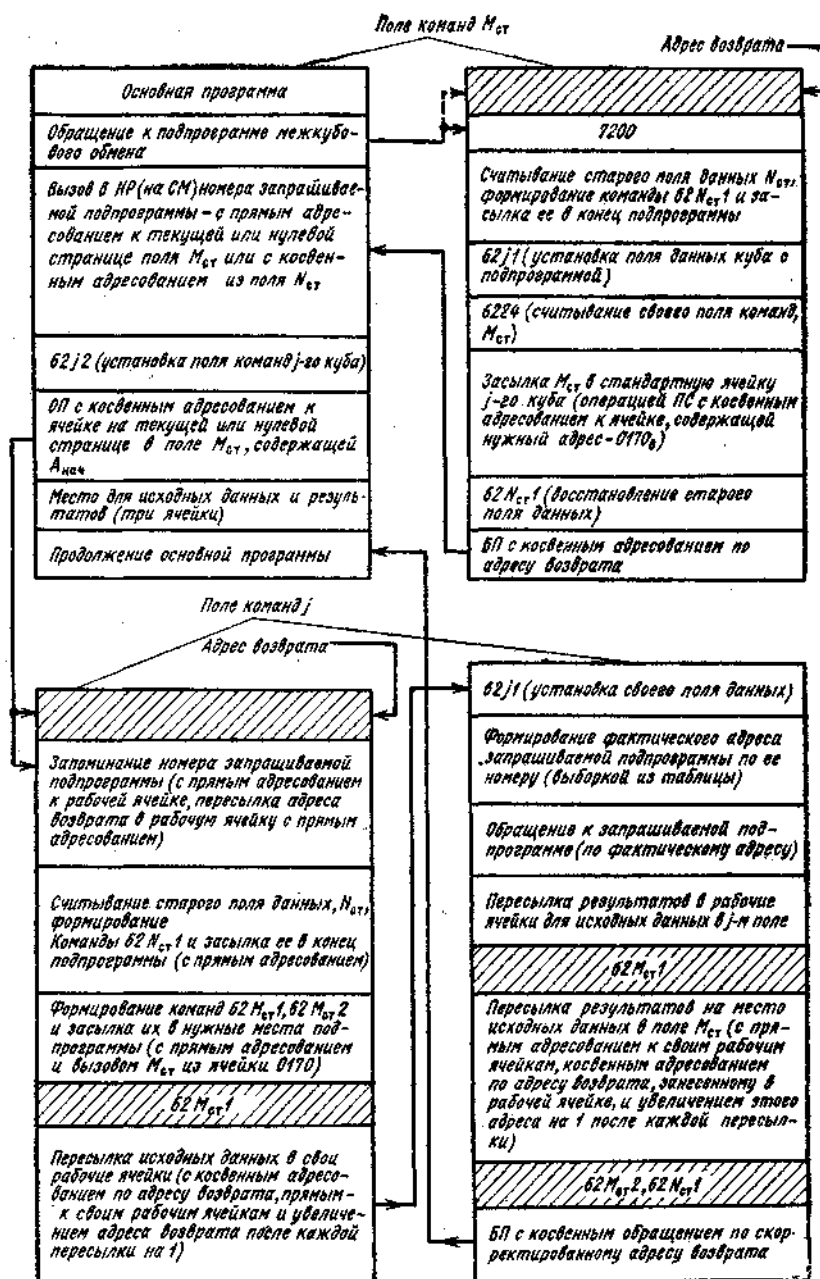


Рис. 4.15. Структурная схема обращения из произвольного куба МОЗУ к библиотеке стандартных подпрограмм, расположенной в j -м кубе; заштрихованы ячейки, содержимое которых формируется при работе программ.

4.11. К СИНТЕЗУ — ЧЕРЕЗ АНАЛИЗ

Среди задач, встречающихся в практике программиста, можно выделить две большие группы: анализ готовых программ (или их участков) и создание (синтез) новых программ. Хотя, казалось бы, именно синтез—конечная цель работы, анализ неизменно сопутствует ему даже на завершающих этапах — приходится изредка вернуться немного назад, чтобы оценить пройденный путь перед очередным шагом. Более того, трудно и нерационально учиться программировать только на основе своего собственного опыта, начинающегося с нуля. Гораздо рациональнее (даже для опытного Программиста, перешедшего к машине с новой для него системой команд) проанализировать несколько разработанных ранее программ с целью выявления основных приемов программирования и методов их реализации для конкретной машины. Полезно также обсудить возникшие при этом вопросы с другими программистами, имеющими опыт работы на данной машине. Лишь после этого целесообразно приступать к составлению собственных программ, идя по возможности от простых к более сложным.

Как при синтезе, так и при анализе программ отправной (базовой) точкой является система команд ЭВМ. Ход рассуждений при синтезе некоторых программ описан в § 4.4 и 4.10. Ниже приводятся основные положения и пример анализа готовых программ.

Основная цель анализа состоит в том, чтобы, зная общую постановку задачи, восстановить по записи команд программы

основные этапы, а также последовательность действий и приемов, реализующих её решение¹. Иными словами, требуется восстановить укрупненную структурную схему программы и схемы её отдельных блоков. Исходя из записи программы, можно идти только в обратном порядке — от мелких деталей к более крупным; полезно заранее попытаться представить, как можно было бы решить требуемую задачу.

Поскольку программа приводится обычно вместе со всеми своими рабочими ячейками, константами, формируемыми командами и т. п., одной из важных деталей анализа является выделение соответствующих ячеек памяти и определение границ расположения собственно команд программы. В первом приближении рабочие ячейки и константы могут быть выделены при беглом просмотре программы по своему специфическому содержанию (типа 0000, 7777, 0001, 0007 и т. п.) и местоположению — в начале либо в конце программы. Однако следует учитывать, что для тех же целей могут использоваться и ячейки, расположенные вслед за командами БП, а текст программы может быть распечатан уже после её работы, при произвольном содержании рабочих ячеек. Поэтому выделение рабочих ячеек и констант в первом приближении имеет целью лишь предварительное ограничение поля анализируемых кодов и нащупывание границ команд; сделанные здесь предположения в дальнейшем проверяются.

Для выделения границ собственно команд программы необходимо учитывать, что заканчивается программа остановом, закикливанием либо безусловной передачей управления за пределы поля самой программы. Для отыскания начального адреса программы, если он неизвестен, приходится исходить из предположения, что программа начинается обнулением НР(СМ), пересылкой констант, проверкой признаков, командой обращения к внешнему устройству и т.п., причем, если управление программе может быть (потенциально) передано с $| \text{НР} | = 0$, обнуление НР(СМ) в начале программы может и не производиться.

Наконец, очень существенными моментами при анализе программ являются выделение разветвлений и подпрограмм. Последние, однако, относительно просто обнаруживаются благодаря специфичным командам, связанным с использованием команд БП, ОП, УП и МК.

Для удобства анализа текста программы лучше сразу переписать на стандартные бланки для программирования на «Электронике-100». Таким образом, последовательность анализа программ сводится к следующим операциям.

1. Беглый просмотр текста программы и предварительное выделение ячеек, содержимое и местоположение которых по своей специфике сходно с используемым для рабочих ячеек и констант.
2. Выделение команд БП, ОП, УП и микрокоманд второй группы; расставление сплошных стрелок безусловной передачи управления и обращения в подпрограммы и штриховых стрелок с указанием условий пропуска очередной команды для УП и МК.
3. Выделить команды обращения к внешним устройствам и МК первой группы с записью в графе пояснений выполняемых ими действий.
4. По оставшейся части команд (куда входят лишь команды СЛ, ПС и ЛУ) уточнить местоположение исходных данных, результатов, рабочих ячеек и констант.
5. Определить границы собственно команд программы и места выхода из нее; определить и надписать около стрелок содержимое НР(СМ) при передачах управления.
6. Выделив смысл выполнения двух-трех соседних команд, начать составление подробной структурной схемы действий, осуществляемых по программе.
7. Выделить разветвления программы, типичные структурные циклы и подпрограммы.
8. Укрупнить полученную структурную схему, выделив смысл взаимодействия мелких блоков и их объединения на этой основе.

Пример. Проанализировать программу «Ввод 1» (текст программы приведен в приложении 1). Программа предназначена для ввода данных и программ с перфоленты, приготовленной в адресном формате; начальный адрес программы 7700.

«Электроника-100» работает с 8-дорожечной перфолентой. При вводе в адресном и адресно-групповом форматах шесть дорожек используются для записи кодов двух восьмеричных цифр² 7-я дорожка отводится под признак адреса, пробивкой по 8-й дорожке снабжается начало и конец данных на перфоленте. Машинное слово или адрес в 12 бит, очевидно, занимает при этом две строки перфоленты; признаком адреса (пробивкой по 7-й дорожке) снабжается первая из двух строк, используемых под код адреса (строка с двумя старшими триадами), затем идет строка с двумя младшими триадами; следующие две строки занимает код информации, которую необходимо занести по указанному адресу; затем пробивается следующий адрес (также снабженный признаком адреса), его содержимое и т. д. (рис. 4.16). Каким образом программа ввода получает и обрабатывает информацию с перфоленты, размещая её по нужным адресам, и предстоит выявить в процессе анализа.

Беглый просмотр программы позволяет предположить по специфическому содержанию (0000) последней ячейки, что она является рабочей. После записи программы на бланках для программирования на «Электронике-100», расставления стрелок передач управления и выделения команд обращения к внешним устройствам программа «Ввод 1» приобретает вид, показанный на рис. 4.17.

Адресная часть двух команд ПС (в ячейках 7715 и 7716) указывает, что последняя ячейка программы действительно является рабочей, причем одна из команд ПС адресуется к ней прямо, а другая — косвенно, используя, таким образом, содержимое этой ячейки в качестве адреса. По смыслу назначения самой программы «Ввод 1» и специфике подготовки для нее данных на перфоленте можно предположить, что при прямом адресовании в рабочую ячейку как раз и должен записываться считанный с перфоленты и сформированный код каждого очередного адреса. Это предположение, однако, должно быть подтверждено дальнейшим анализом.

Стартовый адрес программы 7700 соответствует началу расположения всей программы и собственно команд; последняя ячейка является рабочей.

Из предпоследней ячейки управление передается вновь в начало программы; команды останова в программе нет — очевидно, таким образом, что заканчивается работа программы должна закикливанием, причем место окончательного закикливания ещё предстоит определить. В программе в двух местах встречается последовательность команд, типичная для ожидания флага (ячейки 7701, 7702 и 7711, 7712). Сразу вслед за ними, как бы в подтверждение, идут команды считывания информации из буфера ввода в НР (СМ).

¹ Может быть и более общая постановка—восстановить само назначение задачи. Целиком такая проблема здесь рассматриваться не будет; следует лишь отметить, что большая часть описанных ниже действий и в этом случае является обязательной.

² Пробивка соответствует записи 1; отсутствие пробивки — 0.

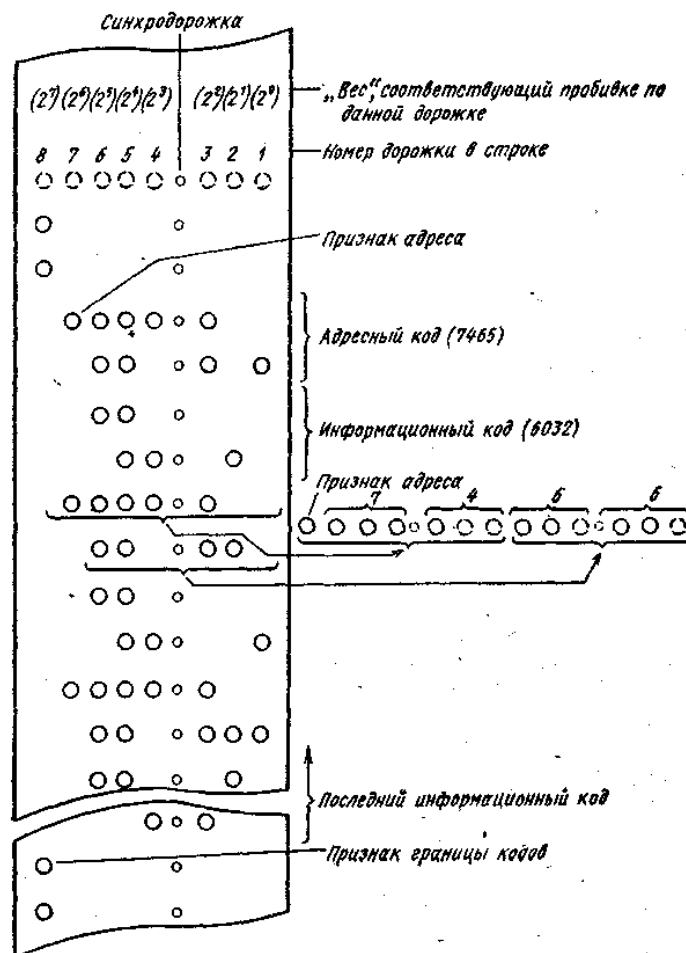


Рис. 4.16. Вид перфоленты, подготовленной в адресном формате.

Три микрокоманды первой группы (в ячейках 7704, 7705 и 7710) осуществляют сдвиг информации влево на 2 разряда каждая, причем первый сдвиг сопровождается предварительной очисткой ДР (ДРС).

„Ввод1”

Наименование(код) задачи

Лист №1

Передача управления	Абсолютный адрес	№ блока	Адрес на стран.	Команда (содержимое)	Пояснение
Флаг ввода в „1”	7700		100	6032	Запуск устройства ввода
	1		1	6031	Опрос флага ввода
	2		2	3301	БП
	3		3	6036	Считывание информации
	4		4	7106	Сдвиги с предварительной очисткой ДР
	5		5	7006	
	6		6	7510	Пропуск по условию $ HP \geq 0$
	7		7	5301	БП
	7710		110	7006	Сдвиг
	1		1	6031	Опрос флага ввода
Флаг ввода в „1”	2		2	5311	БП
	3		3	6034	Считывание информации из P_2BV
	4		4	7420	Пропуск по условию $ DP = 1$
	5		5	3720	ПС с косвенным адресованием
	6		6	3320	ПС с прямым адресованием
	7		7	5300	БП
	7720		120	0000	Рабочая ячейка ?
	1		1		
	2		2		
	3		3		
	4		4		
	5		5		

6		6	
7		7	

Рис 4.17. К анализу программы «Ввод 1»

Две команды (в ячейках 7706, 7707) образуют типичную структуру разветвления программы. Анализ условия разветвления (по неотрицательности содержимого НР), в соответствии с принятой системой представления чисел в «Электронике-100», сводится к анализу содержимого 0-го разряда НР (СМ). Ещё один условный переход (из ячейки 7714) происходит, если содержимое ДР (ДРС) равно 1. По существу здесь также имеет место разветвление программы; возможность упрощения структуры разветвления подлежит дальнейшему анализу.

Приведенные рассуждения и подготовительные операции позволяют перейти к подробному рассмотрению действий, выполняемых по программе.

Первая команда (6032) осуществляет очистку флага ввода, НР (СМ) и (при работе со считывателем с перфоленты) дает команду начать перемещение перфоленты на одну позицию (на одну строку). До окончания этого (электрохимического) перемещения флаг ввода будет оставаться в 0-м состоянии, и следующие две команды обеспечивают задержку работы быстрого процессора по дальнейшей программе. Как только устройство ввода передвинет перфоленту и считает очередную дорожку в свой регистр (буферный регистр ввода), флаг ввода устанавливается в 1, программа выходит из цикла ожидания флага. По команде 6036 происходит считывание информации из буфера ввода в НР (СМ) с предварительным обнулением последнего, обнулением флага ввода и подачей команды на перемещение перфоленты ещё на одну позицию. Это медленное электрохимическое перемещение вновь должно закончиться считыванием информации с перфоленты в буфер ввода, предыдущее содержимое которого уже переписано в НР (СМ), и установкой флага в 1 — сигналом готовности. А пока в результате описанных действий в 11-й разряд НР (СМ) попадает код от первой дорожки считанной с перфоленты строки, в 10-й — от второй и т. д.

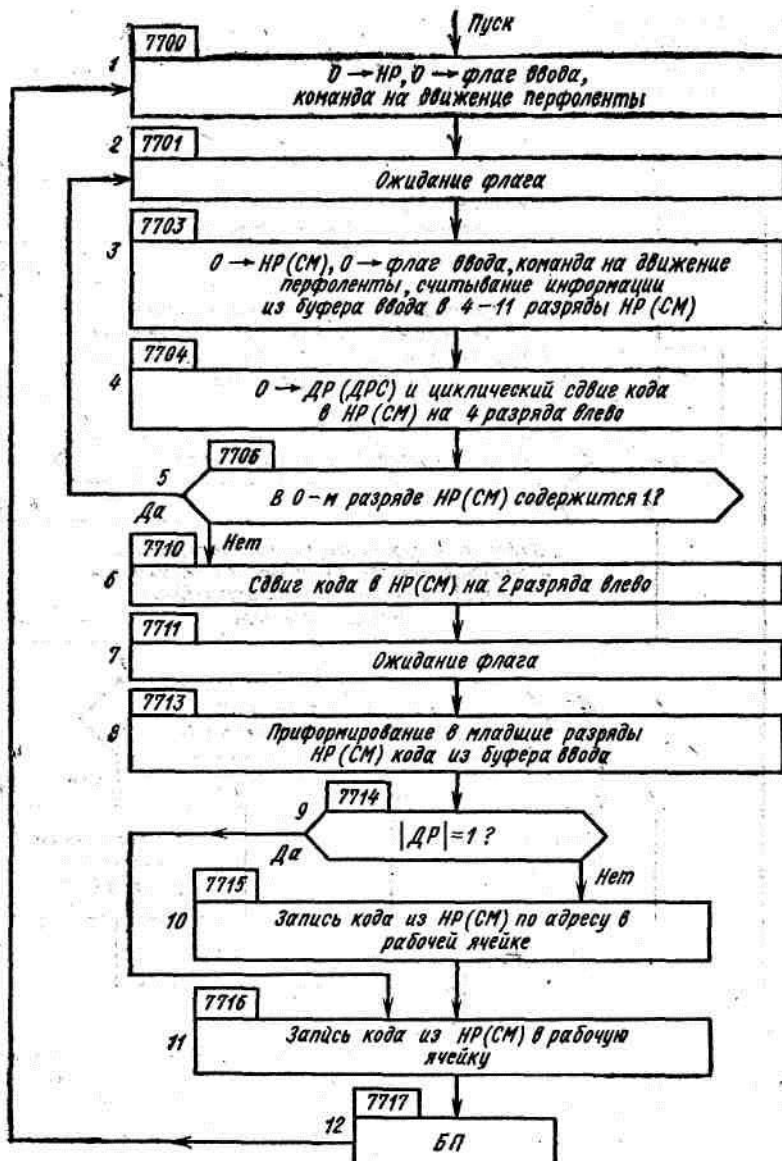
Следующими двумя командами полученное в НР (СМ) содержимое сдвигается на 4 разряда влево (с предварительной очисткой ДР) и подвергается анализу на содержимое 0-го разряда НР (СМ). До сдвига этого разряда, как нетрудно убедиться, находился в 4-м разряде НР (СМ) и, таким образом, он несет информацию о содержимом 8-й дорожки строки, считанной с перфоленты (а затем из буферного регистра ввода) по команде 6036. Переход к дальнейшей программе (по команде 7510) осуществляется лишь в том случае, если не было пробивки по 8-й дорожке, в ином случае по команде 5301 происходит возврат в начало программы, на ожидание флага, с последующим стиранием старого содержимого НР (СМ) в начале выполнения команды 6036. Этот цикл в ячейках 7701 — 7707, очевидно, будет продолжаться все время, пока через считыватель проходит перфолента с пробивкой по 8-й дорожке

Рабочий участок перфоленты начинается с адресного кода, первая строка которого снабжена признаком адреса — пробивкой по 7-й дорожке. При считывании этой строки после команды 6036 в 4-м разряде НР(СМ) оказывается 0 (так как отсутствует пробивка по 8-й дорожке перфоленты), в 5-м разряде — 1 (в соответствии с пробивкой по 7-й дорожке), в разрядах 6—11 — старшие разряды кода адреса. После сдвига на 4 разряда влево в 0-й разряд НР(СМ) попадает 0 и по команде 7510 очередная команда (5301) пропускается. При этом программа переходит к продолжению, осуществляя сдвиг влево ещё на 2 разряда и закливаясь на ожидании флага в ячейках 7711 — 7712. Выход из цикла осуществляется при установке флага ввода в 1 в ответ на последнюю команду 6036, обнулившую флаг и инициировавшую движение перфоленты ещё на одну позицию. Информация, считанная этой же командой из буфера ввода в НР (СМ), теперь сдвинута влево на 6 разрядов и по-прежнему находится в НР (на СМ), причем единица, соответствующая пробивке по 7-й дорожке, попала в ДР (ДРС), а старшие разряды кода адреса заняли соответствующее место в разрядной сетке НР (СМ).

Младшие разряды этого кода находятся на следующей строке перфоленты, уже движущейся в считывателе в результате все той же команды 6036; программа терпеливо ждет окончания движения, заклившись на ожидании флага. Наконец, строка считана в буфер ввода, флаг ввода установлен в 1, по команде 6031 программа из ячейки 7711 переходит в ячейку 7713, и по команде 6034 к старому содержимому НР (СМ) логически присуммируется содержимое буфера ввода. Поскольку в считанной строке перфоленты использованы лишь шесть дорожек для записи младших разрядов кода адреса, а шесть младших разрядов НР (СМ) обнулены при предыдущих сдвигах, по команде 6034 происходит просто приформирование младших разрядов кода адреса из буфера ввода к старшим разрядам, уже находящимся в НР (СМ) на соответствующем месте.

Теперь весь код целиком сформирован и находится в НР (на СМ). Единица в ДР (ДРС), соответствующая пробивке по 7-й дорожке в первой строке перфоленты указывает, что это код адреса. Очередная команда (7420) организует разветвление программы, пропуская (по условию $|ДР| = 1$) одну команду. По команде 3320 код из НР (СМ) запоминается в рабочей ячейке (содержимое ДР пока сохраняется), после чего управление вновь передается в самое начало программы

Следующие две строки на перфоленте занимает код информации, которую необходимо занести по указанному адресу. Считывание этой информации (в два приема по 6 разрядов) и её формирование в 12-разрядное слово происходит так же, как и при считывании и формировании кода адреса, с той лишь разницей, что при выполнении команды 7106 (из ячейки 7704) старое содержимое ДР(ДРС) стирается и сохраняется равным 0, так как информационный код не имеет пробивки по 7-й дорожке перфоленты. Таким образом, к моменту выполнения команды 7420 (из ячейки 7714) в НР (на СМ) находится сформированный информационный код, $|ДР| = 0$, и пропуска очередной команды не происходит. При выполнении команды 3720, конечно адресующейся к ячейке 120₈ на текущей странице, полученный код из НР (СМ) засылается по адресу запасенному в рабочей ячейке; НР(СМ) этой же командой обнуляется. Следующая команда ПО, с



5. ПРАКТИЧЕСКАЯ РАБОТА НА МАШИНЕ

Качество и производительность работы программиста существенно зависят от его умения практически работать на машине. Именно на машине, в конце концов, проверяется правильность функционирования программы, разработка которой может вестись в тиши кабинетов, читальных залов или в другой обстановке, привычной для творческой умственной работы. Именно машина должна «понять» и «воплотить» замыслы программиста в строгие результаты сложных расчетов или мгновенную реакцию на изменение внешних условий. Именно машина является первым «читателем» и «критиком» новой программы, всегда придирчивым и не слишком деликатным. Однако машина «искренне» старается строго выполнить все предписания программы, и лишь в силу своей пунктуальности к ошибке программиста она может добавить дополнительные неверные результаты, порчу содержимого отдельных ячеек и целых участков программы и т. п.

Общие правила работы на ЭЦВМ не слишком сложны и сводятся к знанию типовых операций и умению реализовать их на конкретной машине. Последнее, в свою очередь, требует знания пульта машины (т. е. доступных органов контроля и управления) и овладения стандартным (вспомогательным) математическим обеспечением.

5.1. ПУЛЬТ ПРОЦЕССОРА И БЛОКА УПРАВЛЕНИЯ УСТРОЙСТВАМИ ВВОДА - ВЫВОДА

На пульт процессора выведены индикаторы состояния основных регистров машины и вида выполняемых операций, а также ряд клавишных переключателей для ручного управления работой процессора. Два панельных замка служат для включения питания и блокировки клавиш управления при автоматической работе.

Вид пульта процессора «Электроника-100», конструктивно расположенного на передней панели корпуса, показан на рис. 5.1.

Центральную часть панели занимают индикаторы состояния основных регистров машины. Индикатор каждого регистра представляет собой горизонтально расположенную линейку лампочек, число которых соответствует числу разрядов в регистре. Единичное состояние разряда отображается горящей лампочкой; если содержимое разряда равно нулю, соответствующая лампочка не горит.

Для удобства считывания содержимого регистра в целом на панели нанесены дополнительные вертикальные линии, разбивающие по триадам 12-разрядные слова индикаторов.

Ниже рассмотрены последовательность расположения линеек индикаторов на панели (сверху вниз) и их краткое назначение¹.

Счётчик команд (СК) — 12-разрядный регистр, содержимое которого указывает, по какому адресу будет выбрана очередная команда при работе машины по программе либо считана или занесена информация при работе с клавиатурой пульта.



Рис. 5.1. Внешний вид передней панели процессора.

Регистр адреса (РА) — 12-разрядный регистр, указывающий текущий (последний) адрес занесения или выборки информации.

Буферный регистр (РБ) — 12-разрядный регистр, содержимое которого указывает код машинного слова, считанного или записанного по адресу, указываемому РА.

Дополнительный разряд сумматора и сумматор. Индикаторы этих регистров образуют линейку из 13 лампочек, что соответствует единому 13-разрядному коду при операциях циклического сдвига и взаимодействию этих регистров при переполнениях НР(СМ) в результате арифметических действий. С помощью сумматора осуществляются все арифметические и логические операции в машине, а также взаимодействие с буферными регистрами внешних устройств и с клавишным регистром при автоматической работе. Через сумматор (и буферный регистр) проходит информация при ручной работе с клавишами «Считывание» и «Запись» (см. ниже)².

Дополнительный сумматор — 12-разрядный регистр добавляемого арифметического блока (регистр множителя — частного) для аппаратной реализации операций умножения и деления. При минимальном комплекте оборудования этот индикатор не

¹ Назначение основных регистров машины описано в гл. I.

² В машинах «Электроника 100И» и «Саратов» сумматор в этих операциях не участвует.

задействован.

Все линейки лампочек индикаторов расположены одна под другой так, что положение соответствующих разрядов 12-разрядного кода во всех индикаторах совпадает. Снизу эту часть панели замыкает клавишный регистр (КР), каждый из 12 разрядов которого расположен под своим вертикальным рядом лампочек индикаторов. Каждая клавиша КР может фиксироваться в верхнем и нижнем положениях (угол поворота около 45°). Верхнее положение клавиши соответствует занесению единицы в соответствующий разряд КР, нижнее положение — занесению нуля. Для удобства работы с КР его клавиши выполнены разноцветными, так что три соседние клавиши, образующие одну триаду, имеют один цвет, три следующие — другой и т. д.

Клавишный регистр служит для набора 12-разрядного кода, который может автоматически считываться в НР(СМ) по специальным командам программы, а также заноситься в СК или в РБ (и далее в ячейку МОЗУ) при ручной работе с клавишами «Занесение адреса» и «Запись» (см. ниже).

Слева от поля индикаторов и клавишного регистра на панели управления процессором отведено место для индикаторов и клавиш «Поле команд» и «Поле данных». Соответствующие трехразрядные регистры используются при работе с расширенной памятью. Регистр поля команд дополняет (расширяет) разрядную сетку счётчика команд, а регистр поля данных — сетку регистра адреса. Клавиши «Поле данных» и «Поле команд» позволяют производить необходимые начальные установки состояния этих регистров перед автоматической работой машины и выполнять ручные операции по занесению и считыванию информации в нужном кубе МОЗУ. Клавиши выполнены цветными, с объединением в триады по цвету. Взаимное расположение клавиш и индикаторов полей команд и данных на панели соответствует расположению информации в НР(СМ) при считывании содержимого РЗП по команде 6234.

При минимальном комплекте оборудования клавиши и индикаторы «Поле данных» и «Поле команд» не задействованы.

Справа от индикаторов состояния основных регистров машины расположены два вертикальных ряда световых транспарантов. В левом ряду высвечивается условное обозначение одной из восьми основных операций (ЛЮ, СЛ, УП, ПС, ОП, БП, ВУ, МК), выполняемых машиной в текущий момент времени. В правом ряду высвечивается наименование одного из четырех состояний генератора главных состояний машины («Выборка», «Занесение», «Исполнение», «Разрыв»), принимаемых им на каждом машинном цикле выполнения отдельных команд. В правом же ряду, несколько книзу, расположены индикаторы трех важных триггеров («Вкл. прерывания», «Пауза», «Работа»), определяющих разрешение прерывания,

паузу при обращении к внешним устройствам¹ и (автоматическую) работу по программе.

Снизу поле этих транспарантов замыкают восемь клавиш («Пуск», «Зан. адр.», «Зап.», «Счит.», «Прод.», «Стоп», «Цикл» и «Шаг»), которые по своему горизонтальному ряду примыкают к клавишному регистру. С помощью этих клавиш осуществляется управление работой машины в различных режимах. Клавиши «Цикл» и «Шаг» могут фиксироваться как в верхнем, так и в нижнем положении, остальные клавиши подпружинены, так что всегда стремятся занять определенное положение (клавиша «Запись» — нижнее, а остальные клавиши — верхнее).

Пуск. При нажатии этой клавиши очищаются НР(СМ), ДР(ДРС), флаги всех внешних устройств, сбрасываются триггеры, разрешающие прерывание и устанавливающие паузу при обращении к внешним устройствам, триггер «Работа» устанавливается в единичное состояние и начинается работа по программе с адреса, который предварительно был занесен (или просто находился) в СК.

Занесение адреса. При нажатии этой клавиши код, набранный на КР, заносится в СК (предопределяя, таким образом, адрес, который машина будет рассматривать в качестве начального при следующих операциях).

Запись. При поднятии этой клавиши вверх код, набранный на КР, записывается по адресу, код которого был предварительно занесен в СК, после чего содержимое СК увеличивается на 1, и машина останавливается. В процессе записываемый код последовательно проходит через НР(СМ) и РБ, а код адреса — через РА, так что после записи фактически записанный код и адрес его занесения можно визуально проконтролировать по индикаторам РБ и РА соответственно (НР(СМ) в конце цикла записи обнуляется).

Считывание. При нажатии этой клавиши код из ячейки, адрес которой предварительно был занесен в СК, считывается в НР(СМ), после чего содержимое СК увеличивается на 1, подготавливая, таким образом, возможность считывания кода из следующей ячейки, и машина останавливается. В процессе считывания адрес считываемого кода попадает в РА, а считываемый код проходит через РБ и попадает в НР(СМ), что после считывания фиксируется индикаторами этих регистров.

Продолжение. При нажатии этой клавиши инициируется продолжение работы машины с сохранением содержимого всех регистров машины (до начала выполнения очередной команды или её отдельного цикла — см. ниже).

Стоп. При нажатии этой клавиши заканчивается выполнение очередной команды, после чего машина останавливается. Состояние всех регистров машины сохраняется, триггер «Работа» сбрасывается в нуль, соответствующий световой транспарант на панели процессора гаснет. Для продолжения работы по программе достаточно нажать клавишу «Продолжение».

Цикл. При поднятии этой клавиши вверх машина переходит к выполнению каждой команды с разбиением её остановками после прохождения каждого машинного цикла. Это дает возможность подробно исследовать отдельные участки программы при отладках (см. §5.4). Для исполнения следующего цикла после очередного останова достаточно нажать клавишу «Продолжение».

Такт. Поднятие этой клавиши вверх приводит к выполнению очередного такта программы, т. е. одной команды при каждом нажатии клавиши «Продолжение».

Если подняты одновременно обе клавиши «Цикл» и «Такт», выполняется работа в «Цикле».

В левом нижнем углу панели управления размещен ключ питания процессора, в правом нижнем углу расположен панельный замок — при повороте его против часовой стрелки клавиши управления работой машины блокируются и перестают влиять на работу машины; информация с КР по командам программы может считываться независимо от положения панельного замка.

Пульт процессора «Электроники-100И» незначительно отличается от пульта «Электроники-100». Внизу на панели добавлен 5-разрядный индикатор числа тактов работы добавляемого арифметического блока с надписью «Счётчик тактов». Надписи «Сумматор» и «ДРС» заменены соответственно на «Накопительный регистр» и «ДР».

¹ При этом вместо основного тактирующего генератора, который управляет работой процессора, запускается специальный генератор, вырабатывающий три опорных импульса для работы с внешними устройствами.

Все клавиши сохранили своё назначение и надписи, однако клавиши «Цикл» и «Такт» выполняют свои функции теперь не в верхнем, а в нижнем положении, так что при опускании любой из трех крайних справа клавиш машина выходит из автоматического режима работы.

Несколько изменена компоновка световых транспарантов на правой стороне панели. Все команды обращения к памяти имеют один общий транспарант, за командами обращения к внешним устройствам и микрокомандами сохранены отдельные транспаранты; все транспаранты вида команды расположены в один (верхний) горизонтальный ряд. В следующем ряду расположены индикаторы главных состояний машины; надписи «Выборка» и «Занесение» изменены на «Выборку команды» и «Выборку адреса» соответственно.

Для сопряжения процессора «Электроники-100» с внешними устройствами ввода—вывода служит специальный аппаратный блок УВВУ (управления вводными/выводными устройствами) со своим блоком питания. На переднюю панель блока УВВУ (рис. 5.2) выведены ручки управления, определяющие различные режимы работы внешних устройств, и световые индикаторы.

Ручки управления расположены на панели внизу и сгруппированы в три группы. Слева размещена группа переключателей, включающих питание на весь блок УВВУ и на каждое из внешних устройств (печатающую машинку «Консул», перфоратор «ПЛ-150», считыватель с перфоленты «СП-3»).

Средняя группа переключателей имеет следующее назначение. Левый переключатель в этой группе определяет режим работы по выходу и имеет соответствующие надписи: ПЧ — печать, ПЧ — ПР — параллельная работа пишущей машинки и перфоратора, ПР — перфорация. Средний переключатель на два положения задает режим работы по вводу: в верхнем положении, снабженном надписью КЛ, — считывание входной информации с клавиатуры пишущей машинки, в нижнем положении (СЧ) — ввод со считывателя с перфоленты.

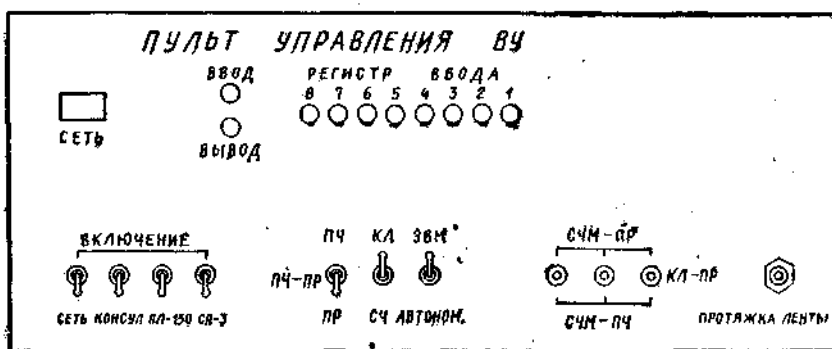


Рис. 5.2. Внешний вид передней панели блока УВВ.

Правый переключатель на два положения определяет автоматический режим работы внешних устройств от ЭВМ (надпись ЭВМ) или автономную совместную работу внешних устройств (надпись «Автоном»).

Правая группа из трех переключателей на три положения (которая должна переключаться вся одновременно) определяет связи между внешними устройствами при их автономной работе. Соответствующие надписи определяют направление передачи информации:

СЧ—ПЧ — от считывателя с перфоленты на печать;

СЧ—ПЧ — от считывателя с перфоленты на перфорацию;

КЛ—ПР — от клавиатуры (пишущей машинки) на перфоратор.

Справа отдельно расположена кнопка с надписью «Протяжка ленты» — для ручного управления перфорацией свободного конца перфоленты.

В верхней части панели блока УВВУ расположен индикатор включения питания, индикаторы состояния флагов ввода и вывода и 8-разрядный индикатор состояния буферного регистра ввода.

5.2. УСТРОЙСТВА И ПРОГРАММЫ ВВОДА — ВЫВОДА

В минимальный комплект машины входит считыватель с перфоленты типа СП-3, перфоратор типа ПЛ-150 и электрифицированная пишущая машинка типа «Консул-254». С их помощью может осуществляться автоматический ввод информации в машину (с перфоленты — с помощью считывателя и с клавиатуры пишущей машинки) и вывод информации из машины (на перфоленту или печать). В блоке УВВУ в соответствии с этим имеются два буферных 8-разрядных регистра — регистр ввода и регистр вывода. Подключение конкретных устройств ввода—вывода к этим регистрам определяется положением переключателей на передней панели УВВУ (см. § 5.1). Важно обратить внимание на два обстоятельства:

1) в пишущей машинке фактически совмещены два внешних устройства; как вводное, так и выводное. Это, в частности, определяет неоднозначность первоисточника информации, напечатанной на бумажной ленте: оператор, нажимая на клавиши машинки, печатает текст (коды соответствующих символов попадают в буферный регистр ввода, откуда по специальным командам могут быть занесены в НР (СМ)); тот же текст может быть, однако, напечатан машинкой и по командам из ЭВМ;

2) буферные регистры ввода—вывода имеют малую емкость. Восемь бит, из которых один или два несут обычно служебную информацию, позволяют закодировать не более одного символа из ансамбля символов, представленных на клавиатуре пишущей машинки, либо две восьмеричные триады при ограничении ансамбля рассматриваемых символов цифрами от 0 до 7, как это, например, имеет место при вводе программ в машинных кодах.

Последнее обстоятельство тесно связано с принципом микропрограммирования и принятой системой команд, в которой предусмотрено лишь по одной команде собственно обмена с устройствами ввода (6034) и вывода (6044)¹. Поэтому все операции

¹ Эти команды, как упоминалось в гл. 3, могут микропрограммироваться с командами обнуления соответствующих флагов и тогда имеют вид 6036 и 6046.

по вводу и выводу требуют применения специальных программ, организующих последовательный ввод и расшифровку либо упаковку введенной посимвольно информации при вводе и подготовку и собственно вывод посимвольной или упакованной информации при выводе. Эти программы разработаны до уровня стандартных программ и носят название сервисных (или обслуживающих) программ ввода—вывода. Чтобы понять их смысл и назначение, необходимо представить себе возможные ситуации, связанные с процедурами ввода и вывода, а также используемые при этом форматы представления информации.

Одной из первоочередных задач при практической работе является ввод программ в память (иногда этот процесс называют ещё загрузкой программ). Если программа написана в машинных кодах, то существуют следующие пути её ввода:

- а) прямой ввод отдельных кодов с помощью клавишного регистра (КР):
 - набрать на КР начальный адрес;
 - нажать клавишу «Занесение адреса»; — набрать на КР требуемый код;
 - поднять клавишу «Запись».

Так как в момент записи содержимое СК автоматически увеличивается на 1, запись массива кодов в последовательные ячейки памяти требует лишь однократного занесения (начального) адреса. Прямой ввод с КР является весьма трудоемкой и медленной операцией;

б) ввод с перфоленты. Этот вид ввода, очевидно, возможен лишь в том случае, если имеется перфолента с нанесенной на нее информацией. Устройством ввода служит считыватель с перфоленты, работающий в стартстопном режиме и заносящий данные о пробивках в каждой строке перфоленты в буферный регистр ввода независимо от вида закодированной информации. Носитель информации— 8-дорожечная перфолента шириной 25,4 мм. В принципе на ленте располагается 9 дорожек, одна из которых (с малыми отверстиями) является синхронизирующей. Синхродорожка располагается между третьей и четвертой информационными дорожками и делит, таким образом, перфоленту на узкую и широкую часть. Счёт информационных дорожек ведется от края узкой стороны перфоленты, начиная с 1. Обычно дорожки с меньшими номерами занимают под младшие разряды информации, заносимой в каждую строку перфоленты. Способ кодирования информации на перфоленте и соответствующие программы её преобразования к виду, используемому в машине, существенно зависят от назначения вводимых данных и требований к контролю их достоверности, возможности редактирования и т.п.¹.

Программа ввода данных и программ в машинных кодах оказывается наиболее простой в том случае, когда информация на перфоленте представлена в так называемом адресном формате, т. е. каждому информационному коду предшествует адрес его размещения в МОЗУ. Каждый код длиной в 12 разрядов (адресный или информационный) размещается в двух строках перфоленты, занимая с 1-й по 6-ю дорожки первой строки под две старшие триады, и те же дорожки во второй строке—под младшие. Первая строка адресного кода снабжается признаком адреса — в виде пробивки по 7-й дорожке. Начало и конец занесенной на перфоленту информации отмечаются пробивкой по 8-й дорожке. Об адресном формате и программе «Ввод 1», которая с ним работает, речь шла в § 4.10. Сама программа «Ввод 1» заносится в память с помощью КР. Она является наиболее короткой, но не осуществляет никаких функций контроля правильности ввода и требует явно избыточной информации (в виде указания каждого последующего адреса) при вводе кодов в последовательные ячейки памяти.

Большие удобства представляет оператору программа «Ввод 2», предназначенная для ввода информации с перфоленты, которая приготовлена адресно-групповом формате. В этом случае на перфоленте перед каждой группой кодов, заносимых в последовательные ячейки памяти, пробивается лишь начальный адрес; если информация занимает один сплошной участок в МОЗУ, то адресный код на перфоленте оказывается единственным, в ином случае вслед за последним информационным кодом предыдущей группы следует адресный код (начальный адрес) следующей группы кодов. Адресный код (как и при использовании адресного формата) имеет признак в виде пробивки по 7-й дорожке, а начало и конец перфоленты—по 8-й дорожке. Две последние строки перфоленты в адресно-групповом формате (перед началом повторной пробивки по 8-й дорожке) занимает контрольная сумма (КС) всех кодов (адресных и информационных), занесенных на перфоленту (при суммировании все единицы переполнения 12-разрядной сетки теряются). Программа «Ввод 2» подсчитывает сумму кодов, фактически считанных с перфоленты, и сравнивает с кодом, пробитым в её конце. При совпадении КС на НР (СМ) содержится 0, в ином случае $|HP| \neq 0$. Так как последней командой программы «Ввод 2» является 7402 (останов), правильность ввода (с точностью до совпадения $K\Sigma 2$) оператор контролирует визуально.

Программа «Ввод 2» обычно заносится в память с помощью программы «Ввод 1» и затем уже используется для «ввода всех остальных программ и массивов данных с контролем по совпадению К». Располагается программа «Ввод 2» в ячейках 7720—7777 и имеет стартовый адрес 7777².

Аналогичная программа для работы с расширенным МОЗУ имеет несколько больший объём и использует в качестве признака информации о номере куба одновременную пробивку по 7- и 8-й дорожкам.

Для применения программы «Ввод 1» или «Ввод 2» программа или исходные данные должны быть пробиты на перфоленте в адресном или в адресно-групповом формате соответственно, а в последнем случае ещё и снабжены контрольной суммой. Получение таких перфолент осуществляется с использованием программ вывода информации, уже размещенной в оперативной памяти машины. Разработаны программы вывода информации на перфоленту в адресном и адресно-групповом формате, причем имеется по два варианта каждой программы — один для расположения самой сервисной программы в малых адресах МОЗУ (так называемый «нижний вариант»), а другой — в участке МОЗУ с большими адресами («верхний вариант»)³.

Наиболее «узким местом» при практической работе на машине является исходный ввод информации в МОЗУ. Небольшие массивы (размером до страницы) относительно просто заносятся в память с использованием КР. Для массивов большей длины применяется ввод с помощью «Консула» и специальных программ. Один из возможных здесь приемов состоит в следующем. «Консул» и перфоратор переключают на работу в автономном режиме (переключатель «ЭВМ-Автоном» на передней панели блока УВВУ — в положение «Автоном», тройной переключатель с правой стороны передней панели УВВУ — в положение «ПЧ—ПР»). Затем печатают на машинке «текст» программы, размещая перед адресными кодами символ* (используемый как признак адреса) и отделяя коды друг от друга одним или несколькими пропусками. Соответствующие коды пробиваются перфоратором на перфоленту, по одному символу в строке. Специальная программа «Ввод 3» позволяет при вводе этой перфоленты в ЭВМ через считыватель компоновать коды отдельных триад в целые машинные слова и размещать их в

¹ Следует отметить, что малая разрядность «Электроники-100» в данном случае обеспечивает определенные удобства, связанные с использованием избыточности 8-дорожечной перфоленты для расположения на ней дополнительной служебной информации.

² Следует отметить простоту запоминания стартовых адресов в сервисных программах: для «Ввода 1 — 7700, для «Ввода 2» — 7777 («все семерки») и т. п.

³ Некоторая нечеткость терминов «нижний вариант» и «верхний вариант», по-видимому, связана с дословным переводом американских обозначений для аналогичных программ. Для достижения зрительных ассоциаций можно рекомендовать представить себе схему распределения памяти на целый куб МОЗУ (в виде прямоугольника с нарастанием адресов сверху вниз) и определить расположение основной (выводимой) программы относительно сервисной: если основная программа «ниже», то это и есть «нижний вариант» сервисной программы.

последовательные ячейки памяти, начиная с последнего указанного адреса.

Другой путь состоит в прямой компоновке восьмеричных кодов, получаемых от «Консула» при печати исходного текста, в целые машинные слова и засылке их в последовательные ячейки памяти, т. е. реализация функций программы «Ввод 3» без получения промежуточной перфоленты в кодах «Консула». Соответствующая программа¹ получила наименование «Ввод 3-У». Она использует те же служебные символы, что и «Ввод 3». В качестве признака конца ввода используется символ «К»; код, попадающий в ЭВМ при переводе строки, игнорируется. Если перед пропуском (признаком конца очередного кода) напечатано более четырех цифр, в качестве вводимого кода используется код из последних четырех цифр, если после последнего пропуска напечатано менее четырех цифр и затем пропуск, то старшие триады заполняются при вводе нулями.

Наконец, к числу сервисных относится программа распечатки восьмеричной информации, например, при документировании отлаженных программ. Существует несколько вариантов подобных программ. Достаточно удобной является программа «Распечатки», komponующая всю печать в два столбца, с указанием адресов и с дополнительной строкой пробелов при переходе к очередной октаве. В программе предусмотрены автоматические остановки после печати содержимого 100₈ ячеек, что соответствует заполнению стандартного машинописного листа при интервале печати между строками с шагом 1,5. Для продолжения печати после смены бумаги в каретке «Консула» достаточно нажать клавишу «Прод.». Разработаны два варианта программы — для размещения в малых («нижний вариант») и в больших («верхний вариант») адресах. Последовательность действий при работе с сервисными программами будет рассмотрена ниже, по мере описания действий программиста за пультом ЭВМ.

5.3. ПОДГОТОВКА И ВКЛЮЧЕНИЕ МАШИНЫ

Работе за пультом ЭВМ предшествует этап подготовки и включения машины. Необходимые при этом действия обычно выполняет инженерно-технический персонал, обслуживающий машину, однако знание этих операций полезно и программисту, чтобы грамотно организовать свою работу за пультом с самого начала.

Внешний осмотр выключенной машины позволяет убедиться в наличии всех внешних устройств и необходимых соединений между ними. Далее следует осмотр положения выключателей внешних устройств. Левый крайний ряд переключателей на панели блока УВВУ и переключатель «Сеть» на его блоке питания должны находиться в нижнем положении («выключено»), клавиша блокировки «Консула» (самая левая продолговатая клавиша в верхнем ряду клавиатуры) — в положении «О» («Консул» заблокирован).

Включение машины начинается с подключения к сети стабилизатора питания либо прямо блока питания ВВУ и процессора машины, если колебания сети в производственном помещении не превышают допустимых. Включение процессора и необходимых внешних устройств происходит раздельно.

Для включения процессора переключатель «Сеть» на задней панели процессора должен быть установлен в верхнее положение — при этом подсвечивается лампочка рядом с переключателем (обычно этот переключатель, используемый лишь обслуживающим персоналом, всегда находится в верхнем положении). Затем на передней панели процессора левый панельный замок слегка нажимают от себя и поворачивают по часовой стрелке² до упора — о включении процессора свидетельствует высвечивание лампочек основных регистров машины (в соответствии со случайным состоянием триггеров, в которое они перешли в момент включения) и звук начавших вращаться вентиляторов, встроенных в процессор. После включения процессор должен прогреться не менее 5 мин.

Для включения любого из внешних устройств должны быть переведены в верхнее положение переключатели «Сеть» на блоке питания и «Сеть» на передней панели блока УВВУ, а также переключатель соответствующего внешнего устройства в левой группе переключателей на панели блока УВВУ. При включении у считывателя с перфоленты загорается лампа и вращаются ведущие валики, перфоратор начинает издавать резкий стрекочущий звук, двигатель «Консула» работает почти бесшумно. Следует иметь в виду, что при любых переключениях на панели УВВУ «Консул» должен быть заблокирован, а процессор по возможности остановлен, так как возникающие при переключениях переходные процессы могут вызывать беспорядочную печать, приводящую к порче пишущей машинки, и сбои в работе процессора³.

После включения и прогрева процессора исправность машины и внешних устройств, а также блока УВВУ может быть проверена с помощью специальных (так называемых тестовых или контрольно-испытательных) программ, диагностирующих основные виды отказов: программы для испытания внешних устройств с УВВУ, работы памяти, работы отдельных команд, а также контрольная задача для проверки правильности работы процессора в целом. Если до последнего включения машина работала нормально, обычно ограничиваются решением контрольной задачи. Более полную проверку на тестах проводят лишь в период профилактики и при обнаружении сбоев в работе процессора или внешних устройств.

5.4. ЗАПУСК И ОТЛАДКА ПРОГРАММ.

Каждая вновь разработанная программа проходит этап запуска и отладки. Хорошо известно, что практически в любую новую программу закрадываются ошибки самого различного характера. Важнейшая задача отладки и состоит в выявлении и устранении ошибок. А так как вся отладка происходит с использованием ЭВМ, программист или оператор должен выполнять за пультом машины ряд технических операций по запуску программы и контролю правильности её работы.

Эффективность работы резко возрастает, если перед каждым выходом на машину программист записывает для себя в виде инструкции всю последовательность действий, которые предстоит выполнить за пультом⁴. Степень необходимой детализации инструкции зависит от квалификации оператора; если программист сам является начинающим оператором, в инструкцию может включаться даже описание последовательности нажатия отдельных клавиш (естественно, при этом объём отладочных операций при первых выходах на машину рационально ограничить — до приобретения необходимых навыков). Хорошо составленная инструкция

¹ Программа разработана под руководством Р. Н. Тимониной.

² В отдельных образцах машин включение происходит при повороте панельного замка против часовой стрелки.

³ Доработка конструкции позволила существенно уменьшить вероятность подобных эффектов в «Электронике-100И».

⁴ Составление таких инструкций является совершенно необходимым, если счёт по программам и их отладка производится не самим программистом, а оператором, часто совершенно не знакомым с программой и лишь обладающим хорошими навыками технической работы за пультом ЭВМ.

экономит для программиста время работы за пультом ЭВМ и освобождает его внимание от технических деталей, позволяя более полно сосредоточиться на анализе результатов, получаемых после каждого шага отладки. Если при машине не организовано постоянное хранение сервисных программ, необходимо заранее позаботиться о наличии перфолент тех программ, которые понадобятся при предстоящей работе, а также инструкций к ним с указанием занимаемого места в МОЗУ, стартового адреса и последовательности настроечных операций, выполняемых с пульта.

Запуск и отладку новой программы начинают с ввода её в МОЗУ и получения перфоленты.

При этом последовательность действий такова¹.

1. Ввести с клавишного регистра «Ввод 1».
2. С помощью «Ввода 1» ввести программу «Ввод 2»:
 - а) перевести на нижний регистр и заблокировать «Консул»;
 - б) включить питание на считыватель с перфоленты;
 - в) опустить вниз направляющую планку считывателя и ввести в паз начало перфоленты «Ввода 2», учитывая, что при считывании перфолента движется справа налево и дорожка, ближняя к оператору, считается первой (а на считывателях «Электроники-100И» — восьмой);
 - г) аккуратно поднять направляющую планку считывателя вверх и легким притягиванием конца перфоленты убедиться в правильности её фиксации в направляющем механизме; подвести под яркую часть луча пробивку по 8-й дорожке и по свечению лампочек индикатора «Регистр ввода» на передней панели блока УВВУ убедиться в правильности считывания информации;
 - д) переключатель «ЭВМ — Автоном» в средней группе тумблеров на передней панели УВВУ перевести в положение «ЭВМ» (вверх), переключатель «КЛ — СЧ» перевести в положение «СЧ» (вниз) — при этом перфолента прижимается в направляющем устройстве электромагнитом и подготавливаются цепи передачи сигналов от считывателя в ЭВМ;
 - е) занести в СК стартовый адрес программы «Ввод 1», для чего набрать на КР код 7700 и нажать на клавишу «Занесение адреса»;
 - ж) подготовить автоматический режим работы процессора для чего клавиши «Цикл» и «Такт» перевести в нижнее положение (а для «Электроники 100И» — в верхнее положение);
 - з) нажать клавишу «Пуск» — начинает работать программа «Ввод 1») и по окончании протяжки перфоленты нажать клавишу «Стоп» (остановить процессор), переключатель «КЛ — СЧ» перевести в положение «КЛ» и выключить питание на считыватель.
3. С помощью программы «Ввод 2» ввести программу «Ввод 3-У» для чего:
 - а) повторить (теперь уже с перфолентой «Ввода 3-У») действия п 2 б) — д);
 - б) занести в СК код 7777 (стартовый адрес «Ввода 2»);
 - в) нажать клавишу «Пуск»;
 - г) по окончании работы «Ввода 2» перфолента в считывателе останавливается на первой пробивке по 8-й дорожке в конце перфоленты процессор выходит на останов, имея в НР (на СМ) код разности контрольной суммы, накопленной при данном считывании всех кодов, и контрольной суммы пробитой в конце перфоленты (при правильном вводе $|НР| = 0$);
 - д) переключатель «КЛ — СЧ» перевести в положение «КЛ» и отключить питание считывателя.
4. Разблокировать «Консул», перевести его на верхний регистр.
5. Занести в СК стартовый адрес «Ввода 3-У» (7600), нажать «Пуск» — машина по программе входит в цикл ожидания флага ввода.
6. Напечатать на «Консуре»: *АВСД, где * — признак адреса; АВСД — код начального адреса программы; — пробел. Перевести строку (чтобы адресный код выделялся), а затем печатать коды программы, заканчивая каждый из них пробелом (строка переводится по мере необходимости). В целях локализации возможных ошибок рекомендуется употреблять адресные коды не только при переходе к новому сегменту программы, но и расчленять ими текст программы через 20 — 30 кодов. Обнаруженные ошибки также лучше исправлять с использованием адресных кодов.
7. По окончании ввода всего текста напечатать символ «К» — машина выходит на останов.
8. Для вывода программы на перфоленту ввести в МОЗУ программу вывода информации в нужном формате (обычно в адресно-групповом, что далее и предполагается). При этом выбирают тот вариант программы, собственные адреса которой не захватывают поле выводимой информации. Вводят нужный вариант программы вывода с помощью «Ввода 2»:
 - а) перевести на нижний регистр и заблокировать «Консул»;
 - б) повторить действия п. 3.
9. Подсчитать число отрезков программы, выводимых на перфоленту с общей контрольной суммой, перевести его в восьмеричный код; выписать на бумагу начальные и конечные адреса каждого отрезка.
10. Переключатель «ПЧ — ПЧ + ПР — ПР» на передней панели блока УВВУ перевести в положение «ПР»; включить питание на перфоратор; нажав кнопку «Протяжка ленты», выпустить чистый отрезок перфоленты из перфоратора.
11. Занести в СК стартовый адрес программы вывода (7465 или 0065 в зависимости от используемого варианта); нажать клавишу «Пуск» — перфорируется восьмая дорожка, после чего машина останавливается (запрашивая настроечные параметры).
12. Набрать на КР восьмеричный код числа выводимых отрезков; нажать клавишу «Продолжение» — машина считывает код, и вновь останавливается.
13. Набрать на КР начальный адрес очередного участка и нажать «Продолжение», затем — конечный адрес этого участка и вновь нажать «Продолжение». После такого ввода границ очередного участка следует его перфорация; вывод последнего участка программа автоматически заканчивает перфорацией контрольной суммы всего массива и участка перфоленты с пробивкой по 8-й дорожке, после чего машина выходит на останов.
14. Нажать кнопку «Протяжка ленты» — до выхода конца пробивок по 8-й дорожке из перфоратора. Повторив п. п. 11 — 13, получить сразу же дубль перфоленты, отключить питание на перфоратор, переключатель «ПЧ — ПЧ + ПР — ПР» вернуть в положение «ПЧ».
15. Проверить правильность полученных перфолент путем ввода их в машину с помощью «Ввода 2» — по совпадению КЗ (таким путем немедленно локализируются случайные сбои в работе схемы вывода, причем в отдельных случаях одна из полученных перфолент все же оказывается верной).
16. Надписать полученные перфоленты, указав наименование программы, занимаемые ячейки, стартовый адрес и дату получения.

После того, как исходная программа набита на перфоленту, она может легко, и при необходимости многократно, вводиться в машину, что существенно на этапе отладки. Сама отладка начинается с контроля соответствия набитой перфоленты исходной программе.

Содержимое отдельных ячеек МОЗУ после ввода программы может быть проверено и легко исправлено с пульта машины, для чего необходимо выполнить следующие операции:

- 1) набрать на КР адрес ячейки;
- 2) нажать клавишу «Занесение адреса» — набранный адрес переносится в СК;

¹ Ниже приводится подробная инструкция, иллюстрирующая одновременно основные действия оператора за пультом машины.

² Одновременно проверяется, переведен ли «Консул» на нижний регистр. В противном случае считывание по 8-й дорожке блокируется.

3) нажать клавишу «Считывание» — код ячейки по заданному адресу будет вызван в РБ и НР (на СМ);
 4) если считанный код оказался неверным и содержимое ячейки необходимо исправить, то действуют следующим образом:

- вновь нажимают клавишу «Занесение адреса», восстанавливая в СК нужный адрес (код которого ещё набран на КР);
- набирают на КР правильный код;
- поднимают вверх клавишу «Запись» — при этом с КР код записывается по заданному адресу, а СК увеличивается на 1.

С помощью таких операций может быть проконтролировано и исправлено содержимое небольших (объёмом до машинной страницы) программ; в случае программ значительного объёма более рациональным приемом является их распечатка с помощью одной из сервисных программ и сравнение полученного текста с исходным уже вне машины. Замеченные ошибки сводят в таблицу с тремя колонками:

Адрес	Новое содержимое	Старое содержимое

Полезную информацию в таблице несут, естественно, две первые колонки, третья колонка облегчает процесс исправления ошибок с пульта, обеспечивая дополнительный контроль правильности обращения в нужную ячейку. Если набирается более 3—5 исправлений, рационально набить их на отдельную перфоленту исправлений, которую при повторных вводах программы вводят вслед за основной перфолентой (в противном случае все исправления приходится повторять вручную с пульта при каждом новом вводе программы).

Следующим этапом отладки является проверка правильности работы отдельных участков программы как с точки зрения формального исполнения всех команд контролируемого участка, так и с точки зрения получения при этом нужных результатов. Так как в мини-машине аппаратные возможности минимальны, программисту приходится самому заботиться, в частности, об остановке машины в нужных местах программы. С этой целью в соответствующие ячейки на время отладки засылают команду останова (7402). Полезно для этого иметь свободные ячейки (с командами 7000) между основными блоками программы и даже заранее занести в нужные при отладке ячейки команды 7402, чтобы уже после отладки заменить их на 7000. В ином случае приходится с пульта заносить команды останова в ячейки, занятые полезными командами (и портить их при этом), что усложняет процесс отладки. Список всех занесенных остановов и старого содержимого (временно испорченных) ячеек сводят в таблицу, аналогичную таблице исправлений ошибок, но добавляют широкую четвертую колонку для пояснений, замечаний, указаний адресов ячеек, содержимое которых надо проконтролировать при данном останове и т. п. Таблицу заполняют в той последовательности, которую остановки будут иметь при отладке.

Последовательность дальнейших действий при контроле правильности формальной работы программы такова:

1. Ввести программу.

2. Внести (с перфоленты или с пульта) все исправления согласно списку замеченных ошибок.

3. Занести все отладочные остановки и программу продолжения работы после остановов с испорченными командами — см. ниже, п. 5в (предполагается, что ячейки 0020—0030 свободны, иначе эту программу надо перенести на другое свободное место, предварительно скорректировав адреса):

0020	3030	Запоминание содержимого НР (СМ)
1	7402	Останов — для заполнения ячеек 0026, 0027.
2	1027	Восстановление команды, испорченной остановом.
3	3426	Восстановление содержимого НР (СМ).
4	1030	БП на продолжение работы.
5	5426	Адрес продолжения.
6	////	Команда, испорченная остановом.
7	////	Команда, испорченная остановом.
0030	////	Рабочая ячейка для запоминания НР .

4. Запустить отлаживаемую программу в автоматическом режиме, для чего:

а) набрать на КР стартовый адрес программы и занести его в СК нажатием клавиши «Занесение адреса»;

б) перевести клавиши «Цикл» и «Такт» в положение, обеспечивающее автоматическую работу машины (для «Электроники-100» — вниз, для «Электроники-100И» — вверх),

в) нажать клавишу «Пуск».

5. При достижении очередного останова (т. е. в случае правильной формальной работы участка программы):

а) проверить по таблице отладочных остановов, какая команда была заменена (испорчена) командой 7402;

б) если это была команда 7000, то для продолжения отладки достаточно нажать клавишу «Продолжение»;

в) если при занесении останова была испорчена другая команда, то необходимо выполнить следующие действия:

— набрать на КР код 0020 (адрес программы продолжения работы после останова с испорченной командой) и занести его в СК нажатием клавиши «Занесение адреса»;

— нажать клавишу «Продолжение» — содержимое НР (СМ) запоминается в рабочей ячейке 0030 и машина выходит на останов в ячейке 0021;

— занести с пульта в ячейку 0026 адрес продолжения задачи (адрес последнего останова), а в ячейку 0027 — нужную команду, испорченную ранее остановом;

— занести в СК код 0022 (адрес продолжения сервисной программы);

— нажать клавишу «Продолжение» (клавишу «Пуск» нажимать нельзя, иначе будет обнулен ДР (ДРС) и погашены флаги, что в ряде случаев может привести к ошибкам).

Остановы в конце правильно проработавших участков можно устранить. Когда машина не выходит на очередной останов, что свидетельствует о наличии ошибки, необходимо предпринять ряд действий для локализации, обнаружения и устранения ошибки. Эти действия сводятся к следующему:

- 1) Ввести заново программу (так как при неверной работе программы содержимое части её ячеек могло быть испорчено).
- 2) Вновь занести все исправления.
- 3) Добавить несколько дополнительных остановов на участке от последнего останова, до которого программа формально проработала, и до того останова, на который выйти по программе не удалось.
- 4) Убрать все остановки на формально проработавших участках программы (восстановив содержимое соответствующих ячеек), составить новую таблицу отладочных остановов и запустить программу с начала.

Анализ текста программы после локализации неверно работающего участка (где машина закидывается или откуда она выходит на непредвиденный останов, или уходит мимо очередного останова) обычно позволяет обнаружить ошибку, сделанную при составлении структурной схемы программы, либо описку.

В некоторых случаях описка и ошибки, связанные, например, со сдвигом участков программы, изменением адресов рабочих ячеек, констант, могут быть найдены и на машине с использованием режима работы в «Цикле». Этот режим, позволяющий останавливать машину после исполнения каждого отдельного машинного цикла и визуально контролировать при этом содержимое всех основных её регистров, является весьма эффективным, хотя и малопродуктивным. Для перехода к такому режиму работы на «Электронике-100» достаточно поднять клавишу «Цикл» вверх (на «Электронике-100И» опустить соответствующую клавишу вниз) и после пуска машины нажатием клавиши «Пуск» продолжать её работу (на каждый следующий машинный цикл) нажатием клавиши «Продолжение». При этом следует иметь в виду, что все микрокоманды, команды обращения к внешним устройствам и команда безусловной передачи управления с прямым адресованием выполняются за один машинный цикл; за два цикла выполняются все команды обращения к памяти (за исключением БП с прямым адресованием) и команда безусловной передачи управления с косвенным обращением; за три машинных цикла выполняются все команды обращения к памяти с косвенным адресованием (за исключением БП).

Начинается исполнение каждой команды с машинного цикла «Выборка команды» («Выборка»)¹, затем, при необходимости, следуют машинные циклы «Исполнение» — для двухцикловых команд или «Выборка адреса» («Занесение») и уже затем «Исполнение» — для трехцикловых. Соответствующие световые индикаторы с надписями расположены на правой стороне передней панели процессора.

Световые индикаторы основных регистров машины на левой стороне передней панели процессора позволяют контролировать при работе в «Цикле» следующую информацию.

Для одноцикловых команд. При нажатии клавиши «Продолжение» очередная команда в цикле «Выборка команды» («Выборка») вызывается из МОЗУ и исполняется. Адрес, по которому была выбрана команда, высвечивается на световом индикаторе РА, код самой команды — в РБ, результат операций, затрагивающих содержимое НР (СМ) — в НР (на СМ). Содержимое СК увеличивается на 1 либо — для команд БП — приобретает значение, предписываемое этой командой (хотя содержимое индикатора СК после выполнения команды БП не изменяется).

Для двухцикловых команд. В цикле «Выборка команды» («Выборка») из ячейки с адресом, указываемым содержимым РА, в РБ вызывается код команды; содержимое НР (СМ) не изменяется. В цикле «Исполнение» происходит исполнение команды; в РА заносится фактический адрес ячейки, участвующей в исполнении команды, в РБ — её содержимое после выполнения команды. Операции ЛУ и СЛ изменяют содержимое НР (СМ), после исполнения команды ПС НР (СМ) всегда обнуляется, остальные команды содержимого НР (СМ) не затрагивают.

Содержимое СК увеличивается на 1 (либо изменяется в результате работы команд УП, ОП, БП) в цикле «Выборка команды» («Выборка»).

Для трехцикловых команд. Цикл «Выборка команды» («Выборка») протекает так же, как и для двухцикловой команды. В цикле «Выборка адреса» («Занесение») по указываемому адресу выбранной команды вызывается адрес ячейки, содержимое которой фактически будет участвовать в выполнении команды; указываемый адрес высвечивается на индикаторе РА, фактически используемый в дальнейшем адрес — в РБ; содержимое НР (СМ) не изменяется. Цикл «Исполнение» протекает так же, как и при двухцикловой команде (внешнее отличие состоит в том, что в цикле «Исполнение» трехцикловой команды содержимое РБ всегда переходит без изменения в РА).

Заканчивается отладка решением контрольной задачи, специально разрабатываемой каждым программистом для своих новых задач. Задача должна обеспечивать прохождение программой всех ветвей её структурной схемы. Контрольные примеры расчётных программ обычно просчитываются вручную для достаточно простых случаев.

В процессе отладки часто приходится контролировать содержимое отдельных ячеек, чаще всего после остановки машины в некоторой точке программы. В этом случае удобно использовать следующую микропрограмму (предполагается, что ячейки 0020— 0030 отлаживаемой программой не используются):

	0020	////	←	Адрес возврата
	1	3027		Запоминание (НР)
	2	1430		Вызов в НР (на СМ) содержимого контролируемой ячейки
На продолжение	3	7402		Останов
→	4	7200	}	Восстановление НР
→	5	1027		
	6	5420		Возвращение управления основной программе
		////		Рабочая ячейка для НР
	7	////		Адрес контролируемой ячейки
	0030			

¹ Одноцикловые команды на этом цикле полностью выполняются.

В отлаживаемую программу вместо останова заносят команду 4020, дойдя до которой машина уходит на описанную подпрограмму, позволяющую перед остановом вызвать в НР (на СМ) содержимое ячейки, адрес которой должен быть предварительно запасен в ячейке 0030. Для восстановления содержимого НР (СМ) и продолжения работы программы после останова в ячейке 0023 достаточно нажать клавишу «Продолжение».

Подобные микропрограммы, позволяющие визуальнo контролировать, а также распечатывать содержимое отдельных ячеек и целых фрагментов программы при различных условиях и в различных ситуациях, составляют основу построения специальных отладочных программ (одна из них входит, в частности, в комплект математического обеспечения «Электроники-100И»).

Часть обнаруженных при отладке ошибок обычно требует некоторой коррекции программы, иногда даже на уровне структурной схемы. После необходимых исправлений процесс отладки повторяют сначала.

Знание времени исполнения отдельных команд (числа машинных циклов) позволяет легко оценить общее время работы программы. В частности, если обнулить все МОЗУ и запустить машину с нулевой ячейки (передать управление в ячейку 0000), то, интерпретируя содержимое каждой ячейки как команду ЛУ с прямым адресованием, которая выполняется за два машинных цикла, машина пройдет все МОЗУ¹ за 8192 цикла, что на «Электронике-100» займёт менее 16,5 мс. Этот пример лишний раз подчеркивает, что во время отладки программист должен заранее заботиться о том, чтобы машина «не забегала вперёд» и не совершала при этом «необдуманных действий»; только отлаженная программа полезно использует высокое быстродействие процессора.

5.5. МАЛЕНЬКИЕ ХИТРОСТИ

Разработка и запуск новых программ требуют от программиста не только знания системы команд и основных приемов программирования на данной машине, но и определенной изобретательности в борьбе с мелкими трудностями, возникающими на каждом шагу. Соответствующие приемы вырабатываются со временем у каждого программиста, однако некоторые из них полезно знать заранее.

Экономия ячеек памяти при переходе на следующую страницу. Если переход со страницы на страницу совершается с нулевым НР (СМ) и место на текущей странице исчерпано, может использоваться следующий приём. В последней ячейке страницы (по адресу на странице 177₈) располагают константу, у которой старшая триада содержит 0₈ (обычно хотя бы одна такая константа в пределах страницы используется — необходимо лишь поместить её заранее по адресу 177₈ на странице). При переходе на следующую страницу используют команду 5377, т. е. безусловную передачу управления в последнюю ячейку памяти на текущей странице. В таком случае машина начинает интерпретировать содержимое этой ячейки как команду (хотя ранее оно с успехом использовалось как константа). Так как старшая триада кода в ячейке по условию содержит 0₈, выполняется операция логического умножения, результатом которой (0000₈) предопределен нулевым содержимым НР (СМ) перед выполнением команды 5377. Таким образом, состояние НР (СМ) и ДР (ДРС) не изменяется, а на следующем машинном цикле управление из последней ячейки на текущей странице естественным путем (см. огл. 1, работа СК) передается в следующую ячейку, которой является как раз требуемая нулевая ячейка памяти на следующей странице.

«Заплаты». В некоторых случаях возникает необходимость добавить несколько команд (поставить «заплату») в середине отлаживаемой программы, когда уже исчерпан весь резерв ячеек МОЗУ вблизи требуемого места. Тогда, заменяя командой БП или ОП последнюю команду перед «заплатой», передают управление на свободное место, где в первую очередь выполняют испорченную (командой БП или ОП) команду основной программы, а затем размещают вновь добавляемые команды, завершая всю цепочку командой БП на продолжение основной программы. Если «заплата» не может быть размещена на текущей или нулевой странице, приходится портить две ячейки основной программы, занимая одну из них командой БП с косвенным адресованием, а другую (обычно следующую) — начальным адресом «заплаты», и усложнять программу самой «заплатой» за счёт косвенного адресования, если она содержит команды обращения к памяти.

Однократная работа участка программы. Один из простых приемов состоит в использовании разветвления и специально вводимого признака, обнуляемого в исходном состоянии и перед тем, как участок программы должен проработать повторно (рис. 5.3). Указанная схема реализуется пятью командами, и одна ячейка МОЗУ занимается под признак.

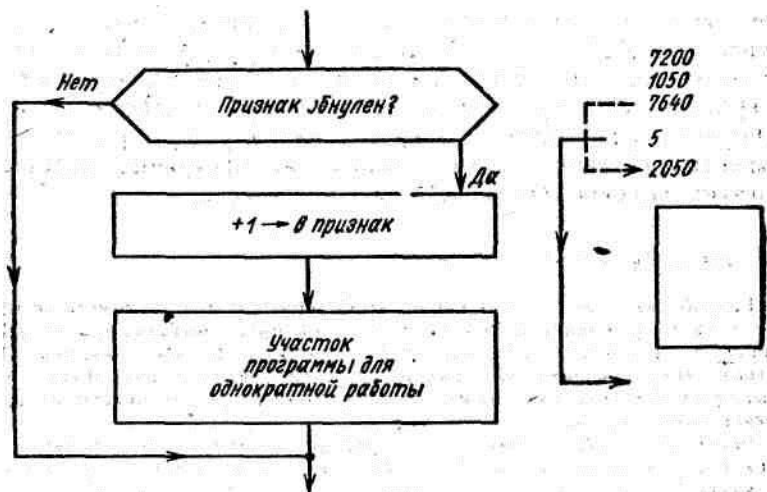


Рис. 5.3. Структурная схема организации однократной работы участка (ячейка 0050 отведена под признак).

Временные задержки. В простых системах программе может быть поручена работа и по отсчёту временных задержек (при отсутствии таймера). Для этого зацикливают команду УП:

¹ Если команда, выбранная из ячейки 7777₈, не является командой ОП или БП, машина добавит в СК очередную единицу и подготовит при этом переход в ячейку 0000₈ того же куба МОЗУ (единица переполнения СК теряется).



Цикл длится до тех пор, пока не переполняется ячейка (обозначим её через «у»), к которой адресуется команда УП. При прямом адресовании команда УП занимает два машинных цикла и команда БП — еще один. Если в исходном состоянии $|y| = 0$, то вся задержка длится $3 \times 4096 = 12288$ машинных цикла (24,5 мс для «Электроники-100» или 18,4 мс для «Электроники-100И»). Если требуется меньшая задержка, то в ячейку у (перед входом в программу задержки) заносят некоторый исходный код, отличный от нуля; для больших задержек организуют счётчик в двух ячейках (диапазон возможных задержек расширяется при этом почти до 100 с).

К сожалению, на время задержки машина никакую другую работу выполнять не может, поэтому такой режим не рационален, хотя и бывает полезен в отдельных случаях.

Отыскание нужного адреса на перфоленте. Для лент, приготовленных в адресном формате, эта операция сводится к прямому чтению кодов адресных пробивок. В случае адресно-группового формата используют то обстоятельство, что строки на перфоленте отстоят друг от друга на 2,54 мм¹, а один код занимает две строки; таким образом, каждый очередной код отстоит от предыдущего на 5 мм. Остается лишь рассчитать, на какую длину надо сдвинуться вдоль перфоленты от последнего адресного кода (переведя разность адресов к десятичной системе счисления и умножив её на 5 мм).

Засылка сервисных программ в новое место МОЗУ. Большинство сервисных программ укладываются в пределах страницы и поэтому могут быть безболезненно перенесены на целое число страниц. Этим можно воспользоваться: не набивая новой перфоленты, достаточно взять вариант такой программы, располагающейся в больших адресах (так называемый «верхний вариант»), и на перфоленте временно закрыть одну из пробивок, соответствующую 0 — 4 разрядам адресного кода. Последнее легко сделать, если расположить перфоленту на твердой гладкой поверхности, взять из кармана для отходов в перфораторе один из выбитых при перфорации кружочков, приложить его к нужному отверстию на перфоленте и запрессовать в отверстие легким заглаживанием. Перфолента с такой заплатой вполне пригодна для неоднократного ввода в считыватель, но благодаря изменению адресного кода её содержимое вводится в новое место МОЗУ, а контрольная сумма, подсчитанная при вводе, не совпадает с пробитой на перфоленте как раз на величину кода временно закрытых пробивок. Вместе со сдвигом программы, естественно, сдвигается и её стартовый адрес.

Запоминание и воспроизведение текста. Относительно простая программа с хорошими демонстрационными свойствами позволяет вводить в машину и запоминать (символ за символом) текст, печатаемый на пишущей машинке. с последующим его воспроизведением:

Запоминание			Воспроизведение			
Пуск	0020	1027	Пуск →	0030	1027	0040 6041
→						
	1	3017		1	3017	1 5040
	2	6031		2	1417	2 7200
	3	5022		3	1044	3 5032
	4	6036		4	7450	4 7403
	5	3417		5	7402	5 0375
	6	5022		6	1045	
	7	0077		7	6046	

После занесения программы в память и запуска её из ячейки 0020 машина закидывается в ячейках 0022 — 0023 на ожидании флага ввода. Код каждого вводимого символа занимает последовательные ячейки памяти, начиная с 0100_в. Заканчивается текст символом 0, используемым программой воспроизведения как признак конца текста. Если при вводе ошибочно напечатан не тот символ достаточно остановить машину, считать содержимое ячейки 0017_в, уменьшить его на 1, запустить машину «Пуском» или «Продолжением» с ячейки 0022 и печатать текст дальше, начиная с исправляемого символа. По окончании ввода текста нажать «Стоп».

Для воспроизведения текста достаточно запустить машину с адреса 0030. По окончании печати текста происходит автоматический останов в ячейке 0035. Воспроизведение можно неоднократно повторять, вновь запуская машину с адреса 0030. Объёма одного куба МОЗУ достаточно для запоминания свыше 4000₁₀ символов (стандартный машинописный лист вмещает обычно менее 2000₁₀ символов).

Следует отметить, что процессы запоминания и воспроизведения текста могут быть существенно разнесены во времени. Более того, программа вместе с подготовленным текстом может быть выведена на перфоленту для долговременного хранения, а для демонстрационных целей программу легко доработать так, чтобы воспроизведение начиналось тотчас же или с некоторой задержкой после поднятия какой-либо клавиши КР (перед этим машина может работать на тесте, эффективно используя световые индикаторы передней панели процессора).

Обнуление МОЗУ (до адреса 7700). Эта операция легко реализуется с помощью следующего цикла:

Пуск →	7700	1310	}	Формирование счётчика цикла
	1	3311		
	2	3312	}	Засылка $A_{нач}=0$ в рабочую ячейку Обнуление очередной ячейки Переадресация
	3	3712		
	4	2312		
	5	2311		
	6	5303	}	Конец цикла?
	7	7402		
	7710	0100	}	Останов Константа и рабочая ячейка для счётчика

¹ С точностью $\pm 0,05$ мм, регламентируемой ГОСТ 10860—68.

² Анализ приведенной программы и разработку интересных тестов для размещения их на свободном месте МОЗУ (ячейки 0046—0077) рекомендуется провести читателю самостоятельно в качестве полезного упражнения

1	////	цикла
2	////	Рабочая ячейка текущего адреса

Поиск команд, портящих содержимое заданной ячейки. Такая задача является достаточно распространенной при отладках новых программ. Программа, приведенная в приложении 2, позволяет обнаружить команды, портящие содержимое ячейки, адрес которой набран на КР. Работа программы основана на поиске кодов с командами УП, ПС и ОП, обращающихся к ячейке по заданному адресу прямо или косвенно. При обнаружении такого кода его адрес вызывается в НР (на СМ) и машина останавливается в ячейке 0130₈. Для продолжения поиска достаточно нажать клавишу «Продолжение». Останов в ячейке 0061 происходит в конце работы программы. Следует отметить, что приведенная программа решает задачу в статическом состоянии программы и не дает возможности обнаружить виновника порчи ячейки по заданному адресу, если он замаскирован в цикле. Построение более универсальной программы такого рода наталкивается на сложность формального программного обнаружения циклов и выделения пределов изменения его текущих адресов. Один из возможных путей преодоления такого препятствия состоит в преобразовании исходной сервисной программы в подпрограмму и введении в «подозрительный цикл» заплаты, обеспечивающей обращение к этой подпрограмме после каждой переадресации.

6. СРЕДСТВА АВТОМАТИЗАЦИИ ПРОГРАММИРОВАНИЯ

Ручное программирование в кодах машины является весьма трудоемкой и часто малопродуктивной процедурой. Дополнительные сложности возникают при перемещении уже написанных и отлаженных программ на новое место в МОЗУ, при стыковке отдельных участков программ, при необходимости внесения добавлений или исправлений в программу и т. п. Большое число приемов программирования может быть формализовано и, следовательно, передано для реализации самой ЭВМ. При этом появляется возможность автоматизации программирования.

Так как каждый тип машин оперирует конкретной системой команд и правилами представления чисел, а при автоматизации программирования используются некоторые более общие понятия, возникает несколько уровней описания одной и той же задачи (минимум два). Самый низший из этих уровней по-прежнему есть язык машинных кодов конкретной машины (так как только он и остается «до конца понятным» машине). Язык более высокого уровня описывает задачу с той или иной степенью формализации и приближения к естественному человеческому языку (техническому, научному, разговорному и другим—в зависимости от характера решаемых задач). Перевод исходного описания задачи к языку низшего уровня осуществляется с помощью специальных программ-трансляторов, выполняющих разнообразные функции формального контроля правильности исходного описания, редактирования программы, сочленения отдельных фрагментов программ, распределения памяти, установления фактических адресов и т. п. Более сложные трансляторы по сильно формализованному исходному описанию генерируют программу на языке низшего уровня, собирая (компилируя) её по заранее заготовленным частям (функции формального контроля, редактирования и прочие здесь также имеются). Трансляторы с языков высокого уровня осуществляют многократный просмотр и обработку исходного описания задачи, иногда с выдачей промежуточного описания на языке более низкого уровня.

Переход к средствам автоматизированного программирования сопровождается для программиста появлением в работе двух новых этапов: овладением формализованным языком описания задач и овладением техникой работы с транслятором. Взамен он передает транслятору значительную часть кропотливой работы по соблюдению частных правил работы на конкретной машине и получает дополнительные средства формального выявления части ошибок, простого добавления и изъятия фрагментов программы, стыковки и увязывания их между собой и др.

Следует отметить, что языки высокого уровня (типа ФОРТРАН) оперируют настолько обобщенными понятиями и имеют настолько развитые трансляторы, что программист может успешно составлять описание задач и отлаживать программы, совершенно не вдаваясь в детали преобразования исходного описания при работе на конкретной машине и даже не знакомясь с основами ручного программирования на ней. Вместе с этим требования универсальности языка часто приводят к тому, что полученные после трансляции рабочие программы не всегда являются оптимальными и по эффективности могут уступать программам, составленным для той же задачи вручную (опытным программистом). С другой стороны, знание основных приемов ручного программирования помогает программисту составлять более эффективные программы и с использованием специальных языков. Это требование оказывается обязательным при работе с низшими языками программирования—так называемыми ассемблерами или автокодами, в которых коды операций конкретной машины заменены символическими обозначениями (обычно сокращенными наименованиями соответствующих команд).

Специфика использования языков программирования на малой машине состоит в том, что приходится особо заботиться об одновременном размещении в оперативной памяти транслятора и обрабатываемого участка описания задачи. Поэтому здесь часто применяется трансляция за несколько проходов и (или) выдача промежуточного описания на перфоленду¹. Для ЭВМ с системой команд, принятых для «Электроники-100», разработаны следующие языки программирования¹: ассемблер, макроассемблер, ФОКАЛ, ФОРТРАН.

Основные особенности работы с использованием средств автоматизации программирования отчетливо прослеживаются уже на примере применения ассемблера по сравнению с обычным программированием в машинных кодах. Текст исходного описания задачи на ассемблере составляется в символической форме, что существенно расширяет ансамбль символов входного языка.

Кроме чисто цифровой информации, разрешается использовать весь латинский алфавит, знаки сложения, вычитания, приравнения и др. При работе на отечественных машинах в комментариях (см. ниже) могут применяться и буквы русского алфавита, но среди символов печатающего устройства, которым комплектуется ЭВМ, есть и такие, которые запрещены при работе с ассемблером (например, скобки). Числовые данные и адреса во входном описании могут иметь не только восьмеричную, но и десятичную форму представления; предусмотрена также возможность прямого указания знака числа. Обозначения команд приобретают вместо цифровой формы символическую, что преследует цель облегчения их смыслового восприятия и уменьшения количества ошибок при написании программы. (Исходная символическая форма всех команд зафиксирована в так называемой постоянной таблице символов транслятора.)

В командах обращения к памяти группа символов, соответствующая коду операции (виду команды), и адресная часть

¹ Помимо упоминавшихся ранее ЭВМ «Электроника-100И» и «Саратов», аналогичную систему команд имеет американская машина типа PDP-8, для которой названные языки и были первоначально разработаны. По данным зарубежной печати, PDP-8 имеет также транслятор с АЛГОЛа-60.

разделяются одним или несколькими пробелами. Адресная часть может быть записана либо в виде кода адреса, либо в виде символического адреса (сочетания букв и цифр, начинающегося с буквы и не содержащего пропусков и запрещенных знаков). Для указания косвенного адресования и (или) адресования на нулевую страницу перед адресной частью дополнительно записывают специальные псевдокоманды (команды транслятору) I и Z, выделяя каждую из них пропусками. Для безадресных команд обращения к внешним устройствам и микрокоманд каждое из отдельных действий имеет своё символическое обозначение, и при использовании микропрограммируемых команд каждое из требуемых действий записывается друг за другом (с пропуском между ними).

Ряд приёмов при составлении исходного описания на ассемблере не имеет аналогий в ручном программировании на «Электронике-100». К их числу относится использование меток, относительного адресования, прямого присвоения числовых значений переменным, прямое указание арифметических действий сложения и вычитания, а также применение комментариев и псевдокоманд.

Метка позволяет символически определить в исходном описании местоположение (и в конечном счёте адрес) того или иного оператора или отдельной ячейки (используемой, например для счётчика) и затем использовать это символическое обозначение как адрес в последующих операторах. Меткой, как и символическим адресом, может служить любая группа букв и цифр, начинающаяся с буквы и не содержащая в своем составе пробелов и запрещенных для использования знаков; транслятор воспринимает шесть первых знаков. В исходном описании конец метки отмечается запятой.

Относительное адресование особенно удобно, если необходимо указать адрес ячейки, расположенной в программе несколькими командами раньше или позже, например, при организации циклов, обхода участков программы и т. п. Для этого достаточно в адресной части оператора поставить точку (что эквивалентно использованию значения счётчика текущего адреса, см ниже) и затем требуемому константу сдвига со своим знаком.

Вновь вводимым символам и символическим обозначениям ассемблер позволяет прямо присваивать числовые значения с использованием знака равенства; при этом слева от него ставится определяемый символ, а справа — число, выражение или ранее определенный символ. В одной записи можно производить многократное присваивание одного и того же значения нескольким переменным, но с использованием не более двух знаков равенства.

Числа и символы с помощью знаков сложения, вычитания и пропуска могут составляться в выражения, что открывает дополнительные возможности при описании требуемых операций, а также бывает полезно для определения адресов без введения дополнительных меток. Применение пропуска в выражении эквивалентно указанию операции логического сложения. Заканчивается выражение возвратом каретки или точкой с запятой.

Введение комментариев облегчает процесс отладки программы и работы с ней. После любого оператора программист может записать комментарий, выделив его начало наклонной чертой; заканчивается комментарий возвратом каретки. Все введенные таким образом комментарии при трансляции игнорируются и никак не влияют на результирующую рабочую программу. Комментарии могут записываться и на русском языке.

С помощью псевдокоманд производится настройка транслятора на все время его последующей работы либо на время трансляции отдельных операторов исходного описания. Кроме уже упоминавшихся псевдокоманд указания на косвенную адресацию и адресования на нулевую страницу, в ассемблере «Электронике 100» используется ещё семь псевдокоманд. Три из них служат для изменения таблицы постоянных символов (см ниже). Две псевдокоманды указывают на используемое основание числения всех последующих цифровых данных (DECIMAL — десятичные числа, OCTAL — восьмеричные; первоначально транслятор настроен так, что воспринимает все числа как восьмеричные). Псевдокоманда PAUSE служит указателем того, что часть исходного описания находится на другой перфоленке (встретив эту команду, транслятор выводит машину на останов — после заправки новой ленты в считыватель достаточно нажать клавишу «Продолжение»). Псевдокоманда FIELD n указывает номер куба МОЗУ, куда должна вводиться рабочая программа (n — целое число или ранее определенный символ; $0 \leq n \leq 7$).

Запись исходного описания задачи на ассемблере выполняется в определенной последовательности. Все описание разбивается на отдельные операторы, называемые утверждениями. Каждое утверждение начинается с новой строки и заканчивается возвратом каретки. В каждом утверждении важна лишь последовательность расположения элементов, сами элементы могут располагаться свободно в произвольных местах строки. Однако для облегчения редактирования, отладки и работы с программой рекомендуется отвести для однотипных элементов утверждения определенное место в строке. В утверждении различают четыре типа элементов, которые опознаются по порядку их появления в утверждении и по знакам, которые следуют за элементом или предшествуют ему:

метка, команда операнд/примечание.

(Приведенный здесь способ записи, характерный для ассемблера, одновременно указывает, что метка опознается по запятой, следующей за символом, команда выделяется пропусками, а за примечание (комментарий) транслятор принимает все, что следует от наклонной черты до возврата каретки.) В утверждении должен входить хотя бы один из таких элементов. В качестве признака конца утверждения, не содержащего комментариев, может служить также точка с запятой (;).

Обо всех элементах утверждения, кроме операндов, говорилось выше. Операнды представляют собой восьмеричные или символические адреса, константы или выражения, а также аргументы псевдокоманд. Смысловое значение операнда зависит от вида утверждения. Операнд после команды обращения к памяти служит адресом; если утверждение не содержит команды, то операнд в виде константы или выражения представляет собой информацию для записи в очередную ячейку памяти (при этом транслятору поручается заменить символическое обозначение или выражение его восьмеричным эквивалентом). Заканчиваются операнды точкой с запятой, наклонной чертой или возвратом каретки.

Помимо утверждений, исходное описание может содержать псевдокоманды, адресные коды, операторы прямого присваивания. Заканчивается исходное описание специальным символом ($\equiv$) — признаком конца программы.

Для работы с ассемблером важно знать некоторые особенности работы его транслятора. Все вводимые и используемые символы (как отдельные знаки, так и неделимые группы знаков) программа-транслятор помещает в специальные таблицы. В таблицах имеется зона постоянных (для всех программ) символов и зона переменных символов, вводимых в исходном описании. Таблица постоянных символов включает символические обозначения псевдокоманд, команд обращения к памяти, микрокоманд, команд блока плавающей запятой, команд работы с низко- и высокоскоростными устройствами ввода — вывода, символическое обозначение команд для работы с устройством управления накопителем на магнитной ленте и другие — всего 75 символических обозначений. Таблица переменных символов позволяет вместить ещё 593 символических обозначения. Под таблицы постоянных символов в МОЗУ заняты ячейки 226₈, — 2741₈, для переменных символов 2742₈ — 7445₈. Каждому символу (и постоянному и переменному) в таблицах отводится по четыре ячейки. В первых трех ячейках хранится в упакованном виде (коды двух знаков в одной ячейке) шесть первых знаков соответствующего символического обозначения. Для постоянных символов в четвертой ячейке хранится восьмеричный код команды; для переменных символов — восьмеричный код числового значения, если оно определено прямым присваиванием, либо восьмеричный адрес символа, если символ определен как метка.

Таблица постоянных символов вводится в МОЗУ одновременно с программой транслятора; таблица переменных символов формируется транслятором в результате обработки исходного символического описания конкретной программы. В распоряжении программиста имеются три псевдокоманды, позволяющие изменять состав таблицы постоянных символов для своей задачи.

Так, если в исходном описании встречается псевдокоманда EXPUNGE, транслятор стирает всю таблицу постоянных символов, кроме псевдокоманд. Псевдокоманда FIXTAB добавляет все символы, определенные перед этой командой, к постоянной таблице символов. По псевдокоманде FIXMRI (вместо стертых командой EXPUNGE постоянных символов) можно определить новые символические обозначения команд, для чего за FIXMRI делается один или несколько пропусков, затем пишется новое символическое обозначение определяемой команды, знак = и восьмеричный код команды. Псевдокоманда FIXMRI с новыми аргументами повторяется для каждой определяемой команды; все команды обращения к

памяти следует определить до введения любого другого символа. За счёт применения этих псевдокоманд можно определить для дальнейшего использования только те команды, которые будут использованы в данной программе, увеличив, таким образом, ёмкость переменной таблицы символов транслятора; ввести более удобные, с точки зрения программиста, символические обозначения команд и т. п. При обработке исходного описания задачи транслятор присваивает последовательные адреса всем формируемым командам, константам и другим данным.

Счёт адресов транслируемых команд (команд рабочей программы) выполняется в ячейке, называемой счётчиком текущего адреса; первоначально транслятор устанавливает этот счётчик на 0200. Значение счётчика адреса доступно для изменения и использования при составлении исходного описания: напечатать * и абсолютный восьмеричный адрес ячейки, программист указывает, куда следует записать очередное слово программы в машинных кодах; точка, за которой следует выражение, указывает на адрес, отстоящий от текущего значения счётчика адреса на величину последующего выражения (см. выше). Увеличение счётчика адреса происходит после считывания утверждения, в котором содержится информация для записи в память; (псевдокоманды и утверждения присваивания в рабочую программу не записываются, а реализуются транслятором немедленно, не изменяя, таким образом, значения счётчика адреса). Опознав метку (по запятой, которой она заканчивается), транслятор немедленно присваивает ей значение, равное содержимому счётчика текущего адреса. При обработке отдельных утверждений транслятор выполняет все необходимые выборки из таблиц постоянных и переменных символов, подстановки, замены символических адресов, формирование адресных частей команд обращения к памяти, вычисления значений выражений и прочее, пока не сформирует одну или несколько машинных команд генерируемой рабочей программы (на число сформированных команд увеличивается значение счётчика текущего адреса).

Транслятор содержит блок диагностики ошибок, формально контролирующей правильность составления исходного описания. При обнаружении недопустимых знаков, прямых обращений за пределы страницы, повторного определения символов или метки, при попытке использования не определенного ранее символа, а также при переполнении таблиц символов трансляция прекращается и печатается соответствующая диагностика ошибки.

Исходное описание в символической форме набивают на перфоленте, используя автономный режим работы «Консула» с перфоратором. Трансляция исходной программы в рабочую осуществляется в два прохода с необязательным третьим проходом. О номере прохода и виде используемого перфоратора транслятор извещается двоичным кодом, набираемым на КР: в разрядах 0 и 1 — код номера прохода, в разряде 11 — код вида перфоратора; в машинах с ПЗУ 10-й разряд КР используется дополнительно для указания вида кодирования исходной символической перфоленты. .

Важно отметить, что всю работу по обработке отдельных утверждений транслятор разбивает на элементарные этапы. Результат обработки тут же выдается на печать или перфорацию, освобождая транслятор для считывания и обработки очередного утверждения с перфоленты, которая служит не только носителем исходной информации, но и своеобразным долговременным запоминающим устройством. Благодаря этому удается иметь в МОЗУ небольшие буферные зоны для вводимой и выводимой информации и значительные зоны памяти отводить для таблиц символов и рабочих подпрограмм транслятора.

При первом проходе всем вводимым символам присваиваются числовые значения. Если обнаруживаются ошибки, то печатаются диагностические сообщения. В конце первого прохода печатается таблица символов с присвоенными им восьмеричными значениями. При втором проходе вновь считывается символическая лента, транслятор преобразует символические команды в машинные и результат преобразования (рабочую программу в машинных кодах) перфорирует в адресно-групповом формате. При третьем проходе символической ленты происходит распечатка полученной при трансляции рабочей программы и исходного описания на символическом языке.

Ниже приводятся распечатки программы, написанной на ассемблере (для удобства сравнения взят пример, уже использовавшийся в гл. 4).

/ ПРОГРАММА ПОДСЧЁТА СУММЫ КОДОВ*			
	*400		/СТАРТОВЫЙ АДРЕС 0400
0400	7000	7000	/РЕЗЕРВ
0401	7200	CLA	/ЗАСЫЛКА
0402	1376	TAD A	/132 В СЧЁТЧИК
0403	3375	DCA	/ЦИКЛА
		COUNTER	
0404	1375	TAD B	/ЗАСЫЛКА НАЧАЛЬНОГО
0405	3374	DCA ADD	/АДРЕСА В РАБ. ЯЧЕЙКУ
0406	7000	NOP	
0407	7000	NOP	
0410	1774	TAD ADD	/СОБСТВЕННО СЛОЖЕНИЕ
0411	7000	NOP	
0412	2374	1SZ ADD	/ПЕРЕАДРЕСАЦИЯ
0413	2375	ISZ COUNTER	/КОНЕЦ
0414	5210	J MP.—4	/ЦИКЛА
0415	7000	NOP	
0416	3150	DCA Z	/ЗАСЫЛКА РЕЗУЛЬТАТА
		RESULT	
0417	7402	HLT	/ОСТАНОВ
		RESULT = 150	
	*0574		
0574	0000	ADD, 0	/РАБОЧАЯ ЯЧЕЙКА
			ТЕКУЩЕГО АДР.
0575	0000	COUNTER, 0	/РАБОЧАЯ - ЯЧЕЙКА
			СЧЕТЧИКА ЦИКЛА
0576	1000	A, 1000	/НАЧАЛЬНЫЙ АДРЕС
			МАССИВА
0577	7646	B, 7646	/ДОП. КОД ЧИСЛА
			ПОВТОРЕНИЙ
		≡	
A	0576		
ADD	0574		

Нетрудно видеть, что ассемблер заметно облегчает составление программ для «Электроники-100» по сравнению с прямым программированием в машинных кодах, хотя и сохраняет значительную специфику её системы команд, организации памяти и т. п.¹. Вместе с этим имеются неудобства применения ассемблера на различных отечественных машинах. Основное из них состоит в использовании символов и сокращенных наименований команд с исходной английской нотации. Другое неудобство связано с использованием различной кодировки символов на печатающих устройствах разных машин: «Электроника-100» работает прямо с кодами «Консула-254», а «Электроника-100И» и «Саратов» применяют более стандартизованные коды, под которые и разработан транслятор ассемблера. Таблица соответствия этих кодов приведена ниже (табл. 6.1). Для работы с ассемблером на «Электронике-100» используют устройство электронной перекодировки «Янтарь», а также программный вариант перекодировки перфолист, что позволяет за счёт введения двух дополнительных проходов работать с ассемблером и на базовых моделях «Электроники-100»².

При использовании *макроассемблера* для «Электроники-100» программист получает дополнительную возможность вводить макрооператоры. Для этого он описывает некоторую последовательность действий над формальными переменными и (предварительно) присваивает такому макрооператору символическое наименование. Если затем в любом месте программы транслятор встречает обращение к этому макрооператору, то вся описанная ранее последовательность действий будет воспроизведена с указанными в обращении (теперь уже не формальными) параметрами. Кроме того, в макроассемблере, по сравнению с ассемблером, имеется дополнительная возможность прямого указания числовых констант, используемых в командах сложения и логического умножения — открывающая скобка перед числом служит псевдокомандой (транслятору) и транслятор сам отводит под это число ячейку на текущей странице и организует обращение к ней. Макроассемблер освобождает программиста и от забот, связанных со страничной организацией памяти в «Электронике-100», беря на себя распределение памяти и переход с одной текущей страницы на другую.

Как и ассемблер, макроассемблер имеет двухпроходный транслятор.

Языком достаточно высокого уровня, удобным для инженерных расчётов на «Электронике-100», является «ФОКАЛ» [5, 6] (сокращенное наименование от «Formulas calculation» — расчёт по формулам). Этот язык допускает использование во входном описании не только символических обозначений простых идентификаторов, но и переменных с индексами. Имеется возможность прямого указания арифметических действий сложения, вычитания, умножения, деления, возведения в степень; числа могут быть представлены в режиме с плавающей запятой. Допускается прямое обращение к библиотечным функциям (включая вычисление синуса, косинуса, логарифма, экспоненты, тангенса, арктангенса) и разработанным библиотечным подпрограммам.

Таблица 6.1

ASCII		CONSUL-		Примечание	ASCII		CONSUL-		Примечание
Код	Символ	Код	Символ		код	Символ	Код	Символ	
211	ГТ	02	Т	Табуляция	30	G	34	G	
		5			7		3		
212	ПГ	22	ПГ	Перевод	31	H	25	H	
		1			0		5		
215	РК	02	РК	строки	31	I	15	I	
				Возврат	31	I	35	I	
				капетки	31	K	05	K	
240	Про- бел	22	Про- пуск		31	L	15	L	
241	!	17	!		31	M	05	M	
242	”	37	”	Кавычка	31	N	15	N	
243	≠	35	≡	Номер	31	O	25	O	
244	\$	37	∅	Знак ле- нейной	32	P	26	P	
				елинии	32	Q	17	Q	
245	%	14	%	Проценты	32	R	16	R	
246	&	15	т	Коммерче- ское "и"	32	S	36	S	
247	,	17	,	Апостроф	32	T	06	T	
250	(	10	(		32	U	16	U	
251	)	10	)		32	V	34	V	
252	*	03	*	Звездочка	32	W	14	W	
253	+	21	+	Плюс	33	X	26	X	
254	.	30	.	Запятая	33	Y	26	Y	
255	-	01	-	Минус	33	Z	14	Z	
256	•	30	•	Точка	33	[	11	[	Левая квадратн скобка
		2			33	↓	36	V	Стрелка
257	/	21	/	Дробная черта	4		2		вниз
		4			33	]	10	]	Правая
260	0	20	0		5		0		
261	1	00	1						квадратн скобка
262	2	00	2		33	↑	30	↑	Стрелка

¹ Следует отметить, что компактность разрядной сетки «Электроники-100» и структура её системы команд скрадывают большую часть недостатков программирования в машинных кодах по сравнению с ассемблером, весьма заметных, в частности, для 16-разрядных машин и машин с байтовой (8-разрядной) организацией данных и команд.

² Соответствующие программы разработаны Р. Н. Тимониной. Возможно также включение программного перекодировщика непосредственно в транслятор (внешне — за счёт некоторого усечения таблицы переменных символов) — такой вариант проработан А. П. Петренко.

263	3	20	3						вверх Полченки- ние
264	4	00	4			33 ←	13 —		
265	5	20	5			34 Ю	07 Ю		
266	6	20	6			34 Б	24 Б		
267	7	00	7			34 П	26 П		
270	8	01	8			34 Л	24 Л		
271	9	21	9			34 Ф	06 Ф		
272	:	34	:			34 Г	04 Г		
273	;	31	;			35 И	25 И		
		0			1		0		
274	<	16	<	Меньше		35 Й	05 Й		
		5			2		1		
275	=	31	=	Равно		35 Л	25 Л		
276	>	11	>	Больше		36 П	05 П		
		35				36 Я	07 Я		
277	?		⊃	Вопроси-	1		6		
		7		тельный знак		36 Ж	04 Ж		
						37 Б	07 Б		
300	@	11	-			37 Ы	27 Ы		
301	A	04	A			37 З	24 З		
302	B	24	B			37 III	07 III		
303	C	06	C			37 Э	27 Э		
304	D	14	D			37 IIИ	27 IIИ		
305	E	04	E			37 Ч	06 Ч		
306	F	36	F			37 ЗБ			Забой

ASCII		CONSUL-		Примечание	ASCII		CONSUL-25 4		Примечание
Ко	Симво	Код	Симво		Код	Симво	Код	Символ	
		23	—	Надчеркивание			307	X	Знак умножения
		0							
		22	BP	Верхний регистр			331		Вертикальная черта
		4							
		02	HP	Нижний регистр			352	≤	Меньше или равно
		6							
	'	22	KL	Красная лента			274	≥	Больше или равно
		2							
		02	CL	Черная лента					
		3							

(вычисления целой части числа, величины модуля числа, квадратного корня, псевдослучайных чисел и др.).

Транслятор с ФОКАЛА является однопроходным и работает в диалоговом режиме, запрашивая у оператора очередную строку исходного текста лишь после обработки предыдущей. В целях экономии оперативной памяти в самом начале работы транслятор запрашивает, нужны ли пользователю стандартные функции (синус и др.), и если получает ответ «No» («нет»), то стирает соответствующие подпрограммы, увеличивая поле, которое отводится под транслируемую программу. Нумерация строк исходного описания облегчает процесс редактирования, исправления и дополнения программы при отладках.

Таблица 6.2

Арсинус		$\text{FATN}(A/\text{FSQT}(1-A^2))$
Аркосинус		$\text{FATN}(\text{FSQT}(1-A^2))$
Арктангенс		$\text{FATN}(A)$
Арксеканс		$\text{FATN}(\text{FSQT}(A^2-1))$
Арккосеканс		$\text{FATN}(1/\text{FSQT}(A^2-1))$
Аркотангенс		$\text{FATN}(1/A)$
Синус	гиперболический	$(\text{FEXP}(A) - \text{FEXP}(-A))/2$
Косинус	гиперболический	$(\text{FEXP}(A) + \text{FEXP}(-A))/2$
Тангенс	гиперболический	$(\text{FEXP}(A) - \text{FEXP}(-A))/(\text{FEXP}(A) + \text{FEXP}(-A))$
Секанс	гиперболический	$2/(\text{FEXP}(A) + \text{FEXP}(-A))$
Косеканс	гиперболический	$2/(\text{FEXP}(A) - \text{FEXP}(-A))$
Котангенс	гиперболический	$(\text{FEXP}(A) + \text{FEXP}(-A))/(\text{FEXP}(A) - \text{FEXP}(-A))$

на языке символического кодирования, который обеспечивает возможность перемещения отдельных фрагментов программы в МОЗУ и стыковки их с другими, ранее написанными участками (за счёт использования соответствующей программы «перемещающего загрузчика», осуществляющего привязку очередного фрагмента к текущему свободному месту в памяти).

Широкие возможности для редактирования открывает специальная *программа редактирования* символических лент. С помощью этой программы реализуются функции ввода текста с перфоленты или с клавиатуры «Консула», вывод текста (целиком или отдельными строками) на печать или перфорацию, стирание одной или нескольких строк, добавление в исходное описание после указанной строки новых фрагментов текста, поиск в указанной строке нужного символа и др. (нумерация строк для удобства общения осуществляется в десятичной системе счисления). Редактор является однопроходной программой и работает в режиме диалога.

Самостоятельную роль может играть и язык символического кодирования в перемещаемом формате перфолент, который служит языком промежуточного описания задачи при работе с ФОРТРАНОм на 8К.

Часть вторая

ПРИМЕНЕНИЕ «ЭЛЕКТРОНИКИ-100»

Малые ЭВМ являются по существу универсальными машинами, возможности которых расширены за счёт наличия периферийных устройств и каналов прерывания. Поэтому диапазон применения малых машин весьма велик¹; здесь можно выделить следующие характерные области:

- а) работа в качестве универсального цифрового вычислителя;
- б) роль периферийного процессора в сложных вычислительных и управляющих системах;
- в) применение в задачах АСУ ТП.

Насколько типична роль малой машины в каждой из этих областей?

Благодаря усилиям программистов и использованию языков программирования малые машины могут успешно выполнять значительную часть той же вычислительной работы, что и большие универсальные ЦВМ, причем одноадресность и необходимость применения ряда специальных программных приемов часто восполняются высоким быстродействием малых машин². Вместе с этим следует отчетливо представлять, что большие машины обладают большими возможностями по объёму памяти, способам представления информации, накопленным библиотекам программ и языков. Расчётные задачи на больших машинах

¹ В последние 10 лет интерес к малым машинам постоянно растёт (см., например, обзоры [8, 9], а также [10]).

² При этом относительно низкая стоимость малой машины иногда побуждает руководителей мелких подразделений приобрести такую «собственную» машину для использования в качестве вычислительной установки внутри своего отдела (лаборатории), часто не имеющего даже достаточно квалифицированного персонала для обслуживания сложной техники.

осуществляются проще и точнее, могут иметь значительно большие размерности и значительное разнообразие входных описаний. Удобной и недорогой формой периодического использования больших машин является аренда машинного времени, а также работа в режиме разделения времени. Поэтому область основного применения малых машин лежит обычно в стороне от решения чисто вычислительных задач — высокое быстродействие, наличие режимов прерывания и системы периферийных устройств делает малые машины незаменимыми в роли периферийных процессоров и в качестве основы (базы) построения автоматизированных систем управления производственными процессами.

Малая машина, работающая в режиме периферийного процессора (или терминальной ЭВМ), освобождает центральный вычислитель от вспомогательных операций преобразования и редактирования вводимой (выводимой) информации, разгружает его от мелких функций по организации и контролю собственно процедур ввода—вывода, снижает общее число прерываний центрального вычислителя, обеспечивая в то же время возможность параллельной работы с большим числом собственно терминальных устройств типа пультов, пишущих машинок, табло и т. п. Эффективность применения малой машины в этих случаях зависит от общей конфигурации и назначения всей сложной системы, в которой она работает, и наличия соответствующего программного обеспечения.

Наконец, применение ЭВМ в автоматизированных системах управления призвано существенно повысить эффективность управляемых систем. При этом автоматизация технологических процессов позволяет увеличить отдачу самых массовых низовых производственных процессов, где каждый процент дополнительного выпуска продукции дает огромный интегральный эффект. Именно здесь малые ЭВМ, которые за счет развитой периферии легко подключаются к процессу, могут, по-видимому, дать наибольший эффект. Для его достижения, однако, требуется решить ещё целый ряд самостоятельных задач. Их рассмотрению и реализации на «Электронике-100» посвящается материал второй части книги.

7. ЗАДАЧИ АСУ ТП

Прежде чем приступить к изложению особенностей применения «Электроники-100» в системах управления, целесообразно осветить круг задач, относимых обычно к АСУ ТП, и подходы, используемые при их решении.

Автоматизированные системы управления в своем развитии прошли через ряд характерных этапов. Соответственно изменилась и роль машины в системе, а также требования, предъявляемые к машине, к её алгоритмическому и программному обеспечению.

Простейшие системы используют ЭВМ в качестве программатора — машина работает в цикле выдачи стандартных управляющих команд через заданные интервалы времени либо по сигналам извне, с помощью которых в машину сообщается об отработке предыдущего воздействия. В таких системах, воспроизводящих электронным путем принцип работы кулачкового механизма, применение машины эффективно лишь при быстро протекающих процессах либо для обслуживания многих, асинхронно работающих идентичных установок. Однако именно отсюда берут своё начало автоматизированные терминальные системы на базе малых машин, которые саму программу либо её настроечные параметры получают от некоторого центрального процессора и лишь реализуют полученную информацию некоторым стандартным образом. Основной эффект достигается тогда за счёт устранения промежуточных форм выдачи и хранения информации; отпадает необходимость идентификации и контроля промежуточных массивов, не требуется заботиться об удобстве их подготовки и последующего ввода в исполнительные автоматы и т. п. Роль малой машины, теперь уже не просто носителя жесткой программы, но и активного посредника между центральным вычислителем и исполнительным механизмом, заметно возрастает. Эффективно используются не только периферийные устройства малой машины, но и её быстродействие и логические функции, особенно при реализации связи с центральной машиной и настройках собственной программы.

Основная роль ЭВМ в таких системах — решение логических задач.

Другое характерное применение машины в задачах АСУ ТП — это работа в контуре прямого цифрового регулирования. Измеряя фактическое текущее значение некоторого параметра на действующем объекте и сравнивая его с заданным значением, машина рассчитывает и выдает управляющее воздействие с целью минимизации величины рассогласования. Заметную роль здесь начинают играть периферийные устройства машины, выполняющие в данном случае все функции связи с объектом; значительная нагрузка приходится на вычислительные функции машины. Хотя с большинством задач регулирования успешно справляются достаточно простые специализированные аналоговые устройства, применение машины может оказаться эффективным при обслуживании значительного количества контуров регулирования, а также в тех случаях, когда регулируемый параметр не поддается прямому измерению, но его значение может быть оценено косвенно, на основе расчета по значениям измеряемых параметров.

При построении систем регулирования отчетливо проявляется одна особенность, имеющая принципиальное значение и для всех других применений ЭВМ в задачах АСУ и АСУ ТП — необходимость учёта статических и динамических свойств управляемого объекта¹.

Развитые методы экспериментального изучения характеристик объекта либо уточнения параметров его теоретических моделей часто требуют активных возмущений рабочих режимов производственного процесса, что сдерживает их широкое применение на практике. Более перспективными здесь оказываются методы изучения характеристик объекта по данным его нормального функционирования. (Некоторые аспекты этой задачи, связанные с использованием ЭВМ, рассматриваются в гл. 10). Знание характеристик объекта позволяет по известным методикам рассчитать оптимальные настройки регулятора, которые затем (в автоматизированных машинных системах) и закладываются в управляющую ЭВМ. По аналогии с обычными регуляторами машине поручается осуществлять пропорционально-интегрально-дифференциальное регулирование (см., например, [11]):

¹ Каждое отдельное возмущение со стороны любого из входных параметров обычно не тотчас приводит к изменению выходного параметра, а инициирует развитие некоторого переходного процесса. В зависимости от свойств объекта этот переходный процесс может иметь существенно различный вид и временные параметры. Вид кривой переходного процесса характеризует динамические свойства объекта, а амплитудные параметры процесса — статический коэффициент передачи по данному каналу в окрестности рабочей точки.

$$u(t) = k_1 e(t) + k_2 \int_{t-T_H}^t e(\tau) d\tau + k_3 \frac{de(t)}{dt}, \quad (7.1)$$

где $u(t)$ — управляющее воздействие; $e(t)$ — ошибка относительно заданного значения, k_1, k_2, k_3, T_H — коэффициенты и параметр регулирования.

При машинной реализации такого процесса регулирования, а также и при решении других, более сложных задач возникают следующие характерные вопросы:

- а) как сказывается неточное знание характеристик объекта на качестве всего процесса?
- б) с какой дискретностью во времени должна работать машина, чтобы поддерживать качество всего непрерывного процесса на достаточно высоком уровне?
- в) является ли вообще традиционный закон регулирования (7.1) наилучшим?

Остановимся коротко на каждом из этих вопросов.

Для достаточно сложных объектов значение выходного параметра всегда определяется совместным влиянием всех входных воздействий, а также неучтенных факторов и помех. Поэтому и экспериментально полученные характеристики объекта неизбежно содержат некоторые ошибочные компоненты, в связи с чем возникает проблема их количественной оценки, а также оценки влияния этих неточностей на качество регулирования в целом.

Другой вопрос связан с использованием дискретно работающей ЦВМ для управления непрерывным процессом. Качественная сторона вопроса состоит в том, что слишком редкие управляющие воздействия не могут поддерживать достаточно высокого качества регулирования¹, а слишком частый опрос датчиков и реализация управляющих воздействий могут неоправданно перегружать машину (кроме того, у каждой ЦВМ есть реальный предел быстродействия). Количественная оценка должна определять, каким образом изменяется качество регулирования по мере изменения частоты отсчетов в системе. При переходе от непрерывной формы (7.1) к её дискретному аналогу выбранная частота отсчетов определяет параметры основных временных циклов программы и масштабные коэффициенты для k_2, k_3, T_H .

Наконец, регулирование вида (7.1) воспроизводит лишь ту форму управляющего воздействия, которую относительно легко могли в своё время осуществлять аналоговые схемы. Нетрудно заметить, что при переходе к дискретной технике алгоритмически просто осуществляется не только равномерное усреднение значений ошибки на некотором интервале с дополнительным учётом её двух последних отсчетов (к чему сводится смысл дискретного аналога (7.1)), но и любое другое «взвешивание» ряда наблюдаемых значений ошибки. Естественно возникает вопрос, не будет ли какая-нибудь другая форма определения $u(t)$ эффективнее, чем (7.1), и если да, то насколько? Кроме того, нет ли другой, более простой для машинной реализации, но и достаточно эффективной формы расчёта $u(t)$?

Хотя названные вопросы имеют отчетливую алгоритмическую окраску, они непосредственно влияют не только на качество математического обеспечения в целом, но и на разработку программного обеспечения. Поэтому без детальной теоретической разработки этих вопросов выбор многих программных параметров осуществляется интуитивно.

Следует отметить, что в контурной системе прямого цифрового регулирования остро стоит вопрос о надёжности длительной работы, особенно в круглосуточном режиме. Специальные меры аппаратного и программного обеспечения надёжности значительно повышают стоимость системы. Поэтому вопрос о применении отдельной УЦВМ в контуре прямого регулирования вместо набора более простых аналоговых регуляторов все ещё остается дискуссионным. Наряду с этим, важной особенностью систем регулирования на базе ЦВМ является использование текущей информации о состоянии объекта на основе непосредственного измерения доступных сигналов².

Наличие информационного режима является весьма характерной чертой АСУ ТП. Для ранних систем этот режим с регулярным опросом датчиков и распечаткой результатов является практически монопольным — считалось, что самое важное состоит в сборе информации и предоставлении её человеку-оператору в удобной форме. Однако очень скоро стало очевидно, что подобные системы буквально обрушивают на оператора такой поток информации, с которым человек не в состоянии справиться. Появились идеи систем-советчиков, ведущих регулярный опрос датчиков и выдачу рекомендаций, направленных на улучшение (оптимизацию) процесса³. Основное направление дальнейших разработок и состоит в развитии методов получения таких рекомендаций.

Как и в любых других задачах оптимизации, одним из исходных вопросов является определение (назначение) функции цели: что же именно требуется улучшить в процессе? По-видимому, на этот вопрос должны давать ответ руководители предприятия (цеха, участка)⁴.

Очередной этап состоит в создании некоторой системы зависимостей, связывающих значения основных режимов параметров с выбранной функцией цели. И наконец, с использованием полученных зависимостей могут быть тем или иным способом рассчитаны рекомендации на улучшение процесса за счёт нанесения управляющих воздействий (изменения значений режимных параметров, доступных управлению). Именно на этих последних этапах и сосредоточены основные трудности рассматриваемого вида систем управления.

Для реальных объектов исходные зависимости на уровне физических переменных в ряде случаев могут быть выражены лишь в виде нелинейных дифференциальных уравнений, а сама функция цели может иметь достаточно сложный вид. Более того,

¹ Эффект регулирования при этом может даже стать отрицательным, так как большую часть отклонений от заданного режима машина перестает замечать, да ещё добавляет к процессу свою реакцию на отдельные, давно прошедшие выбросы.

² Применение машины открывает широкие возможности для реализации гибких алгоритмов контроля сложных параметров, расчёта текущих значений обобщенных технико-экономических показателей, различных коррекций и фильтрации помех в измеряемых параметрах и т. п. Однако, и здесь, и в более сложных задачах автоматизированных систем управления попрежнему сохраняется главенствующая роль текущей информации и необходимость учёта динамических связей внутри информационных потоков (некоторые аспекты этой проблемы рассматриваются в гл. 9)

³ Информационная часть подобных систем обычно обеспечивает также регулярную распечатку сводных показателей и вызов любого интересующего параметра с пульта на индикацию по запросу оператора.

⁴ Критерий оптимизации может иметь самую различную природу — наряду с чисто техническими требованиями (типа максимизации производительности установки) могут выдвигаться и требования технико-экономического характера (увеличить выход годных изделий, снизить технологическую составляющую себестоимости и т. п.).

часть зависимостей на практике оказывается неполной или вообще не описываемой детерминистически (как это, например, имеет место при критерии в виде выхода годных). Наряду с этим на каждом конкретном действующем объекте так или иначе могут быть измерены необходимые сигналы, отражающие его текущее состояние. Таким образом, получение необходимых конечных зависимостей, очевидно, связано с использованием текущей информации от объекта и, следовательно, с построением обоснованных алгоритмов её обработки, позволяющих изучать интересующие зависимости по данным его нормального функционирования. При этом, как уже упоминалось, необходимым оказывается учёт динамических связей, случайного характера возмущений и помех, конечного времени наблюдения, коррелированности сигналов во времени и других факторов. Построение достаточно универсальных алгоритмов и, главное, системы оценок точности результатов, получаемых в таких условиях¹, является большой самостоятельной проблемой.

Остается решить вопрос о способе применения найденных зависимостей для оптимизации процесса. Даже в том случае, когда полученные зависимости описывают значительную область возможных изменений режимных параметров, точное переключение процесса на оптимальный режим за счёт единственного управляющего воздействия является весьма маловероятным из-за погрешностей исходных зависимостей. Наряду с этим в режиме нормального функционирования флуктуации режимных параметров обычно достаточно малы по сравнению со всем возможным диапазоном их изменения, поэтому и получаемое описание характеристик носит сугубо локальный характер. Рациональным в таких условиях является, по-видимому, применение пошаговой процедуры оптимизации² с уточнением характеристик процесса в каждой рабочей точке. Такая процедура является достаточно универсальной, так как не только совмещает изучение процесса с управлением (выводом в оптимальную точку), но и позволяет осуществлять адаптацию к достаточно медленным изменениям условий функционирования объекта вследствие изменений условий окружающей среды либо характеристик самого управляемого объекта.

Применение ЭВМ в такой автоматизированной системе, очевидно, и является наиболее эффективным. Именно здесь в полной мере могут использоваться не только периферийные устройства машины, но и её вычислительные «способности». Важно и то обстоятельство, что на этом пути принципиально просматриваются все этапы последовательного наращивания функций автоматизированной системы (или построения самонастраивающейся системы): подключение машины к производственному процессу, изучение характеристик сигналов, выбор параметров информационного режима, накопление данных нормального функционирования объекта, определение (динамических и статических) характеристик процесса, расчёт рекомендаций по нанесению управляющих воздействий, переход на новый режим с использованием данных переходного процесса для уточнения характеристик, новое накопление данных нормального функционирования, очередная итерация режима и т. д., с получением в качестве полезного побочного эффекта некоторого описания характеристик объекта вдоль траектории движения³. Дальнейшее развитие системы, по-видимому, может привести к оценке наличия дрейфа (и, следовательно, необходимости постоянной последующей самонастройки), а при отсутствии дрейфа — к приближению и стабилизации процесса в области ближайшего локального экстремума (после чего дальнейшее улучшение показателей процесса возможно лишь за счёт оптимизации параметров регуляторов, роль которых на первых порах и работу по оптимизации параметров может выполнять сама машина). Естественно, каждый из названных этапов требует хорошо разработанных и эффективных алгоритмов, особенно при использовании малой машины с её ограниченными ресурсами. Заметно возрастает роль оперативности всей системы в целом на каждом из этапов, так как любая текущая информация стареет и вследствие необратимости времени и случайного характера возмущений прошлые рекомендации спустя некоторое время могут оказаться не только бесполезными, но даже вредными. Поэтому вопрос рациональной организации в машине всех процессов получения и обработки информации в реальном времени является одним из первоочередных при построении общего программного обеспечения любых АСУ ТП. Решению этого вопроса посвящена следующая глава; в гл. 9 и 10 рассматриваются некоторые вопросы алгоритмического и программного обеспечения основных этапов разработки и введения в действие автоматизированных систем на базе малых машин.

8. ПРОГРАММА-ДИСПЕТЧЕР

Работа машины в системе управления имеет значительную специфику не только в характере решаемых задач, но и в организации их решения. Здесь соединяются такие важные требования к программному обеспечению современных ЭВМ, как возможность работы в реальном масштабе времени и практически одновременного доступа к процессору многих пользователей. Поэтому значительная часть полученных при этом решений имеет большую перспективу применения и в других системах.

Рассмотрение типичных условий функционирования минимашин в системе управления позволяет с достаточной конкретностью выявить основные требования к диспетчеру ЭВМ, работающей в реальном времени, определить состав минимально необходимых блоков и наметить возможные пути целесообразного расширения диспетчерских функций.

8.1. ОСОБЕННОСТИ ПОСТРОЕНИЯ ДИСПЕТЧЕРА ДЛЯ МИНИМАЛЬНОГО КОМПЛЕКТА ЭВМ

Уже в относительно простых системах управления имеет место следующая ситуация. На действующем объекте измеряется с помощью датчиков и преобразователей некоторое количество параметров. Соответствующие электрические сигналы периодически или по синхронизирующим сигналам извне должны приниматься в машину и подвергаться обработке, в результате которой обобщенная информация выдается на информационное табло (или печать) и вырабатываются управляющие сигналы, выдаваемые в виде рекомендаций оператору или в виде командных кодов исполнительным механизмам. Кроме того, обычно должна быть предусмотрена возможность введения в систему запросов или корректирующих указаний оператора, а

¹ В частности, нарушение исходных предпосылок широко распространенного регрессионного анализа делает невозможным его прямое применение для изучения характеристик динамических систем по данным нормального функционирования (см. также гл. 10).

² Последовательный анализ и ряд важных теоретических положений, обосновывающие применение пошаговых алгоритмов оптимизации в автоматических системах, приводятся в [12].

³ Следует отметить, что подобные этапы могут оказаться характерными и для многих задач автоматизированного научного эксперимента

также различных аварийных сигналов. Особенностью большинства внешних запросов является произвольный (случайный) момент возникновения и необходимость практически немедленного их приема и начала исполнения.

Совершенно очевидно, что для построения системы управления в таких условиях, помимо введения в состав системы специальных внешних устройств (типа таймера, коммутаторов, блоков управления информационным табло и т. п.), необходимо обеспечить выполнение дополнительных принципиальных требований и по отношению к программному обеспечению¹.

Сама организация основных процессов в машине должна предусмотреть возможность оперативного получения заявок извне и доступа к любой части общей программы, а также совмещения во времени относительно медленных процедур ввода—вывода с использованием процессора для решения новых (очередных) задач. «Электроника-100» обладает необходимыми аппаратными предпосылками для этого — в виде системы прерывания и набора «флагов» (триггеров готовности) внешних устройств; вся остальная нагрузка, как и всегда при реализации сколь-нибудь сложных функций в системах на базе мини-ЭВМ, приходится на программное обеспечение — возникает необходимость в программе-диспетчере. Жесткие рамки МОЗУ (особенно для машин с минимальным комплектом оборудования) заставляют тщательно подбирать состав минимально необходимых блоков. По мере развития системы (за счёт введения дополнительных кубов памяти, магнитных лент т. п.) диспетчерские функции могут расширяться, может измениться и сам способ их реализации.

Возвращаясь к указанным условиям работы системы, можно назвать в качестве крупных блоков любого варианта диспетчера блок обслуживания прерываний, службу времени, службу ввода и службу вывода; с учётом того, что возникающие задачи могут перекрываться во времени, диспетчеру необходим ещё блок установления очередности. Функции каждого из блоков и способ их взаимодействия конкретизируются после определения требований со стороны диспетчера к задачам основного программного обеспечения и возможных запросов к диспетчеру.

Так как каждый из участков общей программы предназначен для выполнения определенной целевой функции общей системы, а все функции управления решением отдельных задач сосредоточены у диспетчера, в распоряжение последнего должны быть предоставлены характеристики всех отдельных задач. К числу требуемых характеристик, прежде всего, относятся стартовый адрес задачи и её приоритет. Необходимость последней характеристики с очевидностью следует из того обстоятельства, что на её основе строго и просто² могут разрешаться конфликтные ситуации при одновременном возникновении заявок. Далее, запрос на решение той или иной задачи должен быть доведен до диспетчера, например, по соответствующему прерыванию извне или по запросу одной из служб самого диспетчера. Наконец, для достижения оперативности в реакции на запросы все отдельные задачи должны решаться с разрешенным прерыванием и каждая задача должна заканчиваться передачей управления диспетчеру (в связи с последним обстоятельством диспетчер, очевидно, должен содержать ещё блок окончания задач).

С точки зрения разработчика программ, для решения отдельных задач работа с диспетчером налагает (пока) лишь два ограничения: нельзя внутри программы запрещать прерывание (по крайней мере, на длительное время) и по окончании задачи надо вернуть управление диспетчеру. Остальные требования больше важны как принципы организации самого диспетчера (и вместе с тем позволяют отделить вопросы организации системы в целом от вопросов разработки отдельных программ). В частности, блок обслуживания прерываний должен иметь таблицу с адресами и приоритетами тех задач, которые инициируются по запросу извне, а также располагать средством установления соответствия номера внешнего устройства, вызвавшего прерывание, требуемой задаче. Блок обслуживания прерываний и все службы диспетчера должны иметь возможность обращения со своими заявками к блоку установления очередности; данные этого же блока, очевидно, должны использоваться и блоком окончания задач.

К числу запросов, приходящих в систему из внешнего мира, относятся и запросы оператора. Обычно связь оператора с системой осуществляется через телетайп или электрифицированную пишущую машинку; в обоих случаях запрос посимвольно (в сопровождении сигнала прерывания) поступает в машину через канал ввода. Служба ввода диспетчера, очевидно, должна обеспечивать формирование и расшифровку запроса (на основе принятого в системе входного языка) и представление запроса в виде соответствующих характеристик (начального адреса и приоритета) блоку установления очередности. Особенностью режима работы с диспетчером является возможность приостановки работы любой программы немедленно по готовности устройства ввода — для выполнения относительно малого числа операций по приему кода очередного символа, проведения расшифровки и проверок запроса с последующим возвратом управления прерванной задаче. Поскольку скорость работы телетайпа такова, что между соседними состояниями готовности устройства ввода процессор успевает выполнить десятки тысяч операций, такой режим за счёт удлинения общего времени обслуживания на доли процента позволяет практически безболезненно совместить вычисления с процессом ввода.

Аналогично совмещаются расчётные операции с процессом вывода информации на медленно действующий телетайп или пишущую машинку: при готовности внешнего устройства (по сигналу прерывания от него) диспетчер приостанавливает решение очередной задачи и передает управление в службу вывода для выполнения небольшого числа операций по формированию и передаче в буфер вывода следующего выводимого кода; после выполнения команды, инициирующей начало цикла работы электромеханического устройства вывода, управление возвращается прерванной задаче. Дополнительной особенностью службы вывода является способ взаимодействия с основными программами системы, обращающимися к диспетчеру за выводом. Так как посимвольный вывод занимает значительное время, в течение которого процессор может обслужить не один десяток задач, служба вывода должна снять с обслуживания (законсервировать) на все время вывода обратившуюся к ней задачу и предусмотреть возможность хранения накапливающихся заявок на вывод; по окончании вывода соответствующая задача должна быть вновь предъявлена блоку установления очередности для продолжения её решения.

Такой же особенностью по отношению к задачам основного программного обеспечения должна обладать и служба времени: задача, обратившаяся к диспетчеру за задержкой, должна быть законсервирована на требуемое время, с последующим восстановлением её в очереди обслуживаемых задач. Кроме того, достаточно общей функцией службы времени является отсчёт реального времени (счёт часов и минут) по импульсам прерывания от таймера.

Наконец, с учётом заметной сложности организации диспетчера фактически необходимым (хотя и редко используемым при

¹ Традиционный способ последовательного решения задач от начала до конца (или использование метода их пакетной обработки) не позволяет оперативно реагировать на заявки, возникающие в случайные моменты времени. При использовании минимшины, имеющей буферное устройство ввода—вывода для хранения единственного машинного слова, дополнительные неудобства в этом случае доставляет малое (по сравнению с процессором) быстродействие внешних устройств.

² А, следовательно, и с максимальной компактностью программы.

нормальной работе) блоком оказывается блок диагностики и индикации отказов диспетчера, обладающий правом экстренного вывода.

Таким образом, на основе приведенных соображений укрупненная структурная схема программы-диспетчера приобретает вид, показанный на рис. 8.1¹.

Функции и организация отдельных блоков начинают существенно зависеть от аппаратного оформления ЭВМ, для которой разрабатывается программа-диспетчер; полнота реализации тех или иных функций зависит также от объема МОЗУ, отводимого для размещения диспетчера. Ниже рассматриваются принципы построения отдельных блоков.

8.2. ОБНАРУЖЕНИЕ ИСТОЧНИКА ПРЕРЫВАНИЯ

Система прерывания «Электроники -100» и её работа подробно рассмотрены в гл. 3, 4. В машине есть общая шина прерывания, на которую имеют выход триггеры готовности всех внешних устройств. Срабатывание любого из них приводит к прерыванию (если оно ранее по программе было разрешено), в результате которого после завершения очередной команды происходит автоматическое запоминание (в ячейке 0000₈ 0-го куба МОЗУ) места прерывания (адреса продолжения) выполнявшейся программы. Для машин с расширенной памятью в специальный регистр запоминания полей списывается содержимое регистров поля данных и поля команд, после чего содержимое этих регистров устанавливается равным нулю. Управление всегда передается в ячейку 0001₈ 0-го куба МОЗУ, где и должно располагаться начало программы обслуживания прерывания.

Для обеспечения возможности продолжения прерванной задачи необходимо на время работ диспетчера запомнить и сохранить состояние основных регистров машины. В «Электронике-100» это можно сделать только программным путем. Именно с этих операций и начинается работу программа обслуживания прерываний.

¹ К сожалению, четкая терминология по операционным системам до сих пор не установилась (см. например, замечания в монографии «Супервизоры и операционные системы». Пер. с англ. М., «Мир», 1972). Поэтому специальные термины («супервизор», «координатор», «планировщик» и т. п.) здесь и далее не употребляются.

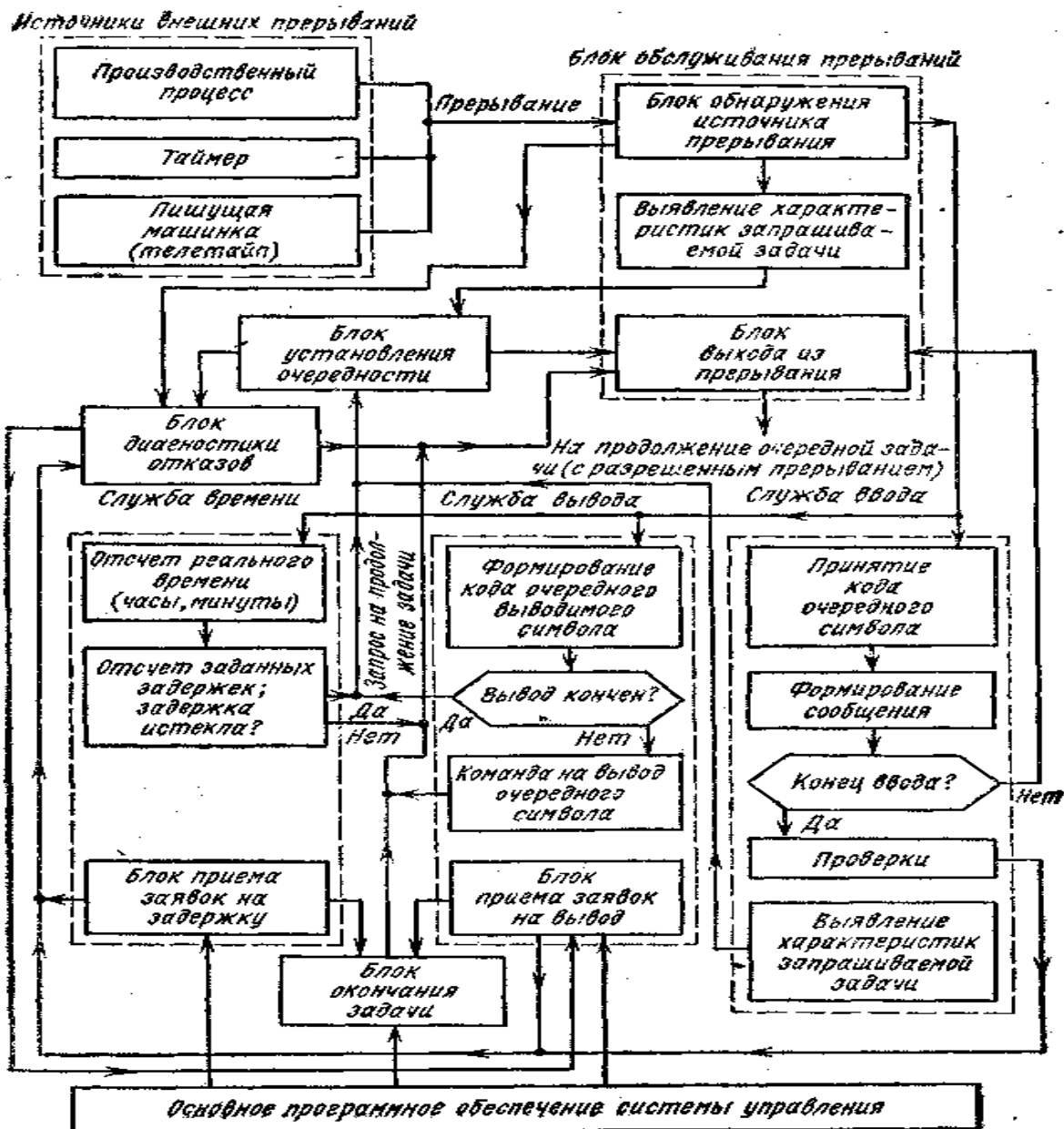


Рис. 8.1. Укрупненная структурная схема программы-диспетчера

Обнаружение источника прерывания в простейшем случае сводится к опросу флагов задействованных внешних устройств со счетом числа пройденных звеньев такой цепочки; выход в следующий блок осуществляется при обнаружении первого же флага, установленного в единичное состояние. Рациональнее представляется последовательность опроса цепочки флагов, начиная с таймера и далее, по мере уменьшения относительной частоты запросов либо снижения приоритетности того или иного внешнего устройства (и связанного с ним потока задач). Когда в ЭВМ используется специальный регистр прерывания, куда заведены флаги всех внешних устройств, либо такой регистр флагов рассматривается как один из источников дискретных сигналов, заводимых в машину через входной коммутатор, структура блока обнаружения прерывания оказывается иной. В цепочке флагов опрашивается лишь групповой флаг регистра прерывания (либо входного коммутатора). Если этот флаг оказался в единичном состоянии, считывается все слово флагов и дальнейшая программа организуется на основе какого-либо цикла поразрядного анализа полученного слова. Очевидно, последовательность опроса флагов в этом случае определяется положением разряда, закрепленного за каждым из них. Иногда таким априорным ранжированием и завершается установление приоритетов для потока задач, однако система оказывается более гибкой, если приоритет задачи рассматривается как её отдельная дополнительная характеристика. В некоторых системах заметные удобства пользователь получает при введении так называемой «маски прерывания» — специального кода, устанавливаемого самим пользователем и позволяющим диспетчеру (путем логического умножения всего слова флагов на этот код) временно исключить реакцию на прерывания от отдельных внешних устройств.

Выявить характеристики запрашиваемой задачи достаточно просто выборкой начального адреса и приоритета¹ из таблиц характеристик исходного набора задач, в соответствии с числом звеньев пройденных при опросе цепочки флагов

Структурная схема программы и описание её работы для общего случая приведены в гл. 4 (см. также рис. 4.9). Блоки 11, 15 и 22 осуществляют обращение к блоку фиксации отказов при обнаружении логических несоответствий (отсутствии хотя бы одного флага, установленного в единичное состояние при обращении в блок обслуживания прерываний², потере единичных разрядов при

¹ При работе с расширенной памятью оказывается необходимым указать также номера кубов МОЗУ, отведенных под программу и под числовые данные (см. ниже).

² В простейшем случае такая ситуация означает наличие помех по каналу прерывания, и в первом варианте программы-диспетчера предусматривался просто выход машины на останов после опроса всех флагов задействованных внешних устройств.

8.3. ОЧЕРЕДИ НА РЕШЕНИЕ ЗАДАЧ

Появление нового запроса на использование процессора приводит к конфликтной ситуации — необходимо вернуть управление прерванной задаче либо начать решать новую. Так как требуется обеспечить достаточно большую глубину прерывания и в обоих случаях исключить потерю задач, в распоряжение блока установления очередности приходится предоставлять буферные зоны памяти для хранения необходимых данных о задачах. В двух указанных случаях объём информации, подлежащей хранению, далеко не равнозначен. Если управление возвращается прерванной задаче, то о новой заявке необходимо сохранить лишь её исходные характеристики. Если же обслуживание задачи прекращается в разгар её решения, прерванного в произвольной точке программы, то для последующего продолжения её работы необходимо сохранить состояние всех основных регистров машины в момент прерывания, а также индексных ячеек и возможно, даже рабочих ячеек. Далее, для четкости организации работы самого блока установления очередности рационально в буферных зонах памяти всю необходимую информацию хранить в виде двух очередей: очереди заявок, обслуживание которых не начиналось (очередь № 1), и очереди прерванных, а затем временно снятых с обслуживания задач (очередь № 2). Целесообразно также принять, что после разрешения конфликтной ситуации всегда обслуживается первая задача из очереди № 1.

Данными из очередей на решение задач активно пользуются блок установления очередности и блок окончания задач; сведения об обслуживаемой задаче (первой задаче из очереди № 1) употребляют также в своей работе некоторые службы диспетчера.

Работа блока установления очередности сводится к сравнению приоритетов новой заявки и прерванной задачи с последующим ростом очереди № 1 или № 2. В процессе работы этого блока задача с более высоким приоритетом попадает на первое место в очереди № 1 и её характеристики заносятся в стандартные рабочие ячейки, используемые блоком выхода из прерывания. Структурная схема блока установления очередности приведена на рис. 8.2

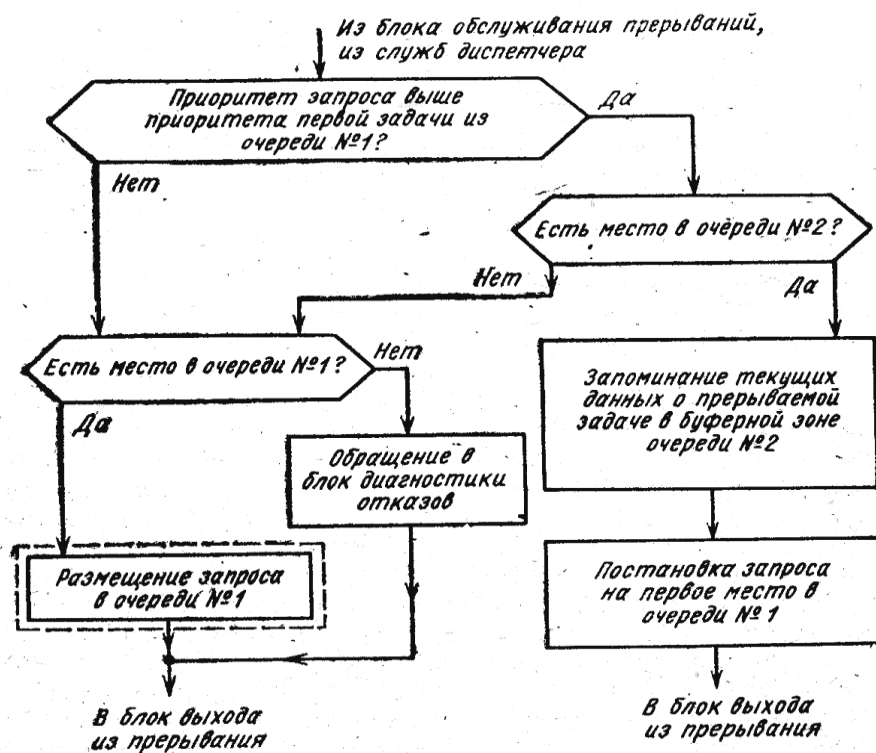


Рис. 8.2. Структурная схема блока установления очередности; структура блока, обведенного штрихом, зависит от избранной стратегии использования приоритетной системы.

Возможны различные варианты размещения запроса в очереди № 1, если его приоритет ниже, чем у прерванной задачи (левая ветвь на рис. 8.2). В простейшем случае запрос помещается в конец очереди № 1. В более общем случае может быть предусмотрено сравнение его приоритета с приоритетами остальных задач очереди № 1 и передвижение всей очереди для размещения запроса в соответствии с его приоритетом. Так как объём буферных зон для очередей № 1 и № 2 ограничен, приходится до размещения новой задачи контролировать наличие свободного места в очередях и принимать дополнительные меры при их переполнении. Так, при переполнении очереди № 2 запрос с более высоким приоритетом, чем у прерванной задачи, не может быть принят к обслуживанию немедленно (некуда снять обслуживаемую задачу). Тогда делается попытка разместить этот запрос в очереди № 1, чтоб все же не потерять его. Если переполняется очередь № 1, то запрос теряется, но за счёт обращения в блок диагностики отказов подготавливается сообщение об этом событии для экстренной выдачи на печать¹.

¹ В более общем случае предпринимается попытка сохранить и запрос с низким приоритетом (если очередь № 1 оказывается переполненной), размещая его в очереди № 2. Здесь, однако, приходится учитывать, что на каждое место в очереди № 1 затрачивается всего две ячейки в памяти. Это в несколько раз меньше, чем требуется на каждое место в очереди № 2, не говоря о дополнительных затратах места в МОЗУ в связи с усложнением программы, оправданным, таким образом, лишь для достаточно универсальных программ.

Перед обращением к блоку установления очередности приоритет запроса и начальный адрес соответствующей задачи засылаются в стандартные рабочие ячейки этого блока. Эта операция может выполняться индивидуально в каждом отдельном случае, либо такая пересылка может выполняться в начале работы самого блока установления очередности, если обращение к нему организовано как обращение к подпрограмме с дополнительно запрашиваемыми данными.

Для облегчения взаимодействия блока установления очередности с блоком обнаружения источника прерывания и блоком выхода из прерывания, особенно в случае одновременного возникновения нескольких запросов, рационально ввести признаки роста очередей № 1 и № 2 либо другие, эквивалентные им, и заменить в таблицах очередей начальный адрес прерывавшейся задачи на адрес места последнего прерывания. Кроме того, в случае одновременного возникновения нескольких запросов приходится дополнительно заботиться о возвращении управления блоку обнаружения источника прерывания (либо службам диспетчера).

Если обращение в блок установления очередности ведет к росту очередей, то работа блока окончания задач состоит в решении обратной задачи. Любое обращение в блок окончания задач диспетчер воспринимает как сигнал об освобождении процессора (и первого места в очереди № 1 в силу принятой структуры взаимодействия рассмотренных выше блоков). На освободившееся место диспетчер подбирает с учетом приоритетов новую задачу из очереди № 1 или № 2, причем благодаря упорядочению потока задач блоком установления очередности достаточно сравнить лишь вторую задачу из очереди № 1 и последнюю из очереди № 2. Соответствующие характеристики задачи заносятся в стандартные ячейки блока выхода из прерывания, которому и передается затем управление. Если к обслуживанию принимается задача из очереди № 2, блок окончания задач сам восстанавливает содержимое индексных ячеек, а более общие функции восстановления $|HP|$, $|DP|$ и другие и в этом случае выполняются блоком выхода из прерывания.

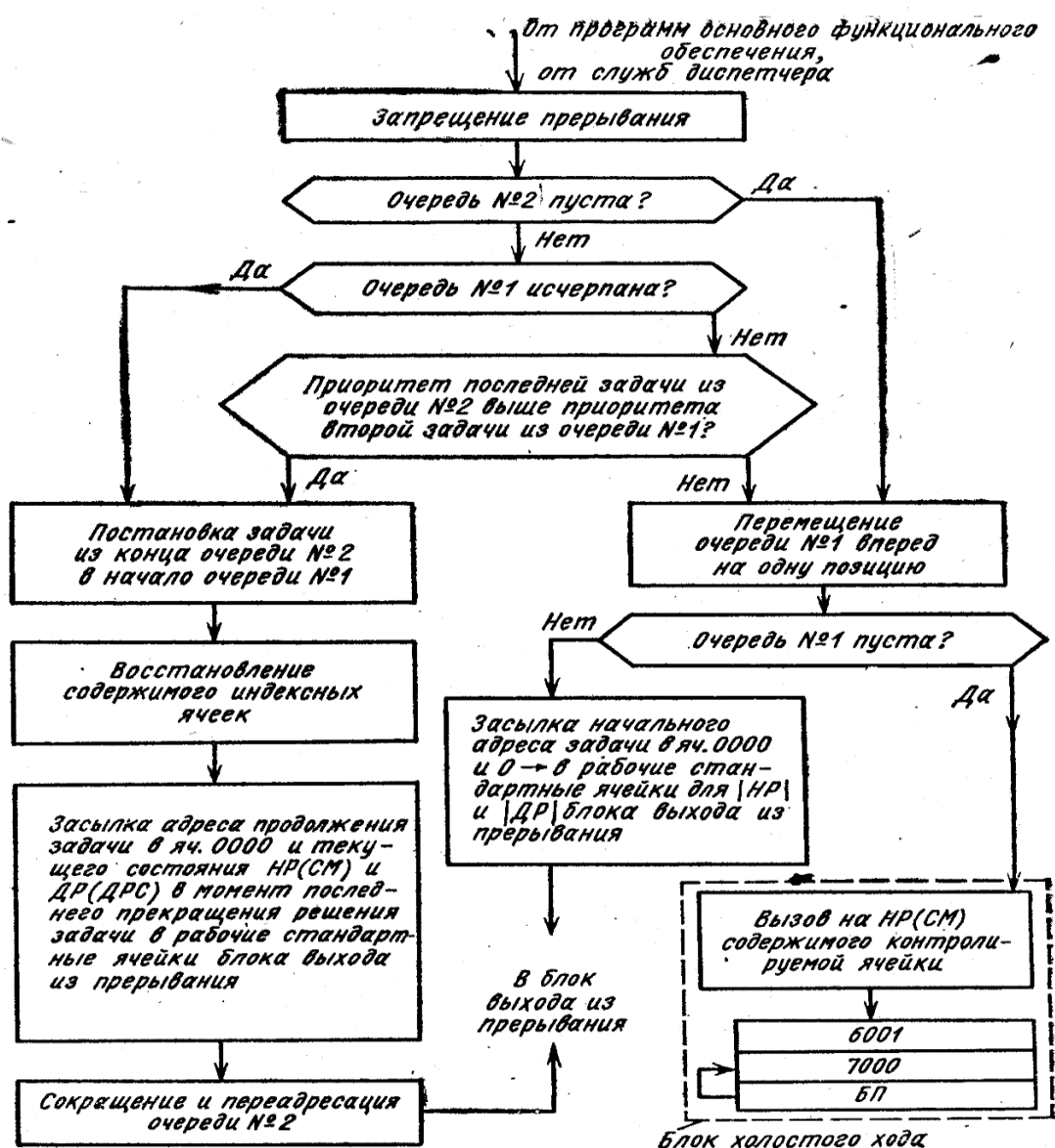


Рис. 8.3. Структурная схема блока окончания задач,

Если все очереди исчерпаны (все задачи решены), управление передается (внутреннему) блоку холостого хода¹, поддерживающему процессор в рабочем состоянии (без выхода на останов) до очередного прерывания, например, путем зацикливания на команде 7000. Так как содержимое НР (СМ) при этом не изменяется, его световую индикацию на передней панели процессора можно с успехом использовать для визуального контроля содержимого отдельной ячейки МОЗУ, что удобно, например, для индикации номера включенного канала (при малой интенсивности работы системы, при отладках и т. п.). Структурная схема блока окончания задач приведена на рис. 8.3.

Блок выхода из прерывания восстанавливает состояние основных регистров машины (НР и ДР), согласно содержимому стандартных рабочих ячеек, обнуляет признаки роста очередей, затем разрешает прерывание и передает управление первой задаче из очереди № 1. Общие принципы организации и программная реализация такого блока для случая одиночного прерывания рассмотрены в гл. 3,4. Особенностью работы этого блока в мультипрограммном режиме является использование его стандартных рабочих ячеек не только блоком обнаружения источника прерывания, но и блоками установления очередности, окончания задач, а также службами диспетчера. Все эти блоки подготавливают блоку выхода из прерывания данные, которые он всегда стандартным образом использует далее. Структурная схема блока выхода из прерывания приведена на рис. 8.4.

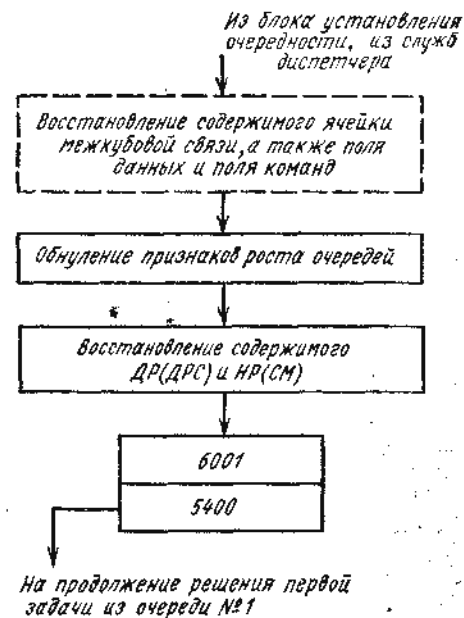


Рис. 8.4. Блок выхода из прерывания. Штрихом указаны блоки, нужные при работе с расширенным объемом МОЗУ

8.4. СЛУЖБЫ ДИСПЕТЧЕРА

Наиболее специфичными блоками служб диспетчера (с точки зрения его организации и функций) являются блоки приема заявок на то или иное обслуживание от программ основного обеспечения системы, а также блоки формирования сообщения и выявления характеристик запрашиваемой задачи в службе ввода.

В целях простоты организации программ основного обеспечения обращение к службам диспетчера рационально выполнить как обращение к подпрограммам. При этом и диспетчер автоматически получает необходимую информацию об адресе продолжения задачи и о месте расположения необходимых дополнительных данных (как в типичном случае обращения к любой многопараметрической подпрограмме). С той же целью диспетчер должен обслуживать заявки так, или почти так, как если бы вся система занималась решением единственной задачи. Однако на время выполнения заявки полезная удельная загрузка процессора этой задачей резко снижается; если сохранить за операциями по организации очередного элементарного шага внешнего устройства наивысший приоритет, то практически на все время выполнения заявки процессор может быть освобожден для решения других задач. Весь механизм освобождения процессора и его загрузки очередными задачами уже содержится в блоке окончания задач, остается лишь позаботиться о восстановлении задачи в общей очереди решаемых задач по окончании исполнения заявки.

Эти обстоятельства предопределяют следующие особенности организации служб диспетчера.

Все программы служб диспетчера (большинство из которых укладывается всего в несколько десятков команд) идут в «привилегированном порядке» (вне очереди и при запрещенном прерывании).

Службы вывода и времени имеют свои буферные зоны для хранения характеристик обратившихся к ним задач. Блоки приема заявок заносят в эти зоны адрес продолжения и приоритет тотчас по обращении задачи из основного программного обеспечения к одной из служб. По окончании обслуживания заявки эти характеристики используются при обращении в блок установления очередности, когда задача вновь готова оперативно использовать процессор.

В буферной зоне службы времени для каждой задачи выделяется еще ячейка для занесения кода (и последующего отсчета) требуемой задержки.

В буферной зоне службы времени заявка помещается на произвольное свободное место; буферная зона службы вывода может быть организована более строго, в виде очереди.

Блок формирования сообщения в службе ввода составляет в целые машинные слова последовательно принимаемые коды отдельных символов; сравнение полученных слов (естественно, на уровне машинных кодов) со словарем возможных задач, запрашиваемых оператором с помощью пишущей машинки, составляет основу работы блока выявления характеристик запрашиваемых задач, которая в остальном не отличается от работы аналогичного блока при обслуживании прерываний.

Структурная схема блоков службы времени приведена на рис. 8.5. В буферной зоне службы времени имеется три таблицы: для адресов продолжения обратившихся задач, для их приоритетов и для отсчета задержек (таблица регистров). Перед обращением к службе времени за задержкой дальнейшего решения на t импульсов таймера задача основного программного обеспечения вызывает восьмеричный код t на НР (на СМ) и передает управление диспетчеру. Образовав от t дополнительный код, служба времени запоминает его в первом свободном регистре, занося адрес возврата и приоритет задачи в соответствующие ячейки двух других таблиц, после чего временно снимает задачу с обслуживания процессором. По прерываниям от таймера в каждый регистр, который не был обнулен, добавляется по единице. Переполнение регистра свидетельствует об истечении задержки соответствующей задачи, характеристики которой тут же предъявляются блоку установления очередности для включения их в очередь на обслуживание процессором. Структурная схема блоков службы вывода показана на рис. 8.6. Для простоты её организации и использования сообщений, выдаваемых на печать, принято, что начавший вывод не прерывается и что очередь на вывод заполняется и продвигается лишь в соответствии со временем поступления

¹ Если позволяет объем памяти, могут проводиться различные тестовые испытания аппаратурного комплекса.

заявок. Право на внеочередной вывод (по окончании обычного вывода, если он уже начался) предоставляется только экстренным сообщениям от блока фиксации отказов.

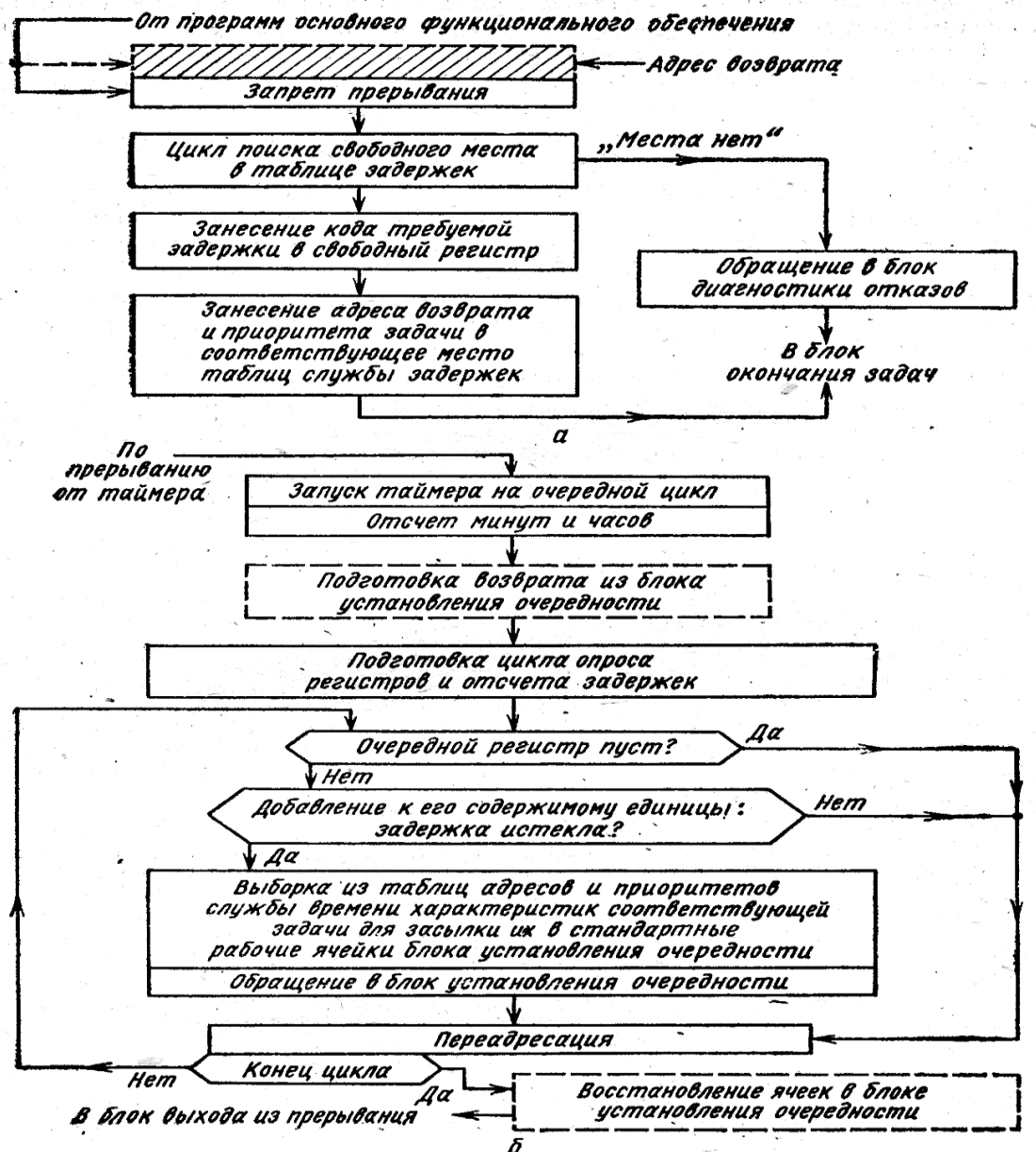


Рис. 8.5. Структурная схема службы времени:
а — блок приема заявок на задержку; б — операции отсчета времени и задержек по прерываниям от таймера (штрихом указаны блоки, зависящие от организации блока установления очередности),

Очередь вывода содержит две таблицы, куда характеристики задач (адреса продолжения и приоритеты) попадают тотчас после обращения к службе вывода. Если канал вывода свободен (заявка является первой в очереди вывода), по адресу возврата запрашивается начальный адрес кодограммы, передаваемый далее блоку собственно вывода. Так как в любом случае задача должна быть снята с обслуживания на все время вывода, а от момента исполнения команды на вывод первого символа до её фактической реализации проходит около 0,1 с, приходится вводить специальный признак вывода первого символа. Кроме того, признак занятости канала вывода имеет дополнительную «окраску» в зависимости от того, идет обычный или экстренный вывод.

Блок собственно вывода имеет два выхода в диспетчер: один после вывода очередного символа — в блок выхода из прерывания, другой по окончании вывода целого сообщения — в службу вывода. В последнем случае анализируется, экстренный или обычный вывод закончился (с тем, чтобы задачу, закончившую обычный вывод, немедленно восстановить в очереди № 1) и не накопился ли ещё экстренный вывод. Если экстренный вывод не нужен, а очередь вывода ещё не исчерпана, она сдвигается вперёд на одну позицию и сведения о первой задаче используются так же, как описано выше.

Структурная схема службы ввода¹ приведена на рис. 8.7. Принято, что любое вводимое сообщение начинается с символа «/» (наклонная черта), имеет не более 10₁₀ знаков и заканчивается «*» (звездочкой). Первые два символа вводимого сообщения определяют вид запрашиваемой задачи, последующие — соответствующие настроечные параметры. Если символ «/» не был введен, все последующие символы игнорируются; всегда игнорируется знак «—» (пробел). В ответ на ошибки оператора производится возврат каретки и выдается на печать «1» или слово «ОШИБКА».

После приема очередного символа и приформирования его к ранее введенной части сообщения служба ввода передает управление блоку выхода из прерывания, а после получения «*» и обработки сообщения — блоку установления очередности

¹ Программа службы ввода разработана А. С. Ждановым

(вместе с характеристиками расшифрованного запроса).

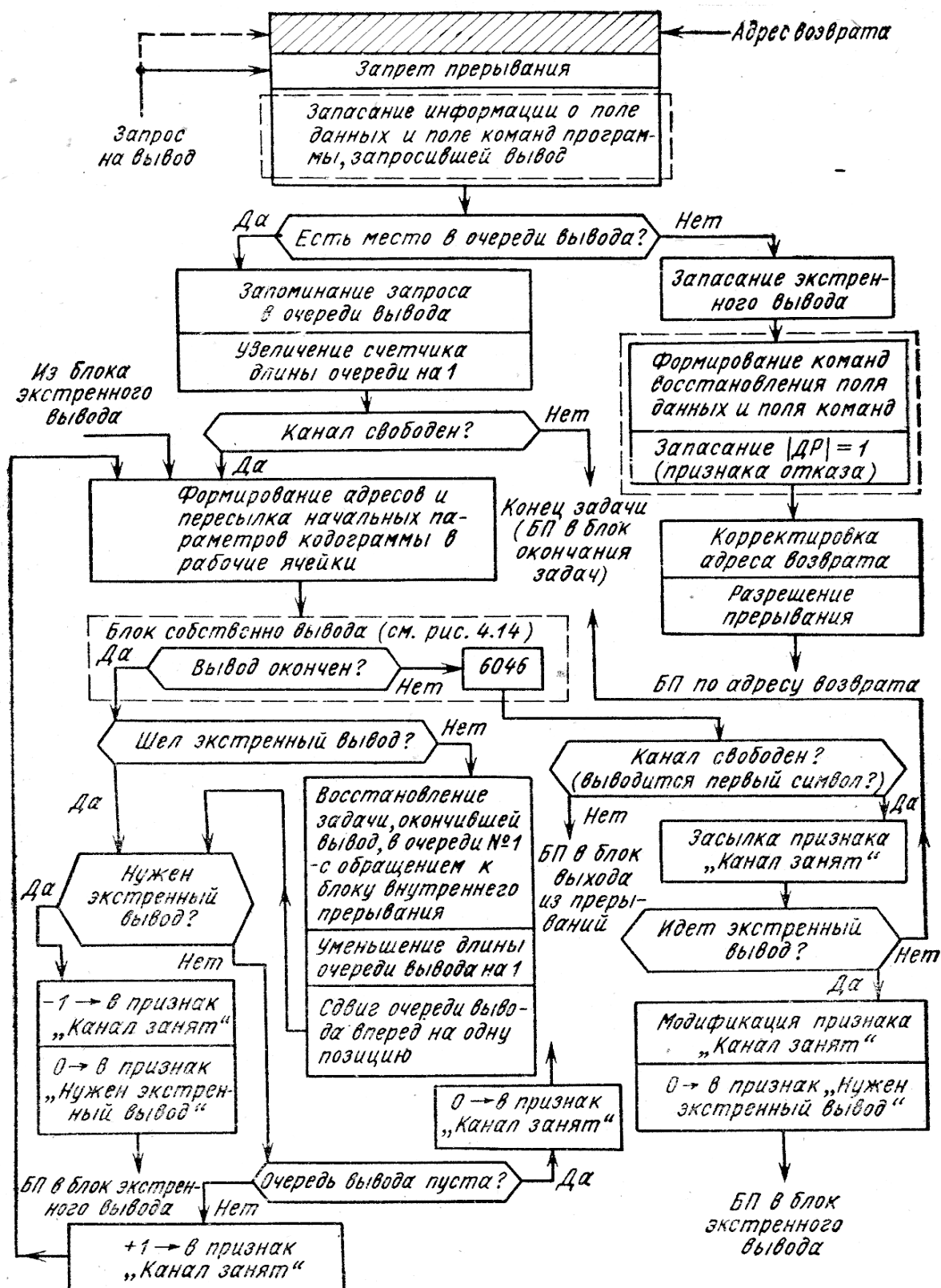


Рис. 8.6. Структурная схема построения службы вывода; блоки, обведенные дополнительной штриховой рамкой, используются при работе с расширенным МОЗУ.

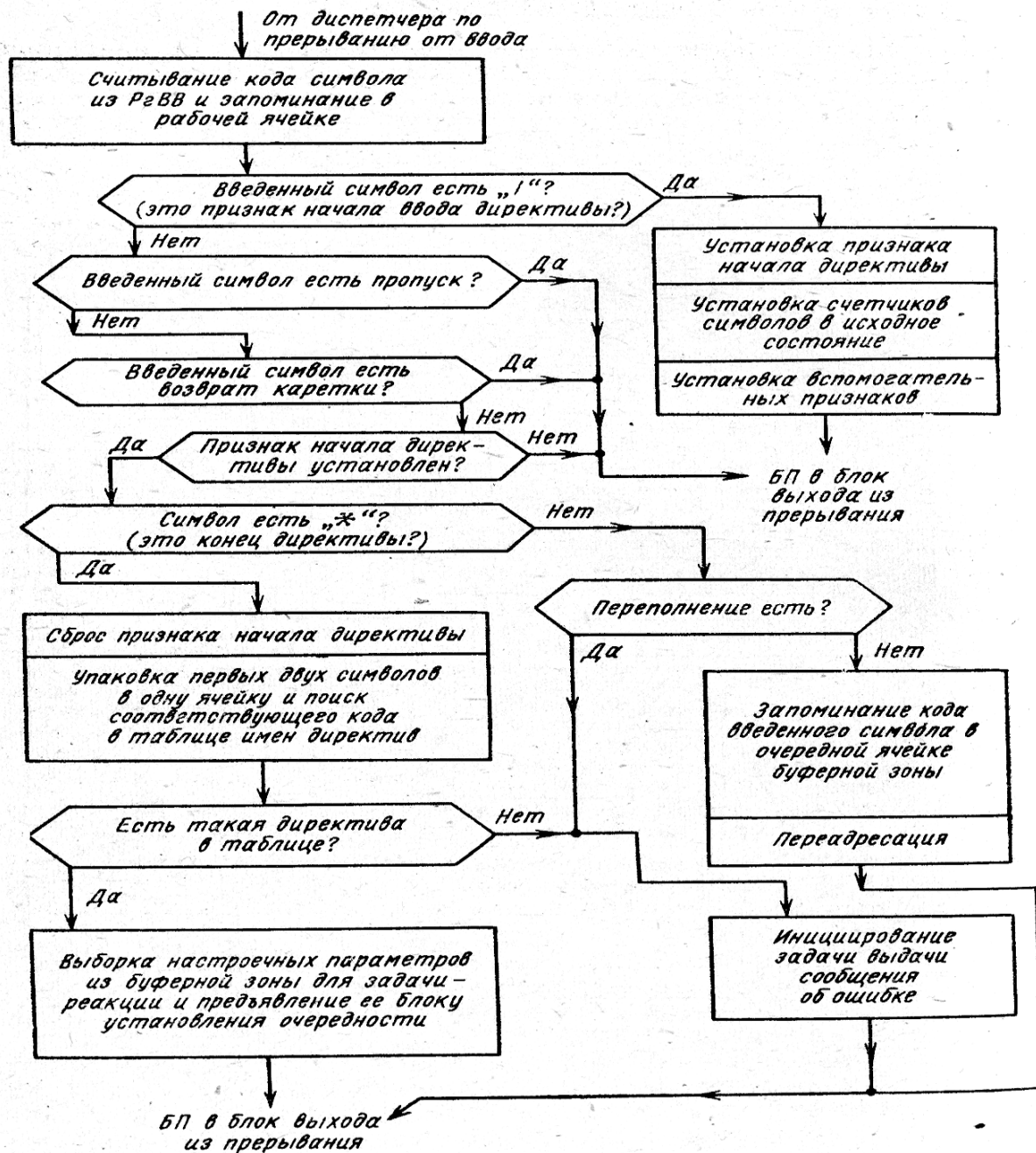


Рис. 8.7. Структурная схема службы ввода.

8.5. ДИАГНОСТИКА И ФИКСАЦИЯ ОТКАЗОВ

Нормальная работа диспетчера, связанная с организацией прерываний, очередей, принятием запросов оператора, может нарушаться вследствие различных помех, наложений, ошибок оператора и т. п. В отдельных случаях это связано с потерей заявок или невозможностью продолжения нормальной работы. Поэтому всегда, когда логический анализ позволяет выявить подобную ситуацию, целесообразно немедленно информировать об этом оператора, сообщая о виде сбоя или отказа и его влиянии на основное программное обеспечение системы. Такой путь относительно просто реализуется благодаря тому, что диспетчеру передается управление при каждом прерывании, при обращении функциональных программ к службам диспетчера и по окончании решения каждой задачи. При этом логический контроль работы основных блоков диспетчера позволяет выявить не только отказы, связанные с ограниченностью ресурсов машины, но и часть сбоев, носящих достаточно общий характер.

Собственно обнаружение того или иного отказа происходит в процессе работы блоков обслуживания прерывания, блока установления очередности и служб диспетчера. Уточнение характеристик отказа и подготовку его фиксации на печать осуществляет специальный блок диагностики и фиксации отказов, которому (как подпрограмме) передается управление в каждом таком случае. Все экстренные сообщения объединены в две группы — с полностью готовым текстом (готовой кодограммой) и с дополнительным формированием уточняющих параметров, выводимых на печать. В соответствии с этим по адресу запрашиваемого блока (либо по содержимому НР (СМ) — в зависимости от принятой структуры обращения к блоку диагностики) формируется информация для обращения к службе вывода (начальный адрес кодограммы и текст изменяемых частей), запасается признак необходимости экстренного вывода, обеспечивающий его начало немедленно по освобождению канала вывода, и управление возвращается блоку диспетчера, обнаружившему сбой.

Структурная схема блока диагностики и фиксации отказов приведена на рис. 8.8. Блок имеет единственное место для адреса кодограммы экстренного сообщения. Поэтому до приема этого сообщения к обслуживанию службой вывода канал экстренного

своих задач). Приоритеты задач имеют натуральную восьмеричную нумерацию и могут назначаться системным программистом как константы программ (приоритет считается тем выше, чем меньше его номер). Если задача снимается с обслуживания ввиду появления заявки с более высоким приоритетом, то в буферной зоне очереди № 2 сохраняется следующая информация о прерванной задаче: содержимое накопительного регистра (сумматора) и ДР (ДРС) в момент прерывания, адрес прерывания и приоритет задачи, содержимое всех (10₈) индексных ячеек.

Для простоты организации очередей приоритет запроса сравнивается только с приоритетом первой задачи из очереди № 1; если приоритет не выше, то соответствующая задача помещается в конец очереди № 1. Заявка попадает в конец очереди № 1 и в том случае, если очередь № 2 заполнена до предела (чтобы не терять заявки).

Служба времени, помимо организации счёта часов и минут по прерываниям от таймера ведет отсчёт заданных выдержек времени в пределах $(1-4095_{10}) \times (40\ m)$ мс, где $1 < m < 4096_{10}$ — программно задаваемая константа; 40 мс — шаг следования импульсов от внутреннего генератора таймера.

С использованием службы времени легко организуется циклическое решение задач во времени — достаточно по окончании решения задачи обращаться не в блок окончания задач, а назначать задержку, вслед за которой вновь передавать управление на начало задачи. Через службу времени легко осуществляется первоначальный запуск отдельных функциональных программ спустя требуемый интервал времени от начала работы системы.

Блок печати в службе вывода позволяет распечатывать на пишущей машинке сообщения, которые подготовлены в виде кодограмм, с возможностью использования упакованной информации, расположенной в произвольном числе разнесенных групп, перевода $2 \rightarrow 10$ в процессе вывода, дополнения выводимой информации необходимым числом пробелов, расстановки запятых в нужном месте десятичных дробей и т. п.

Блок ввода обеспечивает прием и исполнение 14 директив оператора.

Для удобства составления программ основного функционального обеспечения системы на нулевую страницу вынесены лишь стартовые адреса основных блоков и служб диспетчера и минимальное число его команд, обязанных располагаться на нулевой странице.

При переполнении очередей диспетчер игнорирует (теряет) заявку, но печатает в экстренном порядке сообщение о виде одного из трех фиксируемых отказов и адрес отказанной задачи; при появлении неизвестного прерывания машина выводится на останов.

Нетрудно видеть, что даже для минимального комплекта оборудования диспетчеризация основных процессов в машине заметно повышает эффективность её использования.

Полный текст программы-диспетчера для минимального комплекта ЭВМ приведен в приложении 3.

Разработан вариант программы-диспетчера для «Электроники - 100» с расширенным объёмом памяти. В дополнение к указанным функциям диспетчера для минимального комплекта предусмотрена возможность работы с задачами, имеющими различные поле данных и поле команд (в произвольных кубках МОЗУ), и возможность подключения до 15 внешних устройств, которым разрешен запрос новых задач по каналу прерывания.

Блок установления очередности оформлен в виде блока внутреннего прерывания с обращением к нему, как к подпрограмме, что обеспечивает дополнительную возможность предъявления диспетчеру отдельных задач по заявкам, возникающим в ходе выполнения программ основного обеспечения системы. В службе времени добавлен блок, обеспечивающий восстановление задачи в очереди № 1 в точно назначенное время. Расширен состав диагностируемых отказов. Заявки, обслуживаемые по прерываниям от ввода, рассматриваются как обычные задачи, но с наивысшим приоритетом, что позволяет службе ввода пользоваться всеми остальными средствами диспетчера так же, как это делают обычные задачи пользователей.

Для хранения приоритета задачи, её поля команд и поля данных используется одна ячейка. Структура расположения данных при этом имеет вид:

0	1	2	3	4	5	6	7	8	9	10	11
0	Приоритет					Поле команд			Поле данных		

Нетрудно видеть, что для поля данных и поля команд отведено то же место, которое соответствующие коды занимают при считывании содержимого РЗП (по команде 6234). В оставшейся части разрядной сетки располагается код приоритета, причем нулевой разряд всегда содержит 0, чтобы после выделения приоритета (путем логического умножения на константу 3700 или 7700) он всегда оказывался положительным числом согласно представлению чисел в «Электронике - 100» (см. гл.2)¹.

В правила обращения к блокам и службам диспетчера для ЭВМ с расширенным объёмом МОЗУ добавляется требование обязательной засылки своего номера поля команд в специально выделенную ячейку межкубковой связи (см. гл. 4). Содержимое этой ячейки сохраняется диспетчером в очереди № 2 вместе с информацией об адресе продолжения задачи, текущем состоянии НР (СМ) и ДР (ДРС), текущем состоянии полей данных и команд, приоритетом и содержимым индексных ячеек того поля команд, в котором находилась задача в момент последнего прерывания, если эта задача снимается с обслуживания из очереди № 1.

Для обеспечения достаточной глубины прерывания при работе с расширенным МОЗУ блок обслуживания прерываний сразу по получении управления не только запоминает содержимое ДР (ДРС) и НР (СМ), но и считывает (с запоминанием в стандартной рабочей ячейке) содержимое РЗП. В блоке выхода из прерывания содержимое этой ячейки используется для формирования и последующего исполнения команд раздельного восстановления поля команд и поля данных².

Среди внешних устройств, присылающих свой запрос в машину по каналу прерывания (помимо таймера и пишущей машинки), два устройства могут подключаться как обычные внешние устройства, а остальные 12₁₀ — через входной коммутатор

¹ Пяти оставшихся разрядов достаточно для того, чтобы число возможных приоритетов было не менее 31. За счёт некоторого усложнения программы (добавления циклического сдвига вправо на один разряд) это число могло бы быть удвоено, однако и в принятом варианте возможное число приоритетов перекрывает оптимальный диапазон [15].

² Последний прием был бы излишним, если бы в системе команд «Электроники-100» имела команда засылки содержимого НР (СМ) в РЗП или прямо в регистры поля команд и поля данных.

дискретных сигналов.

Объём этого варианта диспетчера и программ ввода-вывода составляет около 5000_8 ячеек при общем объёме МОЗУ в системе от 1 до 8 кубов по 4096_{10} слов в каждом.

Полный текст программы приведен в приложении 3а.

8.7. ОБМЕН ИНФОРМАЦИЕЙ С ЦВМ, РАБОТАЮЩИМИ В АСУП И В СИСТЕМЕ КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ

Эта задача имеет несколько аспектов. Обмен информацией с машинами, работающими в контуре систем более высокого иерархического уровня, может преследовать две цели: снабжение этих систем текущей или обобщенной информацией о состоянии управляемого технологического процесса и использование ресурсов и программного обеспечения более мощных систем для решения своих сложных задач и получения дополнительной информации для её последующего использования в АСУ ТП. Оставляя, по возможности, в стороне вопросы алгоритмического обеспечения этих задач, остановимся здесь лишь на проблеме собственно обмена необходимыми массивами информации.

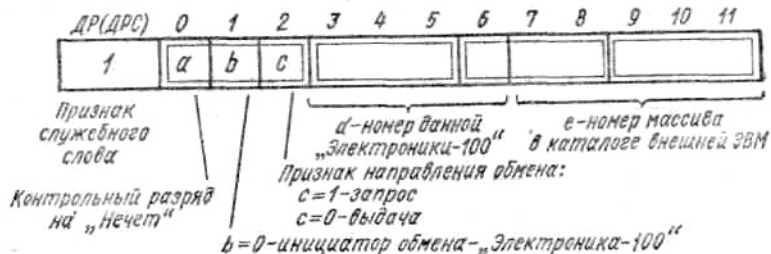
Из технических вопросов первостепенными представляются вопросы синхронизации работы машины, разрядности слов обмена и способа их передачи (последовательно по битам или параллельным кодом). Если исходить из общих возможностей «Электроники - 100» и её типичных каналов в АСУ ТП, наиболее целесообразным представляется асинхронный обмен отдельными 12-разрядными словами параллельным кодом с использованием канала программного прерывания¹. В этом случае другая ЦВМ рассматривается как некоторое внешнее устройство со своим номером, флагом и вводным и выводным буферными регистрами, причем последние могут подключаться к «Электронике -100» соответственно через выходной и входной коммутаторы дискретных сигналов. Асинхронность обмена позволяет легко организовать связь с разнотипными (и обычно более медленными) ЭВМ, работающими на других иерархических уровнях, независимо от их загруженности и быстродействия.

Аппаратурная часть должна обеспечивать необходимое согласование по уровням сигналов и их адаптацию к каналам связи.

Остальная нагрузка, как обычно, ложится на программное обеспечение системы. Основными вопросами здесь являются определение требований к объёму и идентификации массивов, а также вопросы защиты от сбоев и искажений.

Объём массива важен лишь для определения буферной зоны ЭВМ, работающей на ввод, — целесообразно иметь объём отдельных массивов, по крайней мере, не более объёма одного куба памяти (в конкретных условиях эта цифра обычно всегда существенно ниже).

Идентификация запрашиваемого массива является более сложным вопросом. Поскольку предусматривается обмен полными 12-разрядными кодами, идентификатор массива может отличаться от собственно информационных слов только своим местом в посылке. Однако большая надёжность достигается, если ввести специальный дополнительный бит для этой цели. Функции первого слова обмена как служебного слова могут быть расширены для указания, кто является инициатором обмена, на ввод или вывод информации прислан запрос. Часть разрядов может быть отведена также для кода вызываемого абонента и для контроля служебного слова на неточность. В одной из практических систем структура служебного слова имела следующий вид:



В служебном слове указывается только код (номер) запрашиваемого массива, по которому его начальный адрес и число кодов выбираются из таблиц, хранимых в памяти обеих машин, а также адрес подпрограммы предварительной обработки информации на передающей стороне. По другим служебным признакам происходит контроль служебного слова и настройка программы на прием либо передачу массива.

Остальные особенности программы связаны с принимаемыми мерами по защите массива от сбоев. В качестве достаточно простого и эффективного средства их выявления может служить, например, использование контрольной суммы массива. Кроме того, на случай потери одного из сигналов прерывания должны быть приняты дополнительные меры для окончания сеанса связи, как на приемной, так и на передающей стороне. Это может быть сделано путем контроля времени связи. При несовпадении контрольных сумм рационально предусмотреть повторение сеанса связи.

Рассмотренные принципы реализуются в программе, структурная ехема которой приведена на рис. 8.9; временная диаграмма для одного из возможных вариантов связи показана на рис. 8.10.

¹ Весь дальнейший подход можно рассматривать как одно из возможных решений задачи, которое, естественно, изменится, если принять за основу, например, побайтный обмен.

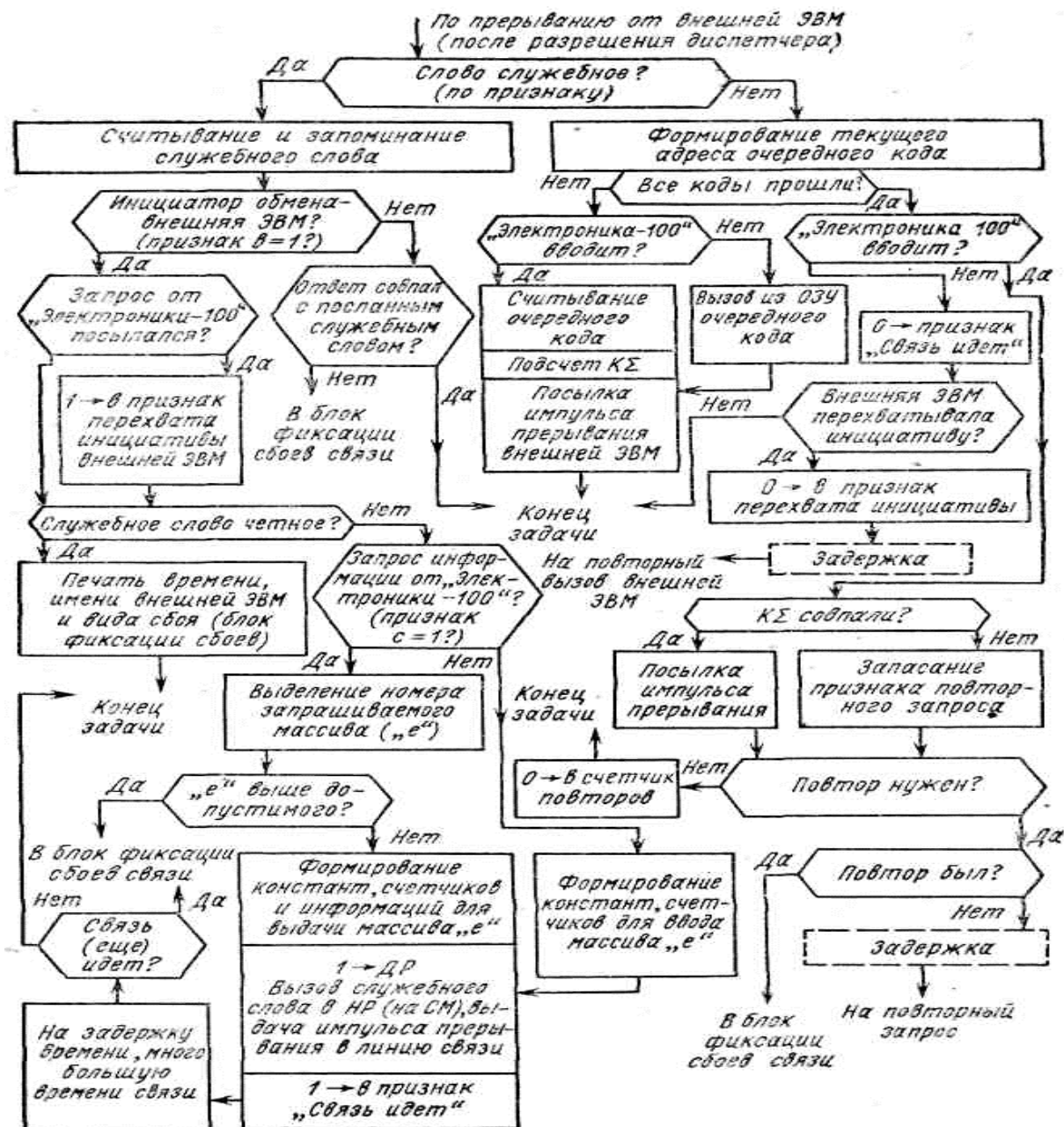


Рис. 8.9. Структурная схема блока обмена информацией с внешней ЭВМ

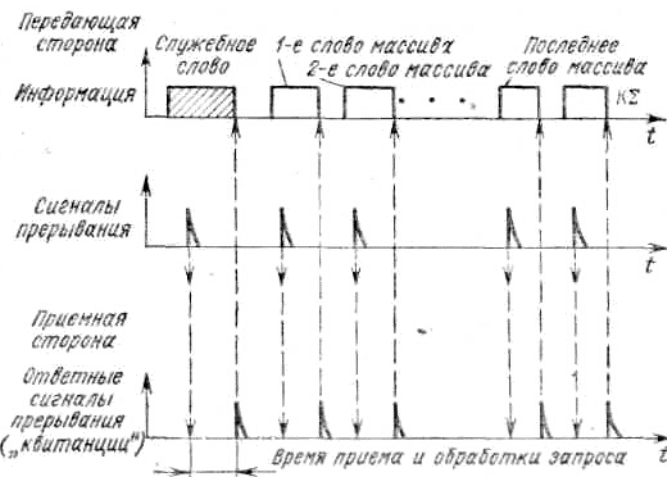


Рис. 8.10. Временная диаграмма связи для случая, когда инициатором обмена является передающая сторона.

Каждый вход в программу, функционирующую в общем обеспечении системы как одна из обычных задач, осуществляется после передачи ей управления от диспетчера в результате прерывания по сигналу от внешней ЦВМ. В своей работе эта программа

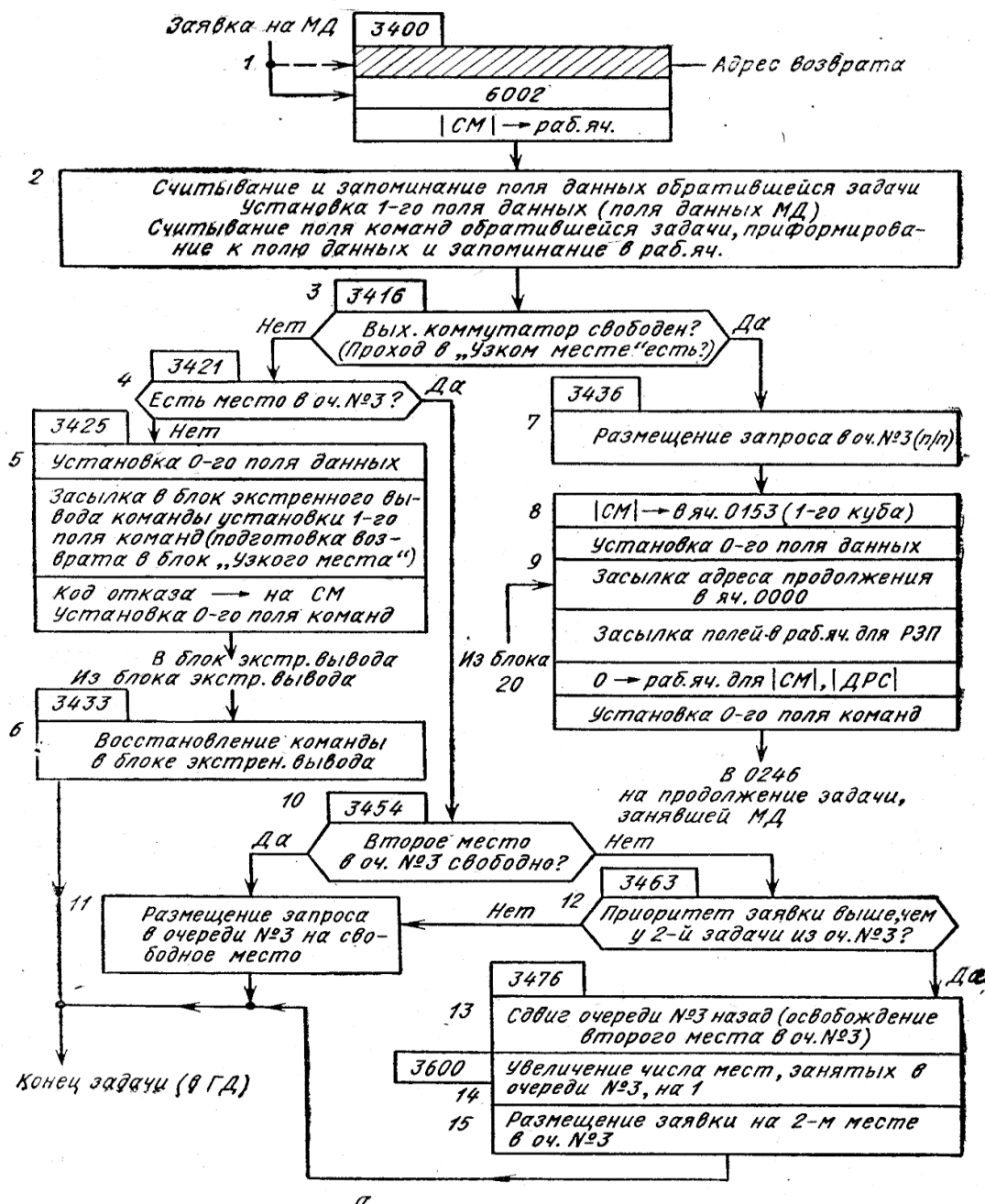
использует также службу времени диспетчера.

Объём программы без учёта буферных зон — около двух страниц памяти.

8.8. ДРУГИЕ РАСШИРЕНИЯ ФУНКЦИЙ ДИСПЕТЧЕРА

Увеличение возможностей системы в целом за счёт роста состава оборудования и программных блоков основного обеспечения приводит к росту запросов к диспетчеру. Прежде всего, это требует расширения функций (и состава операций) старых блоков: удлиняется цепочка опрашиваемых флагов в блоке обслуживания прерывания, в числе характеристик задач добавляются номера кубов памяти, используемых для хранения программы и данных. Появляются и новые программные блоки, организующие обмен с новыми внешними устройствами (см., например, § 8.7). Среди этих блоков есть такие, необходимость в которых возникает довольно часто. В первую очередь здесь можно назвать блок «узкого места» и ввод новых участков программ или числовых данных с перфоленты параллельно с работой системы.

«Узкие места» обязаны своим происхождением в основном мультипрограммному режиму работы оборудования в системе управления. Нетрудно представить себе ситуацию, когда задача с высоким приоритетом, вытеснив с обслуживания у процессора другую, менее приоритетную, пытается использовать те же подпрограммы либо внешние устройства, с которыми уже начала работать только что снятая (в очередь № 2) задача. В таких подпрограммах окажется затёртым адрес возврата в старую задачу; отработка управляющего воздействия, подготовленная и осуществляемая снятой задачей, например, через выходной коммутатор дискретных сигналов, прервется и может так и не закончиться. Аналогичная картина в отношении внешних устройств может произойти и с задачей высокого приоритета, если ей для отработки необходимых операций нужна задержка во времени (так как на время задержки служба времени изымает задачу из очереди № 1).



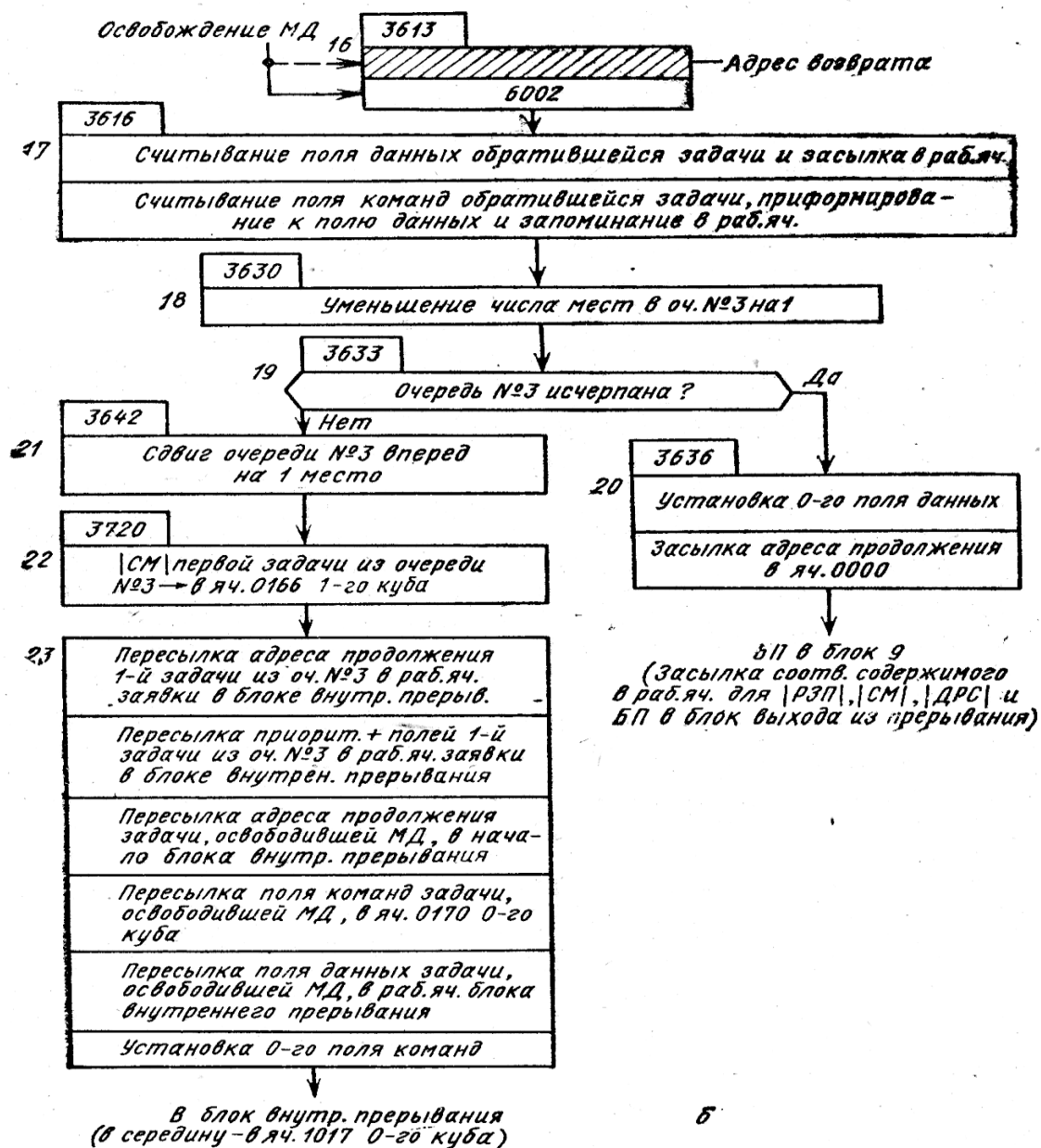


Рис. 8.11. Структурная схема блока «узкого места» (местного диспетчера):
а - «загрузка МД; б — освобождение МД,

Есть несколько обходных путей для преодоления «узких мест» с помощью уже разработанных средств диспетчера, в частности: временное запрещение прерывания (или повышение приоритета обслуживаемой задачи), использование индексных ячеек (содержимое которых при снятии задачи из очереди № 1 сохраняется в очереди №2) в качестве рабочих. Однако в полной мере действенным средством является лишь специальный блок «узкого места». Это своеобразный блок приема к обслуживанию заявок на использование устройств или программ, которые являются «узким местом» в системе. Подобно аналогичным блокам в службе времени и вывода этот блок должен иметь свою буферную зону для хранения исходных данных сдерживаемого потока задач. Как и в службе вывода, здесь может быть организована очередь, где первая задача никогда не снимается с обслуживания, а остальные расставляются по приоритетам. Если к моменту обращения в блок «узкого места» его очередь была пуста, задача попадает на первое место в этой очереди и сохраняется в очереди № 1 (так как продолжает обслуживаться процессором). Все последующие задачи, попадающие не на первое место в очереди «узкого места», снимаются с обслуживания (из очереди № 1) и ожидают освобождения блока. По окончании обслуживания очередной задачи блоком «узкого места» вся его очередь сдвигается на одну позицию вперед и задача, попавшая здесь на первое место, немедленно предьявляется блоку установления очередности — для включения её в очередь № 1 в соответствии с её истинным приоритетом.

Структурная схема блока «узкого места» приведена на рис. 8.11. Объем программы с очередью на семь мест составляет около двух страниц¹.

Для оперативного ввода новых участков программы с перфоленты заманчиво использовать только ту аппаратуру, которая уже есть в минимальном комплекте системы. Однако один и тот же канал ввода используют считыватель с перфоленты и пишущая машинка «Консул». Поэтому в программе оперативного ввода с перфоленты необходимо предусмотреть возможность

¹ В достаточно сложных системах подобных «узких мест» и соответствующих блоков может быть несколько. В связи с этим важную роль для экономии памяти начинает играть хорошая общая организация системы и, в частности библиотеки стандартных и часто используемых подпрограмм.

переключения необходимых тумблеров на панели блока управления ВВУ.

Для оператора средством оперативного запроса самого процесса ввода остается по-прежнему пишущая машинка, поэтому в исходном состоянии именно она должна быть подключена к каналу ввода. После получения директивы оператора о вводе с перфоленты специальная исполнительная программа должна перестраивать службу ввода и инициировать начало движения перфоленты, например, путем периодического исполнения команды 6032. При этом с использованием разобранных выше блоков диспетчера удастся организовать обычное обслуживание остальных задач до того момента, пока оператор не включит считыватель с перфоленты и не переведет переключатель «КЛ — СЧ» на передней панели блока управления ВВУ в положение «СЧ».

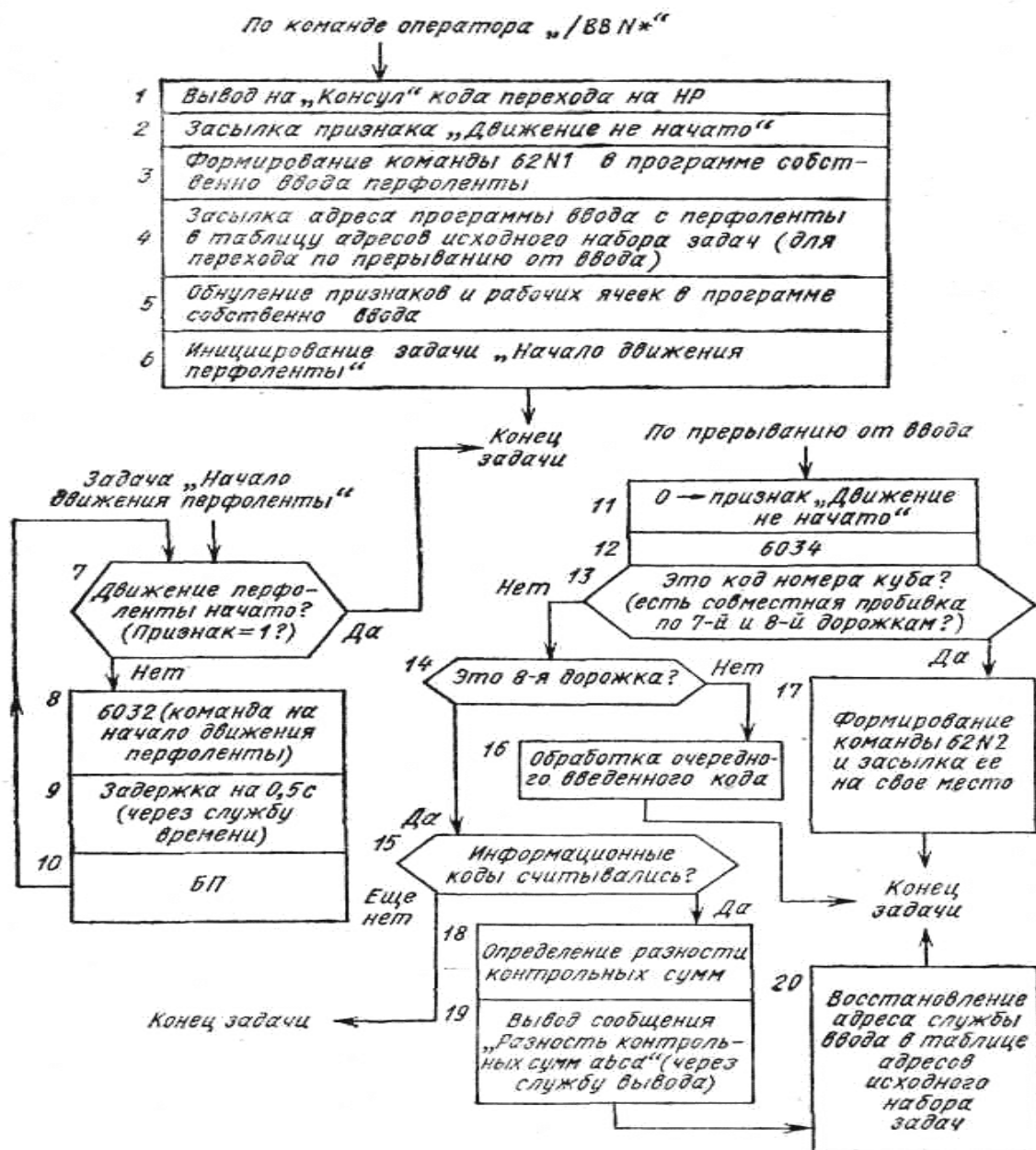


Рис. 8.12. Структурная схема программы ввода информации с перфоленты при работе с диспетчером; (укрупненный) блок 16 выполняет все функции, свойственные программе "Ввод 2".

Тогда очередная команда 6032 приведет перфоленту в движение, и после считывания её очередной дорожки произойдет прерывание по вводу, инициирующее начало работы службы ввода и собственно программ ввода. Последние программы, по своим функциям аналогичные обычным программам "Ввод 1" или «Ввод 2», должны быть перестроены лишь в части использования процессора в режиме прерывания вместо монопольного режима. Дополнительной особенностью в них может служить также контроль границ вводимого массива с целью охраны от затирания программ диспетчера. По окончании ввода (когда через считыватель начинает проходить участок перфоленты с пробивкой по 8-й дорожке)¹ оператор переводит переключатель «КЛ — СЧ» в положение "КЛ" и отключает считыватель. При вводе, подобном «Вводу 2», может быть предусмотрена выдача на печать информации о совпадении контрольной суммы. Структурная схема подобной программы ввода приведена на рис. 8.12. Объём программы около двух страниц памяти.

Важно отметить, что возможность практического использования такой программы существенно зависит от

¹ При работе с диспетчером все флаги должны быть погашены. Однако команда гашения флага ввода 6032 одновременно инициирует движение перфоленты, поэтому остановить перфоленту в считывателе тотчас после появления пробивки по 8-й дорожке, не останавливая машину или не выключив считыватель, в данном случае не представляется возможным.

помехозащищенности аппаратуры, в частности, от возможности безболезненного переключения выключателей для работы процессора и схем управления ВВУ при решении текущих задач.

9. ТИПОВАЯ СТРУКТУРА СИСТЕМЫ И БЛОЧНЫЙ ПРИНЦИП ПОСТРОЕНИЯ ПРОГРАММ

Автоматизированные системы управления интенсивно разрабатываются для весьма широкого круга технологических процессов в различных отраслях промышленности. Однако стержневым вопросом построения системы всегда остается рациональная организация процедур получения и использования текущей информации, отражающей истинное состояние объекта в каждый момент времени. Это обстоятельство предопределяет возможность выделения типовых алгоритмических и программных блоков не только в части диспетчеризации управления работой ЦВМ, но и в общем математическом обеспечении функциональных задач АСУ ТП. К настоящему времени типовые структуры систем управления и значительная часть общих вопросов их программного обеспечения достаточно подробно описаны в литературе (см. [16—24])¹. Вот, например, круг задач, которые считаются достаточно типичными для современных АСУ ТП в энергетике [28]:

1. Гибкий обогатяющий контроль большого числа параметров.
2. Выработка аварийных сигналов в результате сравнения измеряемых величин с (фиксированными и вычисляемыми) граничными значениями их амплитуд и скорости изменения.
3. Накопление и обработка информации с периодическим выводом на печать по требованию оператора или по команде УЦВМ при нарушении режима.
4. Обработка данных телеметрической системы по определенным алгоритмам (равномерное или взвешенное усреднение на некотором интервале, интегрирование, определение скорости изменения сигналов, запоминание максимального и минимального значений и т. п.)
5. Введение оператором с пульта команд на изменение режимных параметров системы, запросов на выдачу сообщений и т. п. с возможностью простого и удобного изменения и дополнения функций.
6. Использование машины во время холостого хода для решения фоновых задач системы (вычисление показателей работы, обработка прошлой информации из ДЗУ, отработка новых программ) за счёт введения различных приоритетов задач.
7. Изменение настроечных констант программ при изменении числа или характеристик агрегатов.
8. Расчёт экономичных режимов и слежение за отклонениями от них.
9. Учёт вероятностного характера отклонений на входе системы регулирования.
10. Предотвращение перегрузок путем регулирования по возмущению.
11. Выявление сбоев и отказов телемеханики и регуляторов.
12. Перераспределение нагрузок у однотипных агрегатов.
13. Оптимизация системы в целом и повышение бесперебойности в работе.
14. Решение задач введения горячего резерва.
15. Использование ЭВМ для управления отдельными агрегатами большой мощности.

Приведенный список задач, по-видимому, мог бы быть расширен (см. гл. 7 и 8, а также [29]), однако в рамках данной главы целесообразно остановиться лишь на некоторых вопросах реализации соответствующего программного обеспечения на «Электронике-100» и наиболее тонких алгоритмических вопросах, составляющих основу построения этих программ.

9.1. СБОР И ПЕРВИЧНАЯ ОБРАБОТКА ИНФОРМАЦИИ

При сборе исходной информации ядро программного блока составляет опрос датчиков. Периферийное оборудование «Электроники-100» позволяет вводить в машину (см. гл. 1) аналоговые сигналы, изменяющиеся в диапазоне 0—10 В, дискретные сигналы, принимающие лишь два значения на уровне 0 и — 3,7 В, и частотные сигналы в диапазоне 4—8 кГц. Число аналоговых и частотных сигналов, вводимых в машину, может быть доведено до 512 за счёт использования коммутатора аналоговых сигналов, число дискретных сигналов — до 432, причем при вводе в машину они компонуется в 6 и 12-разрядные слова.

Таким образом, в опросе каждого датчика обычно участвует тот или иной коммутатор, а для аналоговых и частотных датчиков ещё и блок преобразования сигналов в код. В связи с этим, помимо команды считывания показаний датчика в НР (СМ), процедура опроса датчика «обрастает» командами управления коммутатором и блоками преобразования сигналов в код. Последовательность таких команд², обеспечивающих работу с образцами периферийного оборудования, оформленного в виде центрального поста и выносных постов, рассматривалась в гл. 3. Для нового комплекта периферийного оборудования «Радуга» унифицированы команды обращения к любому внешнему устройству через (общий) блок управления коммутацией (БУК). Эти команды имеют следующий вид.

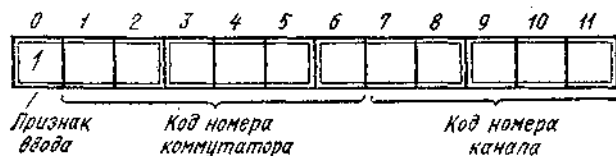
Для ввода дискретных сигналов через входной коммутатор дискретных сигналов (6-разрядными словами):

Слово управления коммутацией →НР (СМ)
6324
7200
6322

¹ Ряд вопросов применения «Электроники-100» в АСУ ТП различного назначения рассмотрен в [24—27]

² В малых машинах родственного назначения набор подобных команд, обеспечивающих всю необходимую последовательность обмена с внешним устройством, называют иногда «драйвером».

Здесь слово управления коммутацией имеет формат



и обеспечивает, таким образом, обращение к любому из 64 коммутаторов, каждый из которых может иметь 32 канала.

(На практике, конечно, лишь часть коммутаторов используется для ввода дискретных каналов, и при компоновке системы им присваиваются определенные коды.)

Команда 6324 обеспечивает запись слова управления коммутацией в регистр коммутации БУК, после чего включается запрашиваемый канал указанного коммутатора; по команде 6322 информация от соответствующей группы дискретных датчиков заносится в шесть младших разрядов НР (СМ).

Для ввода аналоговых сигналов выполняется такая последовательность команд:

Слово управления коммутацией →НР(СМ)
6324
7200
6004
6325

Здесь слово управления коммутацией имеет такой же формат, как и при вводе дискретных сигналов, лишь коды номеров коммутаторов для аналоговых сигналов закреплены свои (отличные от тех, которые используются для дискретных сигналов). Команда 6004, как и ранее, запускает АЦП, так что через 50 мкс в НР (СМ) оказывается код, соответствующий исходному аналоговому сигналу. Команда 6325 обеспечивает закрывание электронных ключей после обработки аналогового сигнала.

Следует отметить, что формат слова управления коммутацией позволяет всем сигналам (и аналоговым и дискретным) присвоить сквозную нумерацию, что может представлять определенные удобства при организации циклов опроса датчиков, и затем лишь позаботиться о соответствии номеров коммутаторов и их каналов принятой системе нумерации. (Техническая реализация БУК «Радуги» обеспечивает присвоение любому коммутатору номера, код которого набирается переключателями на его передней панели; входные сигналы от каждого из каналов вводятся через разъёмы.)

Блок управления коммутаторами имеет (групповой) флаг, который могут устанавливать в единичное состояние входные сигналы, специально скоммутированные для этой цели при монтаже системы. По команде 6323 происходит опрос флага БУК, установка его в нуль и пропуск очередной команды, если ранее флаг был в единичном состоянии.

Дальнейшее развитие программного блока опроса датчиков происходит за счёт предварительной обработки получаемой информации и организации циклического опроса всех датчиков либо опроса отдельных датчиков (например, по запросу оператора).

Предварительная обработка сигналов преследует весьма разнообразные цели — от нормализации и распаковки кодов, получаемых при единичных измерениях, до цифровой фильтрации ряда последовательных замеров — для повышения точности представления исходных данных в машине либо для внесения динамических коррекций. Простейшая обработка в основном зависит от типа датчика. Так, для аналоговых сигналов код, получаемый в НР (на СМ) после работы АЦП, занимает десять старших разрядов и содержит дополнительную единицу в 10-м разряде (как признак окончания работы АЦП), что приводит к необходимости дополнительного сдвига кода в целях нормализации вправо на два разряда с предварительной очисткой ДР (ДРС). Информация о состоянии отдельного контакта может содержаться в виде одного разряда целого «слова», получаемого при опросе группы дискретных датчиков через коммутатор дискретных сигналов, так что требуется, например, дополнительная команда логического умножения для выделения такой информации и т. д.

Предварительная фильтрация сигнала, обычно, прежде всего, преследует цель снижения влияния помех и аппаратных погрешностей, искажающих истинное значение сигнала при каждом измерении (рис. 9.1). Поскольку систематические погрешности измерителей можно устранить при тарировке, основное внимание уделяется борьбе с флуктуационной знакопеременной помехой. Наиболее распространенным способом фильтрации является усреднение ряда последовательных отсчётов. Однако на том же интервале усредняются и текущие значения сигнала, так что существует некоторый оптимальный интервал усреднения, обеспечивающий минимальное результирующее искажение сигнала, причем величина этого интервала, очевидно, должна зависеть от соотношения амплитудных и временных характеристик сигнала и помехи.

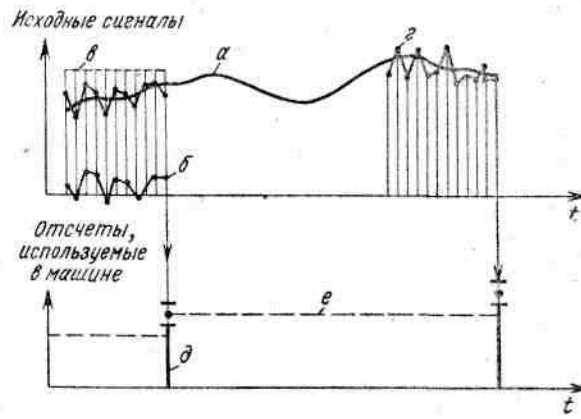


Рис. 9.1. Предварительная обработка сигналов в машине:
 a — исходный сигнал; b — погрешность измерения (помеха); z — огибающая функции отсчетов; z — огибающая фактических замеров; d — значение сигнала, участвующее в дальнейших расчетах (горизонтальными линиями отмечены пределы изменения остаточных флуктуаций помехи после обработки); e — экстраполируемое значение сигнала

Анализ точности и выбор оптимального интервала усреднения производится на основании эквивалентной схемы рис. 9.2 методами теории случайных функций [30]. Можно показать, что для равномерного усреднения дисперсия результирующей погрешности определяется выражением

$$\sigma_{\varepsilon}^2 = \sigma_x^2 - \frac{2}{T^2} \int_0^T \tau K_{xx}(\tau) d\tau + \frac{2}{T} \int_0^T K_{nn}(\tau) \left(1 - \frac{\tau}{T}\right) d\tau, \quad (9.1)$$

где σ_x^2 — дисперсия сигнала; $K_{xx}(\tau)$, $K_{nn}(\tau)$ — ковариационные функции сигнала и помехи соответственно.

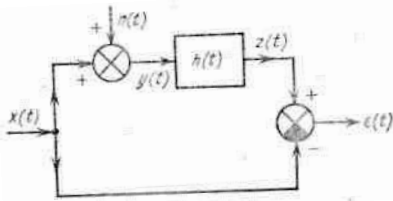


Рис. 9.2. Эквивалентная схема анализа точности предварительной фильтрации; для случая равномерного усреднения

$$h(t) = \frac{1}{T} \text{ при } 0 \leq t \leq T \text{ (0 вне этого интервала)}$$

Дифференцируя результат (9.1) по T и приравнявая производную нулю, получаем соотношение для определения T_{opt} . В частности, для широко распространенного случая экспоненциальных ковариационных функций

$$K_{xx}(\tau) = \sigma_x^2 e^{-\alpha|\tau|}; \quad K_{nn}(\tau) = \sigma_n^2 e^{-\beta|\tau|} \quad (9.2)$$

оптимальный интервал усреднения определяется¹ соотношением

$$T_{opt} = \frac{2}{\beta} \left(1 - \frac{2\sigma_x^2 \alpha}{\sigma_n^2 \beta}\right) \frac{1}{1 - 3\sigma_x^2 \alpha^2 / \sigma_n^2 \beta^2} \quad (9.3)$$

и относительное ослабление влияния помехи составляет

$$\frac{\sigma_{\varepsilon \min}^2}{\sigma^2} = 1 - \frac{(1 - 2\sigma_x \alpha / \sigma_n^2 \beta)^2}{3(1 - 3\sigma_x^2 \alpha^2 / \sigma_n^2 \beta^2)} \quad (9.4)$$

Нетрудно видеть, что такая фильтрация позволяет улучшить исходные точностные характеристики сигнала, особенно в случае существенного различия «ширины» ковариационных функций сигнала и помехи. При этом применение цифровой обработки ряда последовательных отсчетов (цифровой фильтрации) позволяет передать машине для централизованной программной реализации функции повышения качества измерения входной информации, получаемой от многих относительно простых (а, следовательно, и недорогих, что важно для экономических показателей системы в целом) измерителей.

Усреднение последовательного ряда отсчетов одного и того же сигнала при уже выбранном и подключенном с помощью коммутатора канале легко осуществляется несложной программой (рис. 9.3). Накопление суммы производится в двух ячейках, благодаря чему нормировка (путем деления на заданное число отсчетов) осуществляется с использованием стандартной подпрограммы. Основное время в теле цикла для аналоговых датчиков занимает операция преобразования «аналог — код» (50 мкс), минимальный период замеров с учетом длительности исполнения остальных команд около 70 мкс, исходя из чего можно определить необходимое число замеров.

Определенные вычислительные удобства при реализации цифровых фильтров на ЦВМ имеет фильтр с экспоненциальной весовой функцией. Формирование выходного сигнала такого фильтра сводится к использованию рекуррентного выражения, куда входит единственное прошлое значение выходного сигнала и значение входного сигнала в текущий момент времени. Оптимальная величина постоянной времени для такого фильтра и минимально достижимая дисперсия ошибки для сигналов

¹ Приближенное решение промежуточного трансцендентного уравнения осуществлялось путем разложения экспонент в ряд с удержанием четырех первых членов.

и помех с характеристиками (9.2) определяются, как можно показать, следующими соотношениями:

$$T_{opt} = \frac{1}{\beta} \times \frac{1 - \sqrt{xs^{-1}}}{\sqrt{xs^{-1}} - x} \quad (9.5)$$

$$\frac{\sigma_{\varepsilon \min}^2}{\sigma_n^2} = \frac{x}{(1-x)\sqrt{xs^{-1}}} \times \left[\frac{2}{s} - \sqrt{xs^{-1}}(1+s^{-1}) \right], \quad (9.6)$$

где

$$x = \alpha\beta^{-1}, \quad s = \sigma_n^2 \sigma_x^{-2} \quad (9.7)$$

Из сопоставления (9.4) и (9.7) следует, что фильтр с экспоненциальной весовой функцией обладает несколько лучшими точностными свойствами, чем фильтр с равномерным усреднением. Очевидное качественное объяснение такого эффекта состоит в том, что при экспоненциальной фильтрации последние значения сигнала приобретают больший вес и, таким образом, искажаются меньше, чем при равномерном усреднении сигнала. Однако при реализации экспоненциального фильтра необходимо вводить операцию умножения, что снижает быстродействие предварительной обработки сигнала.

Оба рассмотренных способа фильтрации предполагали известной структуру фильтра, так что требовалось лишь определить его оптимальные параметры. Можно ожидать, что существуют ещё какие-то формы весовой функции фильтра, обладающие лучшими свойствами. В этой связи значительный интерес представляет задача определения самой оптимальной структуры цифрового фильтра при заданном числе отсчётов, отведенном для данного канала, и фиксированной частоте отсчетов. Для класса линейных дискретных фильтров, таких, что (см. рис. 9.2)

$$z[iT_s] = \sum_{j=0}^T h[j] y[iT_s - jT_s], \quad (9.8)$$

где T_s — период отсчётов, а $h[j]$ играют роль весовых коэффициентов, их оптимальные значения определяются из системы уравнений, получаемой обычным (для экстремальных задач) путем на основе выражения для второго начального момента сигнала ошибки $\varepsilon[iT_s]$ в дискретные моменты времени:

$$\sum_{j=0}^J h[j] \{K_{xx}[l-j] + K_{nn}[l-j] + x_0^2\} = K_{xx}[l] + x_0^2, \quad l = 0, 1, \dots, J. \quad (9.9)$$

Сумма квадрата постоянной составляющей сигнала и ординат ковариационных функций сигнала и помехи образуют в левой части (9.9) квадратную симметрическую матрицу коэффициентов линейного уравнения в виде суммы трех матриц

$$A = A_{xx} + A_{nn} + Bx_0^2, \quad A_{\xi\xi} = \|\alpha_{ij}\| = \|K_{\xi\xi}[i-j]\|, \quad \xi = x, n, \quad (9.10)$$

а все элементы матрицы. В равны 1.

Используя векторные обозначения

$$h = \begin{Bmatrix} h_0 \\ h_1 \\ \vdots \\ h_J \end{Bmatrix}, \quad K = K_{xx} + Fx_0^2 = \begin{Bmatrix} K_{xx}[0] + x_0^2 \\ K_{xx}[1] + x_0^2 \\ \vdots \\ K_{xx}[J] + x_0^2 \end{Bmatrix}, \quad F = \begin{Bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{Bmatrix}, \quad (9.11)$$

решение системы (9.9) можно определить как

$$h = A^{-1}K \quad (9.12)$$

а соотношение для определения (минимальной) остаточной дисперсии при использовании такого фильтра имеет вид

$$\sigma_{\varepsilon \min}^2 = \sigma_x^2 - K'A^{-1}K, \quad (9.13)$$

(Здесь штрих означает транспонирование вектора).

Последние соотношения позволяют на этапе проектирования системы оценить вид и параметры оптимального фильтра, а также максимально достижимый выигрыш от его применения. Форма соотношений не изменяется и для физически нереализуемого фильтра, который имеет место, например, при отнесении отфильтрованного отсчёта по времени к середине интервала наблюдения. Некоторые результаты для случая, когда ковариационные функции сигнала и помехи описываются выражениями (9.2), показаны на рис. 9.4.

Программная реализация подобных оптимальных фильтров в отличие от равномерного усреднения требует введения в тело цикла программы предварительной фильтрации сигнала (см. рис. 9.3) операции умножения очередного отсчёта на

¹ Важно отметить, что полученные соотношения позволяют непосредственно учитывать конкретные регламентируемые условия цифровой фильтрации в системе. Учёт влияния постоянной составляющей сигнала на процесс оптимальной фильтрации на этапе проектирования фильтра позволяет не вводить дополнительных операций вычитания и последующего восстановления постоянной составляющей при обработке сигнала в реальном времени (что оказывается обязательным, например, при использовании классических методов фильтрации.)

соответствующий весовой коэффициент, сложения не только младших, но и старших разрядов накапливаемой суммы, и принятия дополнительных мер по масштабированию результата. Первая и последняя задача относительно просто решаются, если принять, что сумма весовых коэффициентов равняется 2048. При этом оказывается возможным использовать стандартную подпрограмму умножения чисел, каждое из которых занимает одну ячейку, а результат дополнительно нормировать умножением на 2 (сдвигом на один разряд влево) с использованием старших разрядов суммы (9.8), накопленной в двух ячейках. Для накопления же этой суммы приходится оперировать числами двойной длины, что приводит к появлению в теле цикла ещё нескольких дополнительных операций. Однако основное удлинение времени цикла происходит за счёт появления операции умножения, которая даже в случае использования ДАБ занимает около 10 мкс (без ДАБ обычная подпрограмма умножения идет 460 мкс). Нетрудно видеть, что если все названные вычисления введены в тело цикла, минимальный период повторения отсчётов по уже подключенному каналу возрастает до 100 мкс при использовании ДАБ и до 560 мкс при умножении по стандартной подпрограмме.

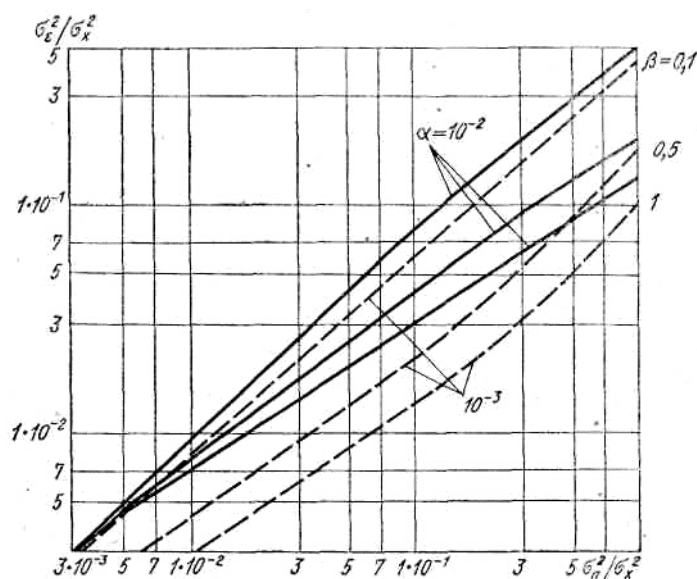


Рис. 9.4. Предельно достижимая эффективность предварительной фильтрации при использовании 20 отсчётов.

Резерв повышения быстродействия при накоплении отсчётов для предварительной фильтрации лежит в выделении дополнительной буферной зоны, куда в цикле собственно измерения с периодом повторения порядка 60 мкс заносятся J последовательных отсчётов, и вынесении операций их обработки в соответствии с (9.8) в отдельный цикл, следующий после окончания цикла измерений. В этом случае общее время фильтрации несколько удлинняется за счёт введения операций подготовки дополнительного цикла, однако увеличение частоты отбора данных позволяет быстрее набрать необходимое число отсчётов, а, следовательно, и уменьшить неизбежные искажения полезного сигнала, вносимые операцией фильтрации вместе со сглаживанием подавляемой помехи.

Общая программа опроса параметров для информационной системы может быть рационально организована в одном (внешнем) цикле, если параметры объединены в смежные группы по признаку однотипности сигналов от датчиков (аналоговых, частотных, дискретных). При этом внутри цикла дополнительно контролируется принадлежность опрашиваемого параметра (его номера) типу подключенного коммутатора, и пока номера параметров не превышают некоторого (конкретного для каждой системы) порога, идет лишь подключение нужного канала уже выбранного коммутатора¹. В момент достижения порога происходит переход на новую ветвь программы с использованием своего коммутатора сигналов либо иного преобразователя исходного сигнала в код. Операции предварительной цифровой фильтрации сигнала, не зависящие от типа исходного сигнала, могут включаться в общий цикл как подпрограмма со своим внутренним циклом. Для достижения возможности использования основных блоков циклического опроса датчиков при измерении значения отдельного параметра (например, по запросу оператора) значительная часть всего внешнего цикла, за исключением формирования номера датчика, оформляется как подпрограмма².

Таким образом, общая программа опроса датчиков с предварительной фильтрацией представляет собой целую систему вложенных друг в друга циклов и подпрограмм, использующих матрицу весовых коэффициентов оптимальных цифровых фильтров.

Сигналы, прошедшие предварительную фильтрацию, используются далее в расчётных соотношениях функциональных программ системы. Часто в этих соотношениях оказываются необходимыми не только текущие, но и прошлые значения параметров, в связи с чем по каждому из таких параметров организуют массив прошлых измерений на необходимую глубину. Перед началом очередного цикла измерения параметров весь массив прошлых значений сдвигают на одну позицию назад, а очередное измеренное значение помещают в начало массива. Если массивы замеров по отдельным параметрам занимают смежные участки в памяти, собственно сдвиг реализуется в простом цикле, так как места засылки новых замеров по каждому из параметров фиксированы и сдвинутые на их место замеры «чужих» параметров будут стерты при засылке новых замеров. Необходимо лишь позаботиться, чтобы при мультипрограммной работе программа обработки сдвигаемых массивов не вклинилась в промежуток времени от сдвига массивов до занесения новых замеров. Последнее условие легко выполняется, например, с использованием приоритетной тактики.

Накопленный массив отдельных замеров сигналов по каждому из параметров, точность представления которых может

¹ В случае использования «Радуги» — изменение младших разрядов слова управления коммутацией

² Подобная программа для «Электроники 100» разработана Ю.М. Петровым.

оказаться существенно повышенной за счёт предварительной фильтрации, является объективной количественной основой для всех дальнейших расчётов текущих и интегральных показателей в системе.

9.2. ОПРЕДЕЛЕНИЕ НАРУШЕНИЙ НОРМАЛЬНОГО РЕЖИМА ПРОЦЕССА

Решение этой задачи является одной из важных функций АСУ ТП. В конечном счёте, требуется по получаемым отсчётам сигналов выявить отклонения, опасные с точки зрения нормального хода процесса либо возможности его контроля. Сами отклонения могут вызываться как дрейфом технологического режима, так и изменением свойств датчиков и промежуточных преобразователей, вплоть до полного выхода их из строя.

С учётом того обстоятельства, что оба указанных процесса являются принципиально случайными, как и текущие значения полезного сигнала, на фоне которого они наблюдаются, единственно адекватным в данном случае оказывается вероятностный (статистический) подход. Одной из основных исходных характеристик является закон распределения сигнала, на основании которого для мгновенных значений сигнала могут быть рассчитаны значения, достигаемые с определённой вероятностью. Ограничив такими значениями зону, где сигнал пребывает с вероятностью, близкой к единице, получают возможность констатировать редкие (и уже поэтому подозрительные) события. Иногда подобной системой установок по каждому из параметров и ограничивают контроль нормального хода процесса.

Нетрудно заметить, что система установок, с которыми сравниваются мгновенные значения сигналов, обладает серьёзными недостатками. Слишком узкие границы часто пересекаются самим сигналом, что может приводить к излишним неоправданным тревогам, а при слишком широких границах система теряет чувствительность и может, например, со значительным запаздыванием обнаруживать дрейф параметра. Кроме того, такая система не может обнаружить факт выхода датчика или преобразователя из строя, если текущее значение ложного сигнала осталось в пределах контролируемых границ.

От подобных недостатков можно избавиться, если сигналы и дрейфы рассматривать как случайные процессы, развивающиеся во времени, и использовать их соответствующие характеристики, например ковариационные функции. В этом случае «полезным» сигналом оказываются (относительно медленные) процессы дрейфа технологических параметров и выхода из строя преобразователей и датчиков, связанные, например, с изменением во времени коэффициентов усиления используемых элементов. Наблюдаются же эти процессы совместно с нормальными флуктуациями технологических параметров, которые теперь начинают играть роль «помехи». Задача, таким образом, вновь сводится к выделению «сигнала» на фоне «помехи» и решается совершенно аналогично задаче оптимальной предварительной фильтрации. Непременным условием в данном случае является использование физически реализуемого фильтра с тем, чтобы наметившийся дрейф и выход датчиков из строя были обнаружены с минимальным запаздыванием. Сам по себе такой факт может быть зафиксирован, например, путем сравнения текущего значения уже отфильтрованного сигнала с соответствующими порогами.

Выход из строя измерительного канала (датчика или преобразователей), приводящий к «зависанию» показаний в некоторой точке внутри допустимого диапазона, может быть обнаружен на основе определения наблюдаемого текущего показателя степени флуктуации процесса. Здесь важно лишь установить рациональный интервал повторения проверок и выбрать простой для машинной реализации расчётный показатель степени флуктуации процесса. По-видимому, достаточную исходную информацию несет ковариационная функция процесса, где ордината при нуле (дисперсия) является объективной количественной мерой степени флуктуации, а число ординат, имеющих существенный вес, предопределяет рациональный интервал «памяти» (математического) оператора, с помощью которого будет выделяться необходимая оценка. Для практической реализации на «Электронике-100» в качестве оценки удобнее использовать упрощённую взвешенную оценку модуля текущих отклонений от среднего, чтобы обойтись без операций умножения¹. Интервал накопления отсчётов, по-видимому, должен быть в 2—3 раза больше интервала корреляции сигнала при нормальном протекании процесса.

Таким образом, задача определения нарушений нормального режима решается на основе текущих значений сигналов с помощью операторов, определяемых по вероятностным характеристикам сигналов, которые имеют место при нормальном ходе процесса.

9.3. ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ СИГНАЛОВ ПРИ РАСЧЁТЕ ТЕКУЩИХ ПОКАЗАТЕЛЕЙ И УПРАВЛЯЮЩИХ ВОЗДЕЙСТВИИ

Одной из важных функций системы является расчет текущих значений некоторых обобщенных показателей технологического процесса, а также расчет управляющих воздействий, которые выдаются оператору в качестве рекомендаций на изменение режимных параметров процесса либо непосредственно реализуются в контуре прямого цифрового управления. Все подобные расчеты строятся с использованием мгновенных значений сигналов на некотором интервале наблюдения, а возможность хранения в памяти машины относительно большого объёма данных о различных параметрах позволяет осуществлять в автоматизированных системах расчет текущих значений сложных показателей, включая такие, как технологическая составляющая себестоимости, расходные коэффициенты и т. п.

Во многих случаях в расчетах одновременно участвуют переменные, измеряемые в различных точках технологического процесса, например, на его входе и выходе. При этом и информационные потоки соответствующих переменных оказываются взаимосвязанными согласно динамическому оператору объекта по тому или иному каналу, что требует специального учёта в расчётных соотношениях. Прямое сопоставление данных, отсчитанных в один и тот же момент времени из разных динамически связанных сигналов, может приводить к значительным ошибкам.

Качественное объяснение такого эффекта состоит в том, что каждое мгновенное значение выходного сигнала динамического объекта складывается под влиянием целого континуума предшествующих значений входного сигнала. Поэтому при случайном характере последнего любое значение выходного сигнала не может быть отнесено или поставлено

¹ Для каждого вида распределения средний модуль текущих изменений отклонения от математического ожидания сигнала однозначно связан с дисперсией, так что эти характеристики в данном случае равноправны. Для упрощённой реализации операции взвешивания значения весовых коэффициентов могут, например, округляться так, чтобы у каждого коэффициента оставалась единственная единица во всей разрядной сетке, и тогда умножение на этот коэффициент сводится просто к единственному сдвигу множимого. Другой путь может лежать в экспоненциальной аппроксимации весовой функции усредняющего временного оператора.

² Следует отметить, что неслучайные «сигналы» (все развитие которых известно заранее) не несут никакой информации и значения

в соответствие какому-либо единственному значению входного сигнала.

Количественный анализ ошибок, возникающих при упрощенном подходе, не учитывающем этого обстоятельства для расчета текущих технико-экономических показателей технологического процесса, проведен в [31]. Результаты анализа показывают, что относительная погрешность от пренебрежения динамическими связями может достигать 200%.

Для другого важного случая, когда сигналы на входе и выходе динамического объекта используются для получения экспериментальных моделей на основании данных так называемого «пассивного эксперимента», возможные погрешности, возникающие в результате пренебрежения динамическими связями, становятся ещё более существенными. Величина относительной ошибки за счет смещенности получаемых оценок здесь может исчисляться тысячами процентов, а в отдельных случаях смещенность достигает такой величины, что приводит к ошибке в знаке коэффициентов модели (см. [32]).

Таким образом, учёт динамических связей является принципиальным требованием корректных расчетов любых показателей в системе, если эти расчёты проводятся на основании текущих значений сигналов, получаемых от динамических объектов. Естественный путь решения этой задачи состоит в преобразовании входных сигналов в соответствии с динамическим оператором объекта (или «приведение» входного сигнала к выходному [33]). Для линейных (и линеаризуемых) объектов в расчётных соотношениях вместо мгновенных значений входных сигналов должны использоваться динамически скорректированные сигналы; одной из наиболее универсальных форм коррекции является прямое использование свертки [31]:

$$x'(t) = \int_{-\infty}^t x(\tau)h(t-\tau)d\tau \quad (9.14)$$

или в случае дискретной реализации

$$x'[i] = \sum_{j=i-J}^i x[j]h[i-j] \quad (9.14a)$$

Здесь $h(t)$ — динамический оператор (весовая функция) объекта по каналу «вход — выход» (от рассматриваемого входного параметра к тому из выходных параметров, который участвует вместе с данным входным в расчётных соотношениях); $J+1$ — число ординат дискретной весовой функции, используемой при машинных расчётах.

Естественно, при динамической коррекции вместо единственного значения входного сигнала каждый раз используются и J его прошлых значений, которые к этому моменту должны ещё храниться в памяти машины. Кроме того, по каждому каналу необходимо иметь $J+1$ ординату соответствующей весовой функции.

В связи с этим в структурных схемах расчётных соотношений, выполняемых в АСУ, должны обязательно присутствовать блоки динамической коррекции сигналов и блоки, организующие сдвиг прошлых массивов сигналов при получении и запоминании их новых текущих значений. Соответственно при распределении памяти должно отводиться место для хранения ординат весовых функций по необходимым каналам и место для массивов соответствующих входных сигналов.

Самостоятельный интерес представляет определение вида весовой функции для формирования управляющего сигнала в контуре прямого цифрового управления. Задача сводится фактически к выбору оптимальной весовой функции звена обратной связи в замкнутой дискретно-непрерывной системе. Даже для чисто дискретной системы² не удается получить в явном виде аналитического выражения для расчета ординат оптимальной весовой функции (как это имело место, например, при синтезе оптимального цифрового фильтра). Здесь, однако, возможна численная оценка эффективности того или иного решения, т. е. фактически его количественный анализ. Это значит, что есть путь отбора лучших вариантов или синтеза некоторой оптимальной - (по крайней мере, в локальной области) весовой функции за счет организации вычислительной процедуры, которая обеспечивает улучшение критерия качества на каждом шаге.

Возможный путь количественной оценки эффективности использования некоторого фильтра, формирующего сигнал управления, можно проследить на следующем примере.

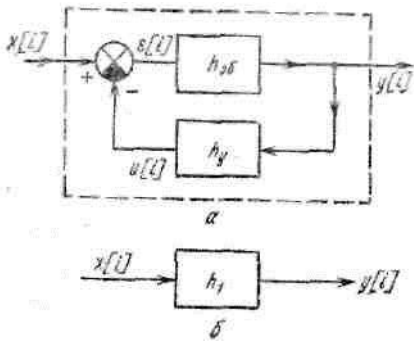


Рис. 9.5. Схема для анализа эффективности различных способов формирования управляющих сигналов; а - схема замещения объекта, охваченного обратной связью; б — сигналы на входе и выходе эквивалентного звена без обратной связи.

Следовательно, достаточно положить

Пусть формирующий фильтр используется в схеме стабилизации значений выходного параметра линейного объекта, модель которого полагается дискретной (рис. 9.5). За оценку качества регулирования принимается дисперсия флуктуации на выходе объекта.

Если допустить, что сигналы $x[i]$ и $y[i]$ на схемах (рис. 9.5, а и б) соответственно равны, т. е. для схемы рис. 9.5, а существует эквивалентное линейное звено с весовой функцией h_1 , то дисперсия сигнала $y[i]$ определится как

$$\sigma_y^2 = \sum_p \sum_q h_1[p]h_1[q]K_{xx}[p-q] \quad (9.15)$$

Здесь $K_{xx}[j]$ — ковариационная функция входного сигнала.

Задача, таким образом, состоит в определении ординат весовой функции h_1 через параметры весовых функций $h_{0\delta}$ и h_y . Такая задача может быть решена, если учесть, что весовая функция дискретного объекта есть не что иное, как реакция на единичный импульс, приложенный ко входу объекта в момент $i = 0$, если ранее объект находился в состоянии покоя.

любых показателей при воздействии на объект таких сигналов также могли бы быть экстраполированы для любого момента времени до $+\infty$. Понятно поэтому, что одной из основных целей любой системы контроля и АСУ является успешная работа именно в условиях случайных входных воздействий.

Следует отметить, что упрощенные методы динамической коррекции (в частности, сдвиг входного сигнала на время, соответствующее максимуму взаимной корреляционной функции между входным и выходным сигналами [34]) для широкого круга соотношений параметров сигнала и объекта оказываются неэффективными. В этом нетрудно убедиться, например, путем моделирования для аperiodических звеньев первого порядка

Анализируемой обычно с помощью аппарата Z- преобразования.

$$x[i] = \begin{cases} 1 & n_{pi} = 0 \\ 0 & n_{pi} \neq 0 \end{cases}, \quad (9.16)$$

$$y[i] \Big|_{i < 0} = \varepsilon[i] \Big|_{i < 0} = u[i] \Big|_{i < 0} = 0,$$

а затем проследить за развитием сигнала $y[i]$, который численно характеризует ординаты $h_1[i]$. При этом в силу очевидного соотношения

$$\varepsilon[i] = x[i] - u[i] \quad (9.17)$$

получаем

$$\varepsilon[0] = 1 - u[0], \quad \varepsilon[i] \Big|_{i \neq 0} = -u[i] \quad (9.18)$$

Учитывая, что для линейных физически реализуемых звеньев

$$y[i] = \sum_{j=0}^J h_{o\delta}[j] \varepsilon[i-j] \quad (9.19)$$

и

$$u[i] = \sum_{p=0}^P h_y[p] y[i-p] \quad (9.20)$$

где J и P — протяженность весовых функций объекта и формирующего фильтра соответственно, можно получить следующее соотношение для расчета значений $\varepsilon[q]$, $q = 0, 1, \dots$ (а на их основании и $y[i]$ согласно (9.19)):

$$\varepsilon[0] = \frac{1}{1 + h_y[0]h_{o\delta}[0]}, \quad (9.21)$$

$$\varepsilon[q] = -\frac{1}{1 + h_y[0]h_{o\delta}[0]} \sum_{p=0}^{q-1} \varepsilon[p] h_{ocл}[q-p], \quad q=1, 2, \dots$$

Здесь $h_{ocл}[i]$ — весовая функция последовательно соединённых звеньев $h_{o\delta}$ и h_y :

$$h_{ocл}[l] = \sum_{i=0}^P h_y[i] h_{o\delta}[l-i] \quad (9.22)$$

Основная трудность последующего анализа обусловлена тем, что выражение для $\varepsilon[q]$ является рекурсивным, так что его раскрытие приводит к появлению нелинейных (степенных) функций от ординат $h_y[i]$. Поэтому обозримый до конца путь анализа и сводился к привлечению численных методов с привлечением аппарата Z-преобразования или следующей схемы.

Для заданных $K_{xx}[i]$ и $h_{o\delta}[j]$ выбирается некоторое исходное приближение ${}^{(0)}h_y[j]$, затем по (9.22) рассчитывается ${}^{(0)}h_{ocл}[l]$ и ряд значений ошибки $\varepsilon[q]$ в соответствии с (9.21). Использование (9.19) позволяет определить ${}^{(0)}y[i] = {}^{(0)}h_1[i]$, и, наконец, (9.15) дает количественную оценку данного приближения. Для некоторого другого ${}^{(1)}h_y[j] = {}^{(0)}h_y[j] + {}^{(1)}\Delta h[j]$ весь этот путь может быть повторен, а разность полученных оценок (псевдоградиент) может применяться для выбора направления следующих приращений $\Delta h[j]$. Таким образом можно достичь ближайшего локального экстремума, после чего вся процедура может быть повторена для нового исходного приближения.

Поиск оптимального формирующего фильтра является одной из задач, обычно решаемых на больших ЭВМ, где легко достигается значительная точность итеративных вычислений. Полученные значения ординат фильтра затем могут быть использованы в малой управляющей машине для формирования сигнала управления на основании текущего и прошлого значений отклонений выходного параметра объекта от требуемого значения.

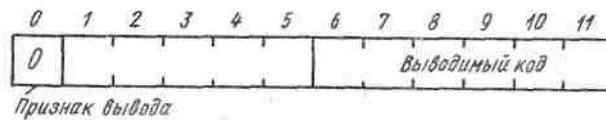
Важно отметить, что во всех рассмотренных в настоящем параграфе случаях требуется располагать весовой функцией объекта $h_{o\delta}$. О способе её получения по данным нормального функционирования объекта см. в гл. 10.

Рассчитанные значения управляющих воздействий либо обобщенных показателей выдаются из машины чаще всего в виде кодов через выходной коммутатор дискретных сигналов. В случае использования «Радуги» вывод осуществляется 6-разрядными кодами из младших разрядов НР (СМ) через промежуточный регистр информации (РИ). Каждый из выходных коммутаторов имеет четыре канала, причем первый, второй и частично третий каналы снабжены выходными триггерами, образующими 16-разрядный регистр памяти данного коммутатора. Последовательность команд, управляющих выводом информации через «Радугу», такова:

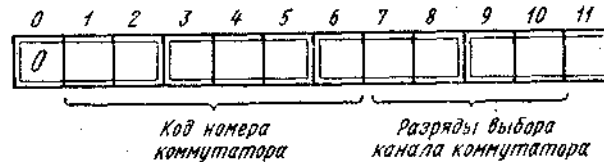
Информационное слово вывода → в НР (СМ)
6311
7200
Слово управления выходными коммутаторами в → НР (СМ)

6314
6312

Информационное слово вывода имеет формат:



Слово управления выходными коммутаторами имеет такой формат:



По команде 6311 БУК настраивается на режим вывода информации (в соответствии с нулевым содержимым 0-го разряда НР (СМ)) и переписывает выводимый код в свой регистр информации (РИ). В слове управления выходными коммутаторами разряды 7—10 закреплены каждый за своим каналом: разряд 7 — за 1-м, 8 — за 2-м; 9 — за 3-м; 10 — за 4-м каналом.

Команды 6314 и 6312 управляют передачей информации из РИ в регистр памяти коммутатора (по команде 6314 происходит лишь логическое сложение содержимого РИ со старым содержимым регистра памяти коммутатора; команда 6312 обнуляет те разряды регистра памяти коммутатора, для которых соответствующие разряды РИ установлены в 0, обеспечивая, таким образом, фиксирование кода РИ в регистре памяти коммутатора).

Наличие регистра памяти у выходного коммутатора позволяет запоминать и удерживать выводимые коды для части внешних устройств-потребителей, не имеющих собственных регистров памяти, до выдачи нового выходного кода. Внешние устройства, управляемые импульсами от машины, могут подключаться к 3-му и 4-му каналам выходного коммутатора либо формировать выдаваемые импульсы при смене потенциалов в регистре памяти коммутатора, когда информация выводится по 1—3 каналам.

Старшинство разрядов регистра памяти коммутатора определяется (назначается) при составлении общей (монтажной) схемы аппаратной части системы, что затем непосредственно учитывается при разработке конкретных программ вывода информации.

9.4. ОБ ОБЩЕЙ КОМПОНОВКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ

Как видно из изложенного материала, создание общего математического и программного обеспечения каждой автоматизированной системы управления технологическими процессами является большой и сложной задачей. Результаты предыдущих разработок, как правило, могут быть лишь частично использованы в новых системах, работающих в своих конкретных условиях. В связи с этим большое значение приобретают методические вопросы таких разработок — от получения исходных данных на проектирование до отладки и внедрения новых программ.

В основе работы большинства автоматизированных систем управления технологическими процессами лежит решение того или иного круга задач из числа приведенных в начале данной главы. Поэтому одним из первоочередных этапов проектирования является четкое определение состава задач, необходимых исходных данных и состава окончательных выдаваемых результатов. Следующий этап предполагает выбор алгоритмов решения каждой из задач (с точностью до укрупненных расчётных соотношений), а также уточнение требований к операционной системе (или возможности использования существующей операционной системы). После этого должны быть сформулированы требования к средствам получения и предварительной обработки необходимых сигналов, а также уточнены параметры программ динамической коррекции входных информационных потоков.

Решение этой части задач опирается в основном на исходные данные, получаемые проектировщиком математического (программного) обеспечения от ведущего специалиста по системе в целом. Целесообразно уточнить (и по возможности унифицировать) состав вопросов в техническом задании на проектирование программ (или математического обеспечения в целом). Примерный круг подобных вопросов¹ должен освещать:

1. Назначение и функции создаваемой АСУТП, точную формулировку критерия эффективности системы управления.
2. Сводную функциональную схему объекта управления (технологического процесса и оборудования) и системы управления.

На схеме должны быть указаны:

- основные технологические операции, материальные и энергетические потоки, а также места контроля параметров и точки приложения управляющих воздействий (для объекта);
- все основные устройства: датчики, преобразователи и устройства сопряжения, ЦВМ, устройства отображения информации, пути операторов, исполнительные механизмы и т. п. (для системы управления).

Функциональная схема должна сопровождаться кратким описанием.

3. Полный перечень задач, возлагаемых на ЦВМ.
4. Таблицу контролируемых параметров, содержащую наименование параметра, его номинальное или среднее значение, диапазон изменения, тип выходного сигнала датчика (аналоговое напряжение, код, частотный сигнал), характеристики

¹ Требования к составлению технических заданий на математическое обеспечение разработаны совместно с Ю. М. Петровым.

точности, функциональную зависимость выхода датчика от значений параметров

5. Временные характеристики поступления (запросов) информации на ЦВМ и выдачи результатов обработки (решений); требуемую точность результатов.

6. Таблицу выходных управляющих воздействий и выдачи выходной информации (исполнительные механизмы с кодовым входом, с шаговыми двигателями, с тиристорным управлением и т. п.).

7. Описание порядка ввода системы в действие, порядка отладки и наращивания программ, перспективы расширения системы

8. Для неотработанных алгоритмов и слабо изученных объектов дополнительное задание на постановку исследовательской работы.

Для разработки конкретных программ и выбора конфигурации ЭВМ важно также иметь данные об интервалах «памяти» объекта по основным каналам, требующим динамической коррекции. Требования к памяти ЭВМ существенно зависят также от объема и форматов печатных сообщений.

Сведения, полученные в ответ на такие вопросы, позволяют приступить к настройке основных блоков операционной системы для проектируемой АСУ ТП и подбору модульных блоков программ из числа ранее разработанных. Важно отметить, что этот процесс может быть автоматизирован (возможно, с привлечением больших ЭВМ), если формализовать сам способ получения исходного описания требований проектируемой системы.

Идеальным случаем можно, по-видимому, считать такой, когда все исходное описание условий функционирования будущей системы составлено на обычном общечеловеческом языке и по нему машина (естественно, в результате работы «высокоинтеллектуальной» программы) выдает распечатку и перфоленту программы, которая готова для непосредственного использования в управляющей малой ЭВМ системы, составленной в соответствии с конфигурацией, рекомендуемой самой машиной. Идеальность такого случая настолько очевидна, что он давно бы должен был быть реализован, если бы его техническое осуществление не вызывало огромных и разноплановых затруднений¹, хотя принципиально весь путь решения подобной задачи просматривается (и можно назвать его узловые моменты).

Предположим теперь, что первоочередной этап ввода и предварительной ассимиляции исходного описания пройден² (или обойден некоторым упрощенным путем). Что бы следовало получить в качестве результата? По-видимому, это должна быть развернутая таблица ответов на вопросы, подобные приведенным выше, которую можно рассматривать как некоторую промежуточную структуру описания требований к системе. С другой стороны, машина должна располагать заранее заготовленными программными модулями для решения достаточно широкого круга возможных задач, причем эти модули должны иметь возможность настройки входящих в них параметров и некоторую свободу перемещения в памяти при компоновке общей программы. Тогда специальная компоновочная программа могла бы превращать разрозненные модульные заготовки в программу системы в соответствии с запросами, содержащимися в промежуточной структуре описания требований к системе.

Двумя следующими узловыми вопросами являются такие:

1) Как должна быть построена компоновочная программа?

2) Каковы принципиальные вопросы разработки и реализации программных модулей-заготовок?

Один из возможных путей разрешения этих вопросов (одновременно обеспечивающий достаточно эффективное получение промежуточной структуры описания требований к системе с привлечением интеллекта заказчика) состоит в применении диалоговой системы ответов на вопросы, задаваемые машиной, программа которой подготавливает тот или иной набор очередных общих или уточняющих вопросов в соответствии с получаемыми ответами. Тогда каждому вопросу или их последовательности машина может сама ставить в соответствие (например, по номеру вопроса) отдельный модуль, настраиваемый полученными ответами, и прикомпоновывать его к уже сформированной последовательности модулей. При этом весь процесс компиляции, очевидно, целесообразно доводить лишь до уровня мнемокодов (ассемблера) малой машины, с тем, чтобы переложить завершающую стадию составления программы на транслятор ассемблера, обеспечивая заодно простоту компоновки и перемещения отдельных модулей. Сами модули при такой структуре компоновочной программы также должны быть написаны на ассемблере.

Вопрос об укрупненности, числе настроек и функциях отдельных модулей, очевидно, связан с глубиной алгоритмической и программной проработки общих вопросов построения автоматизированной системы и особенностей её реализации в условиях действующих производств. По-видимому, число и состав модулей со временем могут претерпевать заметные изменения, а компоновочная система должна иметь возможность легко приспосабливаться к таким изменениям (и расширениям).

Нетрудно видеть, что даже простейшая система программ такого типа заметно продвинула бы вперед решение задач проектирования программного обеспечения АСУ ТП, К настоящему времени, по крайней мере, для «Электроники-100», идет лишь процесс разработки и накопления отдельных программных модулей, что уже само по себе является важным шагом на этом пути. Функции компоновочной программы пока что вынуждены выполнять сами программисты, и очень важно, чтобы накапливаемый опыт становился достоянием широкого круга пользователей (см. заключение).

Исключительно важным вопросом, от решения которого во многом зависит эффективность применения ЭВМ в АСУ ТП и эффективность самих АСУ ТП, является разработка алгоритмического обеспечения системы. Часть этих вопросов рассматривалась в данной главе; другим вопросам этого круга и особенностям их реализации на «Электронике-100» посвящена следующая глава.

10. ОПЕРАТИВНОЕ ИЗУЧЕНИЕ СВОЙСТВ ОБЪЕКТА ПО ДАННЫМ НОРМАЛЬНОГО ФУНКЦИОНИРОВАНИЯ

Возможность изучения свойств конкретных технологических объектов по данным их нормального функционирования является одним из наиболее существенных результатов применения вычислительной техники при создании систем управления и при введении их в действие. Такая возможность важна и для целенаправленной лабораторной отработки новых процессов и других научных исследований. На этом пути, однако, есть много сложностей, и только их совместный учет позволяет наметить основные этапы решения задачи и подойти к решению проблемы в целом.

¹ См., например, [35].

² Этот этап смыслового анализа вводимых сообщений сам по себе влияет на разработку целой специальной системы программ или разработку специального языка программирования (см. [35—37]).

10.1. ПОСТАНОВКА ЗАДАЧИ

В достаточно общем виде задачу можно сформулировать следующим образом.

На действующем объекте ведется регистрация значений входных и выходных параметров. По полученным записям требуется оценить характеристики объекта по каждому из каналов «вход — выход». Для количественного решения такой общей задачи необходимо ввести некоторые понятия и определения.

Измеряемые значения параметров будем называть сигналами; недоступные измерению воздействия, которые искажают истинное значение сигналов, — помехами. Под возмущениями понимаются любые воздействия, приводящие к изменению входных или выходных параметров объекта.

В режиме нормального функционирования объекта значения параметров поддерживаются на некоторых номинальных уровнях, при этом сигналы имеют вид флуктуации около номинальных значений и эти флуктуации носят шумообразный случайный характер. Шумообразный характер носят и помехи, причем их постоянная составляющая может быть исключена из рассмотрения как обычная систематическая погрешность измерений.

Выходные сигналы формируются из входных согласно воздействию на последние некоторого оператора объекта, обычно динамического, благодаря чему каждое отдельное воздействие со стороны входа проходит на выход не тотчас же, а лишь в виде некоторого переходного процесса. И хотя сам оператор объекта может быть детерминированным, выходной сигнал всегда носит случайный характер как благодаря случайности входных возмущений, обуславливающих непрерывный переходной режим от одного случайного уровня к другому — по каждому каналу «вход — выход», так и благодаря дополнительному влиянию помех и неучтенных возмущений.

В общем случае характеристики объекта могут описываться системой нелинейных дифференциальных уравнений. Однако в режиме нормального функционирования диапазон флуктуации параметров объекта сравнительно невелик, и поэтому получаемые данные пригодны лишь для локального описания характеристик объекта в окрестности рабочей точки. Это же обстоятельство позволяет ограничиться изучением линейного приближения модели объекта в виде весовой функции и статического коэффициента передачи в рабочей точке (хотя здесь и возможны некоторые расширения используемого подхода). Поскольку форма оператора связи «вход — выход» в общем случае неизвестна, достаточно универсальным является лишь его представление оператором произвольной структуры (или так называемым «черным ящиком»), сам вид которого уточняется в процессе решения.

Случайный характер сигналов и помех и наличие динамических связей в исследуемых объектах требуют для адекватности всех исследований последовательного применения вероятностного описания на основе теории случайных функций. В первую очередь это касается используемых характеристик сигналов и оценок точности результатов, получаемых в условиях ограниченных интервалов наблюдения. Предполагается гипотетическое наличие множества (или ансамбля) реализаций процесса стационарного и эргодического. При этом такие характеристики сигналов, как математическое ожидание, ковариационная функция и т. п., определяются как характеристики, вычисляемые усреднением по множеству, а среднее значение сигнала, его корреляционная функция — усреднением по времени на доступной длине отдельной частной реализации. Последние характеристики являются оценками первых и сходятся к ним в среднеквадратичном смысле при неограниченном увеличении интервала наблюдения в силу принятой гипотезы об эргодичности процесса, что позволяет применять их для прогнозирования развития процесса и на любом другом отрезке времени. В то же время среднее значение сигнала и его корреляционная функция, вычисленные по ограниченной реализации случайного процесса, сами несут элемент случайности (т. е. несколько отличаются как от характеристик ансамбля, так и от любых других аналогичных оценок, вычисленных по другому отрезку той же реализации такой же длины).

Аналогичные замечания необходимо сделать и относительно взаимных ковариационных и корреляционных функций.

Для определения состава операций, которые в дальнейшем потребуются реализовать на машине, необходимо проследить путь возможного решения задачи в относительно простом случае, когда объект линеен и имеет один вход и один выход. В этом случае истинные приращения выходного сигнала $y(t)$ связаны с приращениями входного сигнала $x(t)$ через выражение свертки

$$y(t) = K \int_{-\infty}^{\infty} h(\mu) x(t - \mu) d\mu, \quad (10.1)$$

где K — статический коэффициент передачи в рабочей точке; $h(t)$ — весовая функция объекта, удовлетворяющая условию нормировки¹:

$$\int_{-\infty}^{\infty} h(t) dt = 1 \quad (10.2)$$

В реальных условиях вместо сигнала $y(t)$ возможно измерение (некоторого другого) сигнала, искаженного в простейшем случае аддитивной помехой $n(t)$:

$$z(t) = y(t) + n(t) \quad (10.3)$$

Задача состоит в том, чтобы найти оценку весовой функции $h(t)$ и оценку коэффициента передачи в рабочей точке K , располагая ограниченными частными реализациями сигналов $x(t)$ и $z(t)$, причем в силу условия нормировки (10.2) достаточно искать оценку функции $h_1(t)$:

$$h_1(t) = Kh(t) \quad (10.4)$$

Полученная оценка должна, очевидно, обеспечивать наилучшее в некотором смысле предсказание выходного сигнала объекта по известному сигналу при структуре связи, аналогичной (10.1):

$$\hat{y}(t) = \int_{-\infty}^{\infty} \hat{h}_1(\theta) x(t - \theta) d\theta \quad (10.5)$$

Из условия минимизации дисперсии ошибки предсказания выходного сигнала (с использованием доступных измерению сигналов и их характеристик по множеству) выражение, которое является исходным для расчётов наилучшей оценки $h_1(t)$, может

¹ Важно отметить, что (10.1) не только математически формализует связь сигналов на входе и выходе динамического объекта, но и отражает физическую сущность формирования каждого мгновенного значения выходного сигнала путем «взвешивания» значений входного сигнала на интервале «памяти» объекта в соответствии с видом весовой функции $h(t)$. Так как физически реализуемая система способна реагировать лишь на текущие и прошлые входные воздействия, фактические пределы интегрирования в (10.1), (10.2) имеют вид $[0, T_n]$, где T_n — интервал «памяти», хотя формально эти пределы могут быть указаны как $[-\infty, +\infty]$.

быть получено в виде известного соотношения¹.

$$K_{xz}(\tau) - \int_{\theta} \hat{h}_1(\theta) K_{xx}(\tau - \theta) d\theta = 0 \quad (10.6)$$

где $K_{xx}(\tau)$ — собственная ковариационная функция входного сигнала; $K_{xz}(\tau)$ — взаимная ковариационная функция сигналов на входе и выходе объекта.

Поскольку последние характеристики (ансамбля) неизвестны, в практических расчетах используют корреляционные функции, вычисляемые по доступной длине частных реализаций сигналов. Особенности необходимого при этом подхода будут рассмотрены ниже. Сейчас важно отметить, что для идентификации характеристик объекта требуется реализовать расчет корреляционных функций сигналов и численное решение уравнения (10.6), осуществив переход к его дискретной форме, а также к дискретной форме других необходимых соотношений.

10.2. ПОДХОД К РЕШЕНИЮ ЗАДАЧИ В УСЛОВИЯХ ОГРАНИЧЕННОГО ОБЪЁМА ПАМЯТИ

Как следует из § 10.1, для решения задачи идентификации характеристик объекта по данным его нормального функционирования требуется осуществить сбор и накопление информации о текущих значениях сигналов, расчет их статистических характеристик и решение системы линейных уравнений (к которой сводится дискретный аналог соотношения (10.6)). Следует ожидать, что повышение объёма исходных данных улучшит точность результатов, как это бывает в большинстве статистических задач (количественное подтверждение этого положения будет приведено в § 10.4). Наряду с этим размещение исходных данных и программ их обработки для решения рассматриваемой задачи, очевидно, не должно препятствовать решению основных задач системы, особенно в тех случаях, когда идентификация характеристик объекта имеет уже лишь уточняющий оттенок.

Одним из возможных путей успешного решения задачи в условиях ограниченного объёма памяти является разделение всей задачи на независимые участки и их последовательное решение на одной и той же машине либо даже на нескольких различных машинах.

Достаточно удобным и практически единственным в условиях минимального комплекта ЭВМ носителем промежуточной информации, выполняющим одновременно роль долговременного запоминающего устройства (ДЗУ), может служить перфолента. Остается позаботиться лишь об организации соответствующей системы программ как для решения задачи в целом, так и для занесения информации в такое ДЗУ и считывания из него.

На этапе сбора информации в состав программного обеспечения действующей системы должен быть включен блок немедленного вывода на перфоленту результатов измерения текущих значений всех сигналов, если быстродействие перфоратора оказывается достаточным. В ином случае должен работать блок предварительного накопления допустимого объёма замеров в буферной зоне с последующей выдачей их на перфорацию. В целях простого опознания и контроля промежуточного массива целесообразно избрать алфавитно-цифровой формат представления промежуточных данных с использованием десятичной системы счисления, указанием идентификаторов переменных перед каждым их значением в кодах «Консула-254». Чтобы с полученной перфоленты можно было непосредственно распечатывать на пишущей машинке в автономном режиме таблицу, отражающую развитие процесса во времени, целесообразно также помимо значений сигналов включить в состав выводимой информации отсчеты текущего значения времени (номер временного интервала, играющий роль одного из индексов при дискретном представлении сигналов) и служебные коды пробелов и возврата каретки. Такая форма представления и вывода данных является удобной ещё и в том отношении, что позволяет легко получать в таком же формате дополнительные данные о процессе, подключая в автономном режиме перфоратор к «Консулу-254» и распечатывая предварительно составленную таблицу значений переменных, измерения которых нельзя было по каким-то причинам непосредственно вводить в машину. Некоторая избыточность выводимой информации окупается, таким образом, простотой опознания каждого измерения в массиве и локализацией отдельных возможных сбоев, весьма существенной при дальнейших статистических расчетах.

Структурная схема программы сбора и регистрации исходных данных приведена на рис. 10.1.

Последующая обработка накопленных данных начинается с использования программы, обеспечивающей выбор с перфоленты, перевод 10→2 и засылку по стандартным адресам нужных для очередного шага исследования участков исходного массива данных. Очевидно, поиск нужных элементов массива должен осуществляться с помощью идентификаторов; сами идентификаторы, а также служебные коды (пробелов, возврата каретки и т. п.) и неиспользуемые элементы исходного массива должны игнорироваться. Эта программа должна иметь настройки для занесения идентификаторов одной или нескольких переменных, участвующих в последующей обработке.

Структурная схема подобной программы показана на рис. 10.2.

После работы такой вспомогательной программы выборки из ДЗУ любая последующая программа может вводиться на её место и начинать немедленно использовать подготовленные ей исходные данные. По окончании работы очередного участка программы новые промежуточные результаты вновь могут быть выведены на перфоленту вместе со своими идентификаторами и т. д. Важно отметить, что дополнительные возможности открывает применение многопроходного режима использования данных, запасенных на перфоленте, и построение основных программ обработки сигналов на основе рекуррентных соотношений, позволяющих с учётом быстродействия «Электроники-100» значительную часть вычислений осуществлять тотчас после ввода очередного отсчёта — за время, затрачиваемое вводным устройством на получение следующего отсчёта.

¹ Называемого иногда «уравнением типа Винера—Хопфа».

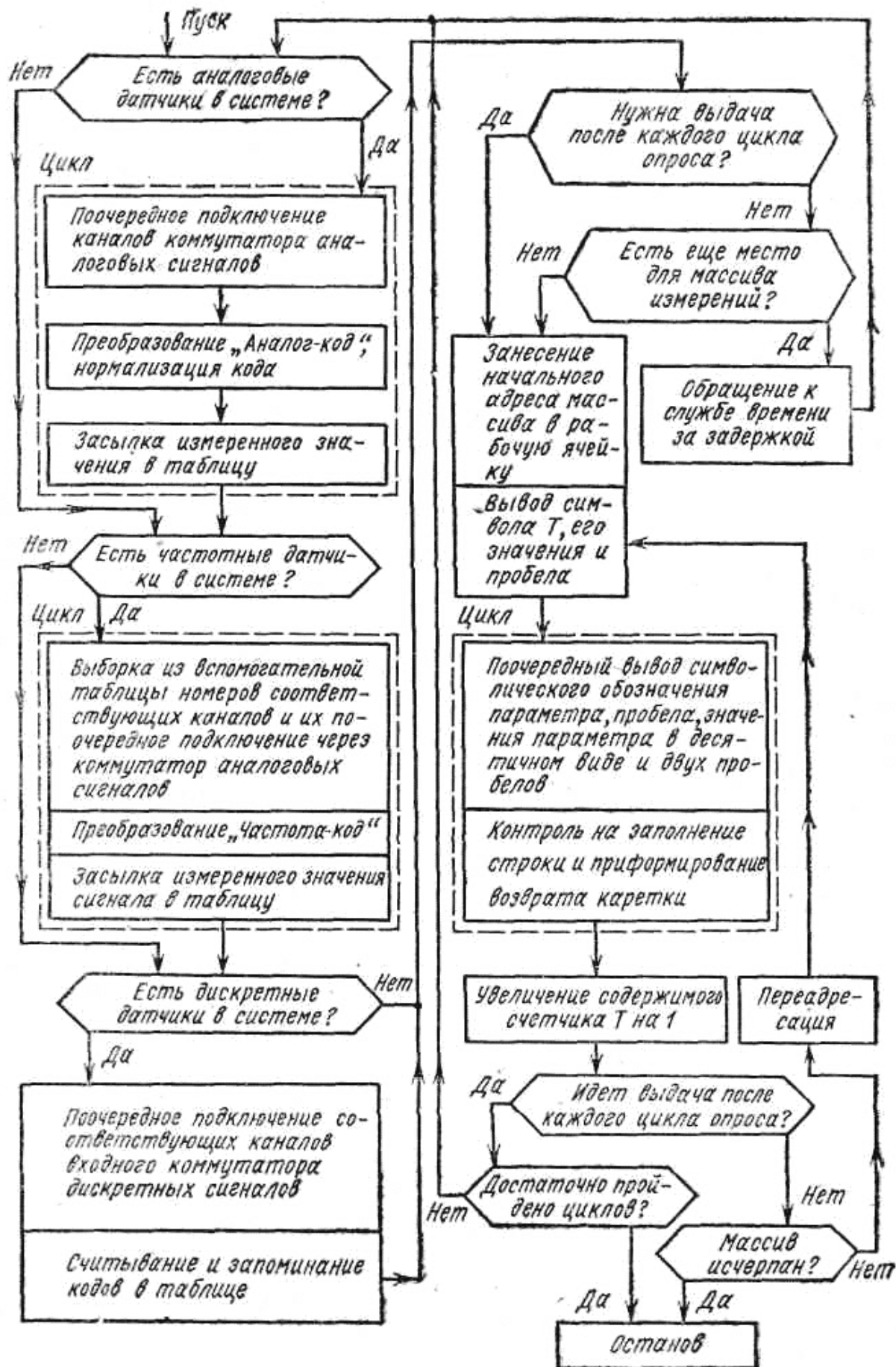


Рис. 10.1. Структурная схема опроса и регистрации параметров

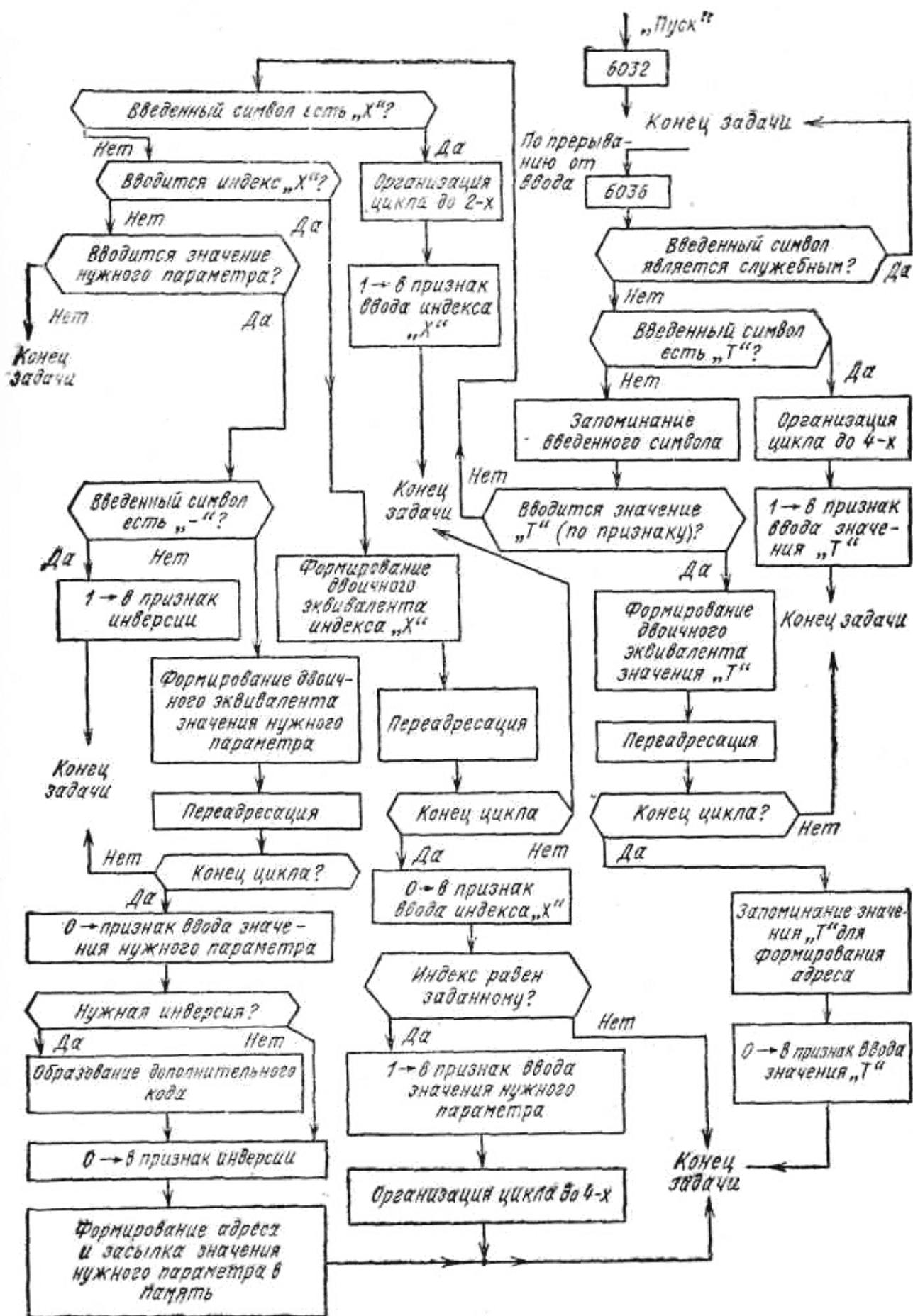


Рис. 10.2. Структурная схема блока выборки из ДЗУ на перфоленте массива измерений нужного параметра.

10.3. ОПРЕДЕЛЕНИЕ ХАРАКТЕРИСТИК СИГНАЛОВ

Наиболее употребительными характеристиками сигналов являются их средние значения и корреляционные функции, определенный интерес представляют также гистограммы мгновенных значений сигналов как оценка плотности вероятности закона распределения амплитуд. Специфика расчёта этих характеристик с использованием «Электроники-100» ограничивается вопросами масштабирования переменных и сохранения достаточной точности результата в условиях малой разрядной сетки машины. Эти вопросы в основном рассмотрены в гл. 2 и 4, здесь следует лишь отметить, что для задач АСУ ТП значения входных и выходных переменных измеряются чаще всего с помощью датчиков ограниченной точности. Поэтому точности входного аналого-цифрового преобразователя «Электроники-100» (лучше, чем 0,1%) и соответственно одной ячейки памяти в большинстве случаев оказывается достаточно для представления отдельных измерений сигналов в машине как чисел в стандартном представлении машины.

На выбор параметров рабочих программ существенное влияние оказывает не только разрядность исходных данных, но и длина реализации. Точность статистических расчетов возрастает с увеличением объема исходных данных, однако этот рост весьма заметен, когда число замеров составляет десятки и сотни, и заметно замедляется, когда замеры начинают исчисляться тысячами.

Таким образом, в качестве исходных условий при разработке программ статистического анализа сигналов на «Электронике-100» рационально принять, что значение каждого замера располагается в отдельной ячейке и имеет формат, принятый для представления чисел в машине, а объем реализации (число отсчетов) не превышает объема одного куба МОЗУ, т. е. 4096¹.

Среднее значение сигнала x_{cp} определяется как

$$x_{cp} = \frac{1}{N} \sum_{i=1}^N x_i, \quad (10.7)$$

где i — текущий номер отсчета; N — объем выборки; x_i — мгновенное значение сигнала.

Если принять последовательность вычислений в точном соответствии с (10.7), (т. е. сначала определение суммы всех отсчетов, а затем деление на N), то очевидно, что результат промежуточных вычислений в общем случае не укладывается в разрядную сетку одной ячейки. Однако зная, что максимальный объем выборки не может превышать 4096, нетрудно определить, что максимальное увеличение длины разрядной сетки для суммы не превышает 12 разрядов и весь результат, таким образом, всегда укладывается в две ячейки².

Подсчет суммы известного количества чисел легко осуществляется в цикле; единственной особенностью здесь является организация сложения знакопеременного (в общем случае) числа, занимающего одну ячейку, с промежуточным результатом, занимающим две ячейки. Такая операция выполняется в два приема — сначала производится сложение очередного числа с младшими разрядами результата, а затем учёт единиц переполнения и значащих разрядов (отрицательных) слагаемых в старших разрядах результата (так как результат занимает две ячейки, можно принять, что и исходные данные имеют 24-разрядную сетку; при этом по-прежнему только младшие 12 разрядов несут полезную информацию, и для положительных чисел старшие 12 разрядов всегда содержат нули, а для отрицательных чисел, представляемых своим дополнительным кодом, который теперь распространяется на все 24 разряда, старшие 12 разрядов всегда содержат единицы).

Полученную сумму необходимо разделить на N . Если $N < 2048$, то для деления можно непосредственно использовать стандартную подпрограмму деления числа в двух ячейках на число, занимающее одну ячейку. В противном случае простейший путь состоит в предварительном уменьшении N вдвое с соответствующим уменьшением вдвое и получаемого после деления результата (при этом для нечётных N вносится дополнительная погрешность, не превышающая 0,05%, т. е. все 11 значащих разрядов мантииссы частного оказываются ещё верными). При более строгом подходе на этапе сбора исходных данных либо при подготовке цикла суммирования целесообразно вводить дополнительное ограничение на чётность N .

Рекуррентная схема расчёта оценки среднего с немедленным использованием каждого введенного отсчёта становится очевидной из следующих соотношений.

Для $(m + 1)$ отсчётов оценку среднего можно представить в виде

$$(x_{cp})_{m+1} = \frac{1}{m+1} \sum_{i=1}^{m+1} x_i = \frac{1}{m+1} \left(\sum_{i=1}^m x_i + x_{m+1} \right) = \frac{m}{m+1} \frac{1}{m} \sum_{i=1}^m x_i + \frac{1}{m+1} x_{m+1} = \frac{m}{m+1} (x_{cp})_m + \frac{1}{m+1} x_{m+1} \quad (10.8)$$

или

$$(x_{cp})_{m+1} = \frac{1}{m+1} [m(x_{cp})_m + x_{m+1}]. \quad (10.8a)$$

Смысл выражения (10.8) сводится к тому, что каждая последующая оценка среднего значения сигнала получается из предыдущей, скорректированной в соответствии с числом использованных отсчетов, путем сложения с очередным текущим значением сигнала, деленным на номер его отсчета.

¹ Следует отметить, что последнее число не является принципиальным ограничением, так как несмещенность и состоятельность большинства рассчитываемых характеристик позволяет при больших объемах реализации повышать точность характеристик за счёт обработки реализаций по частям, с дополнительным усреднением уже самих характеристик.

² Это следует из такой цепочки неравенств: $\sum_{i=1}^N x_i \leq \sum_{i=1}^N |x_i| \leq \sum_{i=1}^N |x|_{\max} = N |x|_{\max}$ или $\sum_{i=1}^N x_i \leq N |x|_{\max}$. Соответственно $\log_2 \sum_{i=1}^N x_i \leq \log_2 N + \log_2 |x|_{\max}$, а так как любое из исходных чисел, в том числе и $|x|_{\max}$, по условию занимает целую ячейку, то максимальное удлинение разрядной сетки для $\sum_{i=1}^N$ определяется первым слагаемым в правой части последнего выражения и в силу условия $N < 4096$ не превышает 12 двоичных разрядов.

Важно отметить, что для такого подхода не требуется хранить весь массив прошлых значений сигнала, достаточно иметь лишь накопленную оценку среднего, число прошедших замеров и значение очередного отсчета сигнала. Такое изящество достигается, правда, за счет дополнительного введения операций умножения и деления на каждом шаге вычислений. Кроме того, рост числа операций в условиях ограниченной разрядной сетки быстро приводит к ухудшению точности результата. Тем не менее, для вычисления собственно оценки среднего на «Электронике-100» в процессе ввода исходных данных с объемом массива вплоть до $m < 2048$ рекуррентная процедура может быть успешно построена на основе соотношения (10.8а)¹. Использование процессора для вычисления очередного приближенного значения $(x_{cp})_m$ на то время, пока вводное устройство обеспечит ввод очередного $(m + 1)$ -го значения из массива исходных данных, позволяет к концу первого прохода перфоленты иметь и среднее значение анализируемого параметра, которое оказывается необходимым на следующем этапе анализа — при вычислении корреляционной функции сигнала.

Для вычисления собственной корреляционной функции сигнала обычно используется соотношение

$$R_{xx}[n] = \frac{1}{N-n} \sum_{i=1}^{N-n} x_i x_{i+n}, \quad (10.9)$$

где n — аргумент корреляционной функции (интервал временного сдвига внутри каждой пары отсчетов, участвующей в вычислениях); N — объем выборки; i — текущий номер отсчета; x_i — центрированное значение отсчета:

$$x_i = x_j - x_{cp}. \quad (10.10)$$

Взаимно-корреляционную функцию вычисляют как

$$R_{xy}[n] = \frac{1}{N-n} \sum_{i=1}^{N-n} x_i y_{i+n} \quad (10.11)$$

Другими формами тех же соотношений обычно считаются

$$R_{xx}[n] = \frac{1}{N-n} \sum_{i=1}^{N-n} x_i x_{i+n} - x_{cp}^2, \quad (10.9a)$$

$$K_{xy}[n] = \frac{1}{N-n} \sum_{i=1}^{N-n} x_i y_{i+n} - x_{cp} y_{cp} \quad (10.11a)$$

Для организации соответствующих расчетов на «Электронике-100» важно проследить за изменениями разрядности исходных данных и результатов и тонкостями, связанными с центрированием реализаций.

Прежде всего, необходимо отметить, что в общем случае процесс центрирования может приводить к увеличению разрядности отдельных отсчетов, но не более чем на один двоичный порядок². В этих условиях формы (10.9a) и (10.11a) являются более предпочтительными для построения рабочих программ. С другой стороны, в случае применения стандартного 10-разрядного АЦП при вводе исходных сигналов в «Электронику-100» увеличение разрядной сетки исходных данных на один двоичный разряд в процессе центрирования ещё не приводит к переполнению разрядной сетки машины и поэтому не является существенным. Кроме того, для режима нормального функционирования объекта флуктуации его параметров обычно имеют малую относительную величину, в связи с чем возникает другая ситуация.

Если флуктуации параметров имели малую относительную величину и при измерениях и вводе сигнала в машину не были приняты специальные меры для уменьшения его постоянной составляющей с целью увеличения точности представления данных (см. гл. 4), то в центрированной реализации большая часть старших разрядов каждого из отсчетов (в пределах прежней разрядной сетки) оказывается незначительной. В то же время из-за повышения точности представления x_{cp} в связи с использованием при его вычислении значительного объема данных, у него появляются дополнительные значащие разряды, располагающиеся, однако, уже за пределами прежней разрядной сетки. Точно отцентрированная реализация могла бы также располагать этими дополнительными разрядами, но при условии представления каждому отсчету центрированной реализации дополнительной ячейки памяти. Помимо резкого увеличения объема требуемой памяти дополнительно возросли бы и требования к программному обеспечению по работе с массивом данных, причем каждое число из этого массива располагается в двух ячейках. Возможный путь преодоления этого затруднения состоит в предварительном (неточном) центрировании реализаций в пределах прежней разрядной сетки и последующее применение к этим реализациям соотношений (10.9a) и (10.11a).

Таким образом, и в том случае, когда центрирование реализации не ведет к увеличению разрядной сетки в сторону больших разрядов, применение последних соотношений оказывается более предпочтительным.

Наконец, для малых относительных флуктуаций сигналов и небольших объемов реализаций существенную роль начинают играть дополнительные погрешности, связанные с использованием усреднения по времени. Дело в том, что соотношения (10.9a) и (10.11a) записаны по аналогии со случаем использования характеристик ансамбля. При использовании усреднения по времени эти соотношения нуждаются в поправках.

Действительно, для оценки второго начального момента, вычисляемой усреднением по времени, с учётом (10.10) можно записать:

¹ Когда текущая оценка среднего не представляет самостоятельного интереса, алгоритм текущей обработки данных ещё более упрощается — достаточно хранить накапливающуюся сумму (вместо оценки среднего) и номер текущего отсчёта (см. выражение после второго знака равенства в (10.8)), т. е. после получения каждого замера здесь выполняется ядро цикла обычного способа вычисления оценки среднего и изменение содержимого счётчика использованных отсчётов.

² Это следует из анализа взаимосвязи модулей переменных, входящих в (10.10). Когда x_i и x_{cp} имеют разные знаки, можно записать:
 $|x_i| = |x_i - x_{cp}| = |x_i| + |x_{cp}| \geq |x_i|$. Однако из очевидных соотношений $|x_i| \leq |x|_{\max}$ и $|x_{cp}| \leq |x|_{\max}$, справедливых при тех же предположениях, следует также, что $|x_i| = |x_i| + |x_{cp}| < 2|x|_{\max}$.

$$\frac{1}{N-n} \sum_{i=1}^{N-n} x_i x_{i+n} = \frac{1}{N-n} \sum_{i=1}^{N-n} (x_i + x_{cp})(x_{i+n} + x_{cp}) = \frac{1}{N-n} \sum_{i=1}^{N-n} x_i x_{i+n} + x_{cp}^2 + \frac{x_{cp}}{N-n} \sum_{i=1}^{N-n} (x_i + x_{i+n}) \quad (10.12)$$

Нетрудно видеть, что последняя сумма в (10.12) как частная сумма конечного числа отсчетов (хотя и центрированного) случайного процесса в общем случае всегда отличается от нуля случайным образом, создавая стохастические добавки к первым двум слагаемым. Следует отметить, что при усреднении по множеству последние члены в выражении, аналогичном (10.12), всегда обращаются в нуль, здесь же эти слагаемые лишь флуктуируют около нуля. Таким образом, статистические флуктуации оценки второго момента, получаемой усреднением по времени, складываются из флуктуации частных оценок корреляционной функции около ковариационной функции ансамбля (флуктуации первого слагаемого в (10.12)) и дополнительных флуктуаций, обусловленных последней суммой, причем вклад этих флуктуаций в дисперсию оценки второго момента оказывается аддитивным для широкого класса сигналов с симметричным законом распределения. Можно показать, что относительная среднеквадратичная погрешность, возникающая в ординатах оценки корреляционной функции при отбрасывании последней суммы в (10.12) (или, что эквивалентно, при использовании соотношения (10.9а)), составляет в лучшем случае величину порядка $2/\sqrt{N} W_x$ (где $W_x = \sigma_x/m_x$ — коэффициент вариации) и, например, для $W_x = 0,1$ и $N = 400$ достигает 100%.

Поэтому в практических расчетах корреляционной функции целесообразно устранить указанный дополнительный источник статистических флуктуации оценки, определяя на основании (10.12) и (10.7) $R_{xx}[n]$ в виде¹

$$R_{xx}[n] = \frac{1}{N-n} \sum_{i=1}^{N-n} x_i x_{i+n} = \frac{1}{N-n} \sum_{i=1}^{N-n} x_i x_{i+n} - x_{cp}^2 - x_{cp}^2 \frac{2n}{N-n} + \frac{x_{cp}}{N-n} \left(\sum_{i=1} x_i + \sum_{i=N-n+1}^N x_i \right). \quad (10.96)$$

Аналогично для взаимной корреляционной функции

$$R_{xy}[n] = \frac{1}{N-n} \sum_{i=1}^{N-n} x_i y_{i+n} - x_{cp} y_{cp} - x_{cp} y_{cp} \times \frac{2n}{N-n} + \frac{x_{cp}}{N-n} \sum_{i=1}^n y_i + \frac{y_{cp}}{N-n} \sum_{i=N-n+1}^N x_i. \quad (10.116)$$

Структурная схема программы для оценки статистических характеристик сигналов, разработанная для «Электроники-100» с учётом рассмотренных особенностей, приведена на рис. 10.3.

Нетрудно видеть, что под промежуточные произведения при вычислениях корреляционных функций приходится занимать по две ячейки, а при их суммировании промежуточный результат занимает уже три ячейки. В связи с этим целесообразно использовать стандартные подпрограммы умножения и деления, как с обычной, так и с удвоенной точностью. Завершающими штрихами вычисления корреляционной функции при малой относительной величине флуктуации (что оценивается по значащим разрядам $R_{xx}[0]$) может являться учёт разрядов, вес которых ниже веса младшего разряда в исходном представлении данных.

¹ Следует отметить, что при практических расчётах дополнительным источником погрешности (теперь уже в виде смещенности оценки корреляционной функции относительно ковариационной) является процедура определения оценки среднего по той же реализации, с последующим её использованием в (10.96), (10.116). Величина смещенности зависит от объёма выборки и относительной (при выбранной частоте отсчётов) скорости убывания (оцениваемой) корреляционной функции, а также от её вида. Однако относительная величина этой смещенности, как можно показать, имеет порядок $n_1 N^{-1}$ (где n_1 — число ординат корреляционной функции, существенно отличных от нуля), при надлежащем выборе N не превышает нескольких процентов и может быть дополнительно снижена использованием в расчётных соотношениях делителя $(N - n - n_1)$ вместо $(N - n)$.

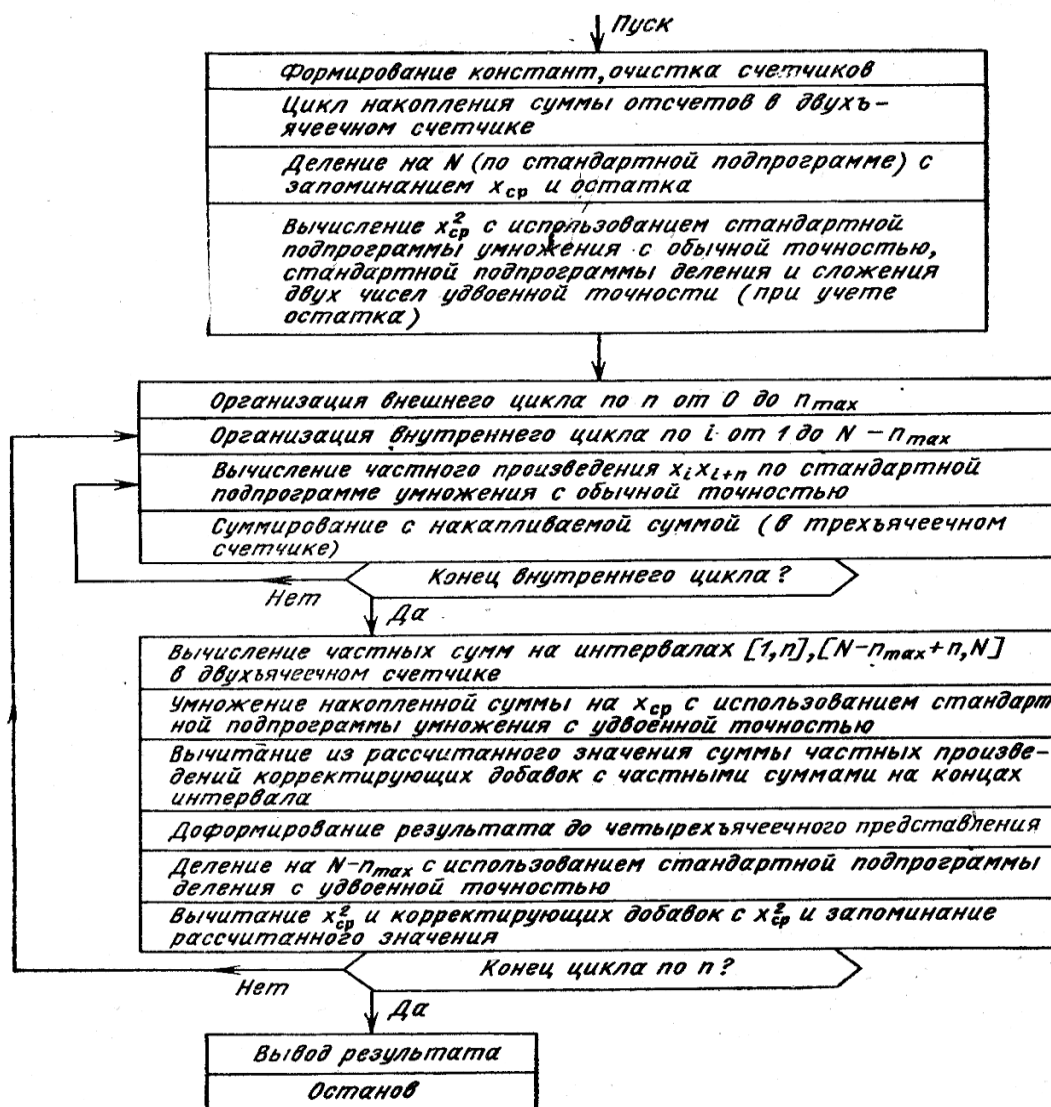


Рис. 10.3. Структурная схема расчета статистических характеристик сигнала (среднего значения и корреляционной функции).

10.4. О ПОЛУЧЕНИИ ОЦЕНОК ВЕСОВОЙ ФУНКЦИИ ОБЪЕКТА ПО ОСНОВНЫМ КАНАЛАМ¹

Теперь, когда рассмотрены вопросы расчета корреляционных функций на «Электронике-100», возникает вопрос о возможности их непосредственного использования для получения оценок весовой функции на основе соотношения, аналогичного (10.6). Пример, показывающий некорректность прямого использования подобных аналогий, уже приводился в предыдущем параграфе. Дополнительную почву для размышления создает достаточно широко распространенная ситуация, когда часть измерений выходного параметра отсутствует². Не требует ли и построение рабочих программ в этих условиях некоторой коррекции исходных соотношений?

Для определения вида необходимой коррекции достаточно рассмотреть несколько иной подход при получении требуемых характеристик с применением широко распространенного на практике метода наименьших квадратов. Следует отметить, что привлечение этого метода позволит установить лишь путь возможного решения задачи — статистические свойства такого решения (его характеристики точности) могут быть изучены только с использованием характеристик ансамбля.

Полагая, что характеристики объекта достаточно хорошо представимы дискретной весовой функцией³ $h[l]$, мгновенное значение разности доступного измерению выходного сигнала $z[j]$ и предсказываемого на основании отсчетов входного сигнала $x[j]$ и этой весовой функции можно записать как

¹ Некоторые результаты этого параграфа опубликованы в [29].

² Отсутствие отдельных измерений может быть не только случайным событием, связанным со сбоями и временными отказами аппаратуры или отбраковкой полуфабриката ещё в середине производственного цикла. Существует целая группа так называемых дискретно-непрерывных процессов, где в технологических агрегатах непрерывно поддерживаются определенные условия (температура, давление, расходы и т. п.), а изготавливаемые изделия дискретно проходят через агрегаты только после очередной загрузки, не обязательно периодической во времени.

³ Вопросы точности дискретно-непрерывных преобразований реальных сигналов разработаны Ю. М. Быковым (см., например, [38, 39]).

$$\varepsilon[j] = z[j] - \sum_{l=0}^L h_l[l] x[j-l] . \quad (10.13)$$

Для метода наименьших квадратов критерий качества имеет вид

$$F = \sum_{j \in J} \varepsilon^2[j] = \sum_{j \in J} \left\{ z^2[j] - 2 z[j] \sum_{k=0}^L h_k[k] x[j-k] + \sum_{k=0}^L \sum_{p=0}^L \hat{h}_k[k] \hat{h}_p[p] x[j-k] x[j-p] \right\} \quad (10.14)$$

Здесь обозначение в пределах первой суммы указывает, что j принимает лишь те значения, которые принадлежат множеству J . В данном случае это множество может содержать не все целочисленные значения, соответствующие последовательным периодическим моментам дискретных отсчетов в системе, а лишь ту часть из них, для которых отсчет выходного сигнала в действительности осуществлялся. Изменение переменной суммирования с участием коэффициентов весовой функции в последнем слагаемом внутри фигурных скобок, очевидно, значения не имеет, но увеличивает наглядность необходимых преобразований.

Нетрудно видеть, что критерий качества в виде (10.14) является функцией $(L + 1)$ -й переменной $\hat{h}_l[l]$, $l = 0, 1, 2, \dots, L$. Дифференцируя (10.14) по каждому из коэффициентов $\hat{h}_l[l]$ и приравнявая производные нулю, получаем систему $(L+1)$ линейных уравнений относительно искомых коэффициентов:

$$\begin{aligned} \frac{dF}{d\hat{h}_1[0]} &= \sum_{j \in J} \left\{ -2 z[j] x[j] + 2 \sum_{q=0}^L \hat{h}_1[q] x[j] x[j-q] \right\} = 0, \\ \frac{dF}{d\hat{h}_1[1]} &= \sum_{j \in J} \left\{ -2 z[j] x[j-1] + 2 \sum_{q=0}^L \hat{h}_1[q] x[j-1] x[j-q] \right\} = 0, \\ &\dots\dots\dots \\ \frac{dF}{d\hat{h}_1[L]} &= \sum_{j \in J} \left\{ -2 z[j] x[j-L] + 2 \sum_{q=0}^L \hat{h}_1[q] x[j-L] x[j-q] \right\} = 0, \end{aligned} \quad (10.16)$$

Сокращая на 2 и производя суммирование по j , можно всю систему уравнений записать в виде равенства функций:

$$R_{xx}[l] - \sum_{q=0}^L \hat{h}_1[q] R_{xx}[l-q] = 0, \quad l = 0, 1, 2, \dots, L. \quad (10.16)$$

И хотя (10.16) внешне аналогично (10.6), способ определения функций в (10.16) существенно отличен от (10.9), (10.11):

$$R_{xx}[l-q] = \sum_{j \in J} x[j-l] x[j-q], \quad (10.17)$$

$$R_{xz}[l] = \sum_{j \in J} z[j] x[j-l]. \quad (10.18)$$

Дело здесь не только в отсутствии нормирующих делителей перед суммами. Эти делители, одинаковые во всех случаях и равные числу доступных измерений выхода, т. е. объёму множества J , могут быть легко восстановлены делением правой и левой частей (10.15), (10.16) на это число.

Особенностью является то, что собственная корреляционная функция входного сигнала в (10.17) определяется как матрица, коэффициенты которой вычисляются с «привязкой» к тем моментам времени, для которых есть в наличии измерения выхода. И хотя по исходным предположениям процесс стационарен и эргодичен, для разных пар l и q , имеющих даже одинаковую разность, частные оценки $R_{xx}[l-q]$, вычисляемые по различным участкам одной и той же реализации конечной длины, всегда несколько различаются. Такая матрица отличается от той, которая могла бы быть построена только на основе корреляционной функции (10.9) с учётом свойства чётности.

Другой особенностью соотношений (10.17), (10.18), важной для построения практических программ, является указание на использование прошлых (относительно замеров выходного сигнала) замеров входных сигналов. Это, в частности, указывает на то, что для данных расчетов реализация выходного сигнала должна начинаться не ранее, чем спустя L отсчетов от начала реализации входного сигнала (т. е. с разностью на длину «памяти» весовой функции объекта). Такое требование имеет отчетливый физический смысл: в противном случае исследователь не располагает замерами всех тех значений входного (случайного) сигнала, которые фактически определили значение выходного сигнала в начале реализации.

Отмеченные различия могут приводить к очень заметным искажениям результатов при попытках использования в (10.6) характеристик, рассчитываемых на основании (10.9), (10.11), особенно в случае малой длины исходной реализации. Один из таких результатов, полученных путем моделирования, приведён на рис. 10.4.

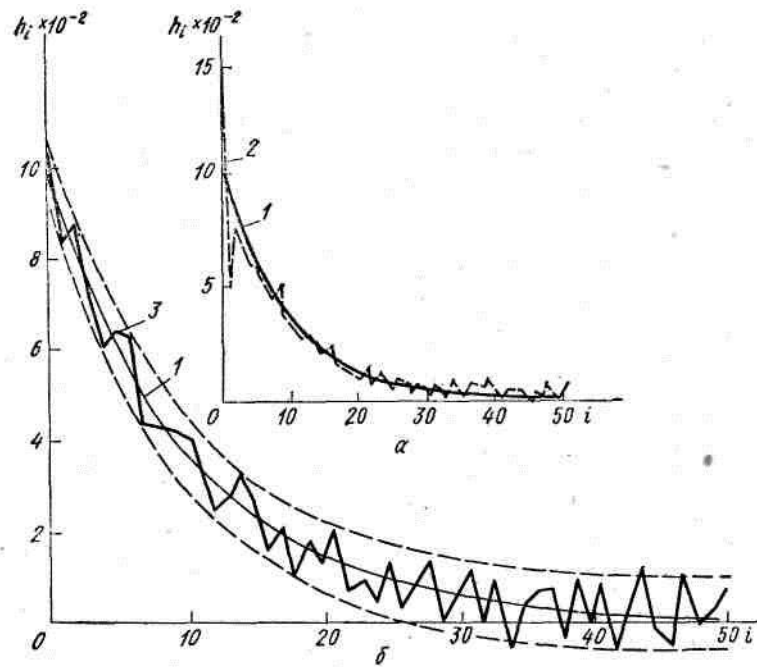


Рис. 10.4. Результаты модельной идентификации динамического объекта:
 α — для случая $n(t)=0$; 1 — результат (совпадающий с истинной моделью) при использовании соотношений (10.16) — (10.18) и объеме выборки, начиная со 100 отсчетов; 2 — объем выборки 1000 отсчетов при использовании уравнения типа «Винера — Хопфа»; 3 — влияние помехи; 1 — истинная весовая функция; 3 — результат расчета; штрихом указаны доверительные границы $\pm 2\sigma_{\Delta h}$, рассчитанные в соответствии с (10.35 а).

Соотношения (10.17), (10.18) указывают точный путь построения необходимых расчётных операций. Тонкости, связанные с их реализацией на «Электронике-100», ничем не отличаются от рассмотренных выше. Окончательное определение искоемых коэффициентов весовой функции происходит при решении системы линейных уравнений (10.16). Один из возможных способов осуществления этого этапа рассмотрен ниже.

Большой самостоятельный интерес представляют вопросы оценки точности получаемых результатов и возможность расширения развитого подхода на случай многопараметрического объекта. Для этой цели удобно ввести векторное представление переменных и использовать соответствующие матричные операции.

Пусть h и x_j являются $(L + 1)$ -мерными вектор-столбцами:

$$h = \begin{Bmatrix} h_0 \\ h_1 \\ \vdots \\ h_L \end{Bmatrix}, \quad x_j = \begin{Bmatrix} x_j \\ x_{j-1} \\ \vdots \\ x_{j-L} \end{Bmatrix}, \quad (10.19)$$

где j — индекс (целочисленное значение) текущего момента времени.

Для простоты дальнейших обозначений будем полагать, что входной и выходной сигналы центрированы. Тогда мгновенное значение истинного выходного сигнала объекта в момент времени j можно представить в виде следующего векторного эквивалента (10.1):

$$y_j = Kh'x_j = Kx_j'h, \quad (10.20)$$

где $'$ — символ транспонирования.

Все доступные измерения выходного сигнала можно также представить в виде вектора-столбца:

$$z = y + n = \begin{Bmatrix} y_{j_1} + n_{j_1} \\ y_{j_2} + n_{j_2} \\ \vdots \\ y_{j_m} + n_{j_m} \end{Bmatrix}. \quad (10.21)$$

В этой записи отражен векторный эквивалент соотношения (10.3), а также принадлежность отдельных моментов измерения сигналов $j_1, j_2, \dots, j_m$ множеству J , которое может включать не все подряд целые числа (т. е. не обязательно $j_{i+1} - j_i = 1$) и имеет общий объем в m замеров.

С учётом (10.19) и (10.20) последнее выражение можно записать в виде

$$z = KXh + n, \quad (10.22)$$

где X — матрица измерений входного сигнала размерностью $m \times [L + 1]$ (т. е. эта матрица имеет m строк и $L + 1$ столбцов).

Теперь собственную корреляционную функцию входного сигнала в виде (10.17) можно определить как матрицу

размерностью $(L + 1) \times (L + 1)$:

$$A_{xx} = X'X, \quad (10.23)$$

а взаимно-корреляционную функцию входного и выходного сигналов объекта в виде (10.18) — как $(L + 1)$ -мерный вектор-столбец:

$$R_{xz} = X'z, \quad (10.24)$$

после чего уравнение (10.16) принимает вид

$$R_{xz} - A_{xx} h_1 = 0. \quad (10.25)$$

Решением этого уравнения является

$$h_1 = A_{xx}^{-1} R_{xz}, \quad (10.26)$$

где A_{xx}^{-1} — матрица, обратная матрице A_{xx} , так что

$$A_{xx}^{-1} A_{xx} = E = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \quad (10.27)$$

Если теперь принять во внимание, что с учётом (10.22) и (10.23) соотношение (10.24) можно представить как

$$R_{xz} = X'z = K'X'Xh + X'n, \quad (10.28)$$

то полученное решение (10.26) приобретает вид

$$h_1 = A_{xx}^{-1} [KA_{xx}h + X'n] = Kh + A_{xx}^{-1}X'n. \quad (10.29)$$

Последнее соотношение имеет весьма важное значение. Прежде всего, из него следует, что основным источником отличия рассчитываемых коэффициентов весовой функции от их истинных значений является помеха в выходном сигнале n . При $n = 0$, как следует из (10.29), $\hat{h} = h$ независимо от того, по какой частной реализации сигнала $x(t)$ рассчитывалась оценка h_1 , какой вид (характер) имеет истинная весовая функция h и какова величина статического коэффициента передачи в рабочей точке K . Кроме того, из этого же соотношения следует, что оценка h_1 является несмещенной, если помеха не зависит от сигнала и её математическое ожидание равно нулю. Действительно, если

$$M\{n\} = 0, \quad (10.30)$$

то

$$M\{\hat{h}_1\} = Kh + M\{A_{xx}^{-1}X'n\} = Kh + M\{A_{xx}^{-1}X'\}M\{n\} = Kh. \quad (10.31)$$

Здесь $M\{\}$ обозначает операцию определения математического ожидания.

На основании рассмотрения нечётных моментов нетрудно показать также, что при симметричном законе распределения помехи и закон распределения оценки весовой функции является симметричным.

Выражение (10.29) позволяет интерпретировать путь трансформации флуктуации помехи во флуктуации результата:

$$\Delta h_1 = h_1 - Kh = A_{xx}^{-1}X'n = A_{xx}^{-1}R_{xn}. \quad (10.32)$$

Вначале помеха компонуется с сигналом, образуя после усреднения взаимно-корреляционную функцию сигнала и помехи R_{xn} , и уже её значения через обратную матрицу сигнала A_{xx}^{-1} пересчитываются в значения компонент ошибки в оценке весовой функции. В силу независимости сигнала и помехи их взаимная корреляционная функция в пределе, при неограниченном увеличении интервала наблюдения, стремится к нулю, однако статистические флуктуации частных оценок именно этой функции играют определяющую роль в статистических флуктуациях оценки весовой функции, вычисляемой при использовании реализаций конечной длины.

Количественную оценку величины и взаимосвязи флуктуации отдельных ординат собственно весовой функции дает ковариационная матрица погрешности $C_{\Delta h}$, которая в случае известного коэффициента передачи определяется как

$$C_{\Delta h} = M\left\{\frac{1}{K}\Delta h_1 \frac{1}{K}\Delta h_1'\right\} = M\{K^{-2}A_{xx}^{-1}X'nn'X[A_{xx}^{-1}']\} = K^{-2}M\{A_{xx}^{-1}XC_nX[A_{xx}^{-1}']\}, \quad (10.33)$$

где C_n — ковариационная матрица помехи.

Для δ -коррелированной помехи последнее выражение с учётом (10.23) и симметричности матрицы A_{xx} упрощается:

$$C_{\Delta h} = K^{-2}\sigma_n M\{A_{xx}^{-1}\} \quad (10.34)$$

И наконец, для достаточно больших m последнее выражение приобретает вид¹

$$C_{\Delta h} = \frac{\sigma_n^2}{mK^2} K_{xx}^{-1} \quad (10.35)$$

Таким образом, дисперсия флуктуации ординат оценки весовой функции зависит от уровня помехи, интервала наблюдения

¹ Матрицу A_{xx} можно представить как $A_{xx} = m(K_{xx} + B)$, где K_{xx} — матрица, образованная на основе собственной ковариационной функции входного сигнала; B — матрица (случайных) статистических флуктуаций корреляционной функции около K_{xx} . При достаточно больших m матрицу, обратную A_{xx} , в этом случае можно представить в виде (см. [40]) $A_{xx}^{-1} \approx m^{-1}[K_{xx}^{-1} - K_{xx}^{-1}BK_{xx}^{-1}]$, откуда в силу несмещенности оценки корреляционной функции относительно ковариационной следует: $M\{A_{xx}^{-1}\} = 1/m K_{xx}^{-1}$.

(числа накопленных отсчетов), коэффициента передачи в рабочей точке и значений диагональных элементов матрицы K_{xx}^{-1} . Увеличение дисперсии помехи ведет к росту дисперсии ординат оценки; рост объема исходных данных и крутизны характеристики объекта в рабочей точке уменьшает погрешности идентификации. Ковариация ординат оценки весовой функции зависит от тех же параметров и вида остальных элементов матрицы K_{xx}^{-1} .

В частности, для широко распространенного типа сигналов с экспоненциальной ковариационной функцией

$$K_{xx}(\tau) = \sigma_x^2 e^{-\alpha|\tau|} \quad (10.36)$$

соотношение (10.35) приобретает вид

$$C_{\Delta h} = \frac{\sigma_n^2}{mK\sigma_x^2(1-e^{-2\alpha T_s})} \times \begin{bmatrix} 1 & -e^{-\alpha T_s} & 0 & \dots & 0 & 0 \\ -e^{-\alpha T_s} & (1+e^{-2\alpha T_s}) & -e^{-\alpha T_s} & & \vdots & \vdots \\ 0 & -e^{-\alpha T_s} & (1+e^{-2\alpha T_s}) & & & \\ \vdots & \vdots & \vdots & \ddots & (1+e^{-2\alpha T_s}) & -e^{-\alpha T_s} \\ 0 & 0 & 0 & \dots & & 1 \end{bmatrix} \quad (10.35a)$$

Здесь σ_x^2 — дисперсия входного сигнала; T_s — принятый в системе период отсчетов.

Последнее выражение показывает дополнительно, что для точности идентификации весовой функции имеет значение не абсолютная, а относительная величина помехи в выходном сигнале и относительная скорость изменения входного сигнала (в принятом масштабе отсчетов времени в системе). Важным результатом здесь является также вывод о взаимной коррелированности лишь трех соседних ординат оценки весовой функции в случае экспоненциальной ковариационной функции входного сигнала.

Какие практически важные выводы можно сделать на основе приведенного анализа?

1. Принципиально важной является сама возможность получения оценок характеристик динамического объекта на основе данных его нормального функционирования.

2. Оценки весовой функции объекта могут рассчитываться по статистическим характеристикам сигналов, получаемым усреднением по времени при конечной длине реализаций. Рассчитываемые по ним оценки ординат весовой функции являются в первом приближении несмещенными и состоятельными (а при конечном времени наблюдения имеют конечную дисперсию).

3. Характеристики ансамбля сигналов играют роль лишь на этапе оценки точности получаемых результатов; на всем протяжении расчетов собственно ординат весовой функции эти характеристики и факт сходимости к ним их частных оценок (корреляционных функций) роли не играют.

Последние обстоятельства создают прочную базу для оперативного количественного изучения характеристик действующих объектов. Оказывается возможным успешно использовать понятие квазистационарности [41], применять при идентификации сравнительно короткие реализации сигналов, а при анализе точности полученных результатов использовать статистические характеристики сигналов, изученные (возможно, предварительно) по реализациям достаточно большой длины.

Следует отметить, что используемый подход имеет некоторое сходство с регрессионным анализом, но есть и существенные отличия. Первое из них состоит в способе отбора отсчетов сигнала из входной реализации — сопоставление каждого отсчета выходного сигнала с целым вектором прошлых отсчетов входного сигнала (см. (10.19), (10.20)), причем один и тот же отсчет входного сигнала может служить (каждый раз новой) компонентой вектора x_j при различных $j \in J$. Другое отличие состоит в коррелированности отсчетов случайного входного сигнала, что делает невозможным применение обычных оценок точности регрессионного анализа.

Оба указанных обстоятельства связаны, конечно, с использованием более широкого класса входных воздействий — случайных процессов вместо случайных чисел, как это имеет место в случае регрессионного анализа.

Для объекта, имеющего q отдельных каналов, достаточно ввести в рассмотрение следующие обобщенные векторы весовой функции и входного сигнала:

$$H = \begin{Bmatrix} h_1 \\ h_2 \\ \vdots \\ h_q \end{Bmatrix}, \quad x_j = \begin{Bmatrix} x_{j1} \\ x_{j2} \\ \vdots \\ x_{jq} \end{Bmatrix}, \quad (10.37)$$

где каждый из векторов h_i представляет весовую функцию по i -му каналу и каждое из соответствующих входных воздействий определяется соотношением (10.19).

За счет составления отдельных векторов, каждый из которых может иметь свою собственную размерность L_i , увеличивается общая размерность векторов H и x_j . Для вектора x_j дополнительной характерной чертой является q -кратное использование момента времени j , в который производятся измерения по всем q каналам, а также прошлых моментов времени $j-1, j-2, \dots$, причем в начале вектора x_j располагаются L_1+1 замеров первого входного параметра, затем L_2+1 замеров второго параметра и т. д.

$$H_1 = \begin{Bmatrix} K_1 h_1 \\ K_2 h_2 \\ \vdots \\ K_q h_q \end{Bmatrix} \quad (10.38)$$

Дополнительная особенность результатов состоит здесь в том, что при независимых входных сигналах матрица K_{xx}^{-1} в

выражении, аналогичном (10.35), имеет вид

$$K_{xx}^{-1} = \begin{bmatrix} K_{x_1 x_1}^{-1} & 0 & \cdots & 0 \\ 0 & K_{x_2 x_2}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K_{x_q x_q}^{-1} \end{bmatrix}, \quad (10.39)$$

который предопределяет независимость оценок весовых функций по отдельным каналам друг от друга и зависимость статистических свойств этих оценок, являющихся несмещенными и состоятельными, лишь от относительного уровня своего входного сигнала и помехи n , объёма выборки и вида ковариационной функции сигнала только по своему каналу.

Существуют ли какие-либо объективные препятствия для реализации процедуры идентификации характеристик объекта, обладающей столь хорошими свойствами?

К сожалению, да. Размерность матрицы A_{xx} составляет $\sum_{i=1}^q (L_i + 1) \times \sum_{i=1}^q (L_i + 1)$. Так для объекта, который имеет 10 динамических каналов по 25 ординат весовой функции в каждом, число коэффициентов матрицы A_{xx} составляет 62 500. При таком объёме промежуточных данных ресурс не только «Электроники-100», но и более мощных машин оказывается недостаточным для выполнения дальнейших вычислений и, в частности, для обращения матрицы. По-видимому, полностью решить задачу идентификации в таких условиях способна лишь очень мощная вычислительная система, например система коллективного пользования. Для реализации соответствующего подхода необходимо обеспечить оперативный процесс получения исходных данных от объекта и их передачу в эту систему, возможно, с некоторой предварительной обработкой. Соответствующие вопросы реализации низовых функций такой комплексной системы на базе «Электроники-100» и её связи с ЭВМ, работающей на более высоком иерархическом уровне, рассматривались в гл. 8, 9; требования к системе коллективного пользования со стороны отдельных АСУТП-абонентов рассмотрены, например, в [42].

Существуют ли какие-либо приближенные или упрощенные подходы к решению задачи идентификации с менее ощутимыми затратами?

В качестве одного из таких путей можно указать возможность раздельной идентификации характеристик объекта по каждому из каналов. При таком подходе все остальные каналы своим вкладом в выходной сигнал создают дополнительную помеху по отношению к идентифицируемому каналу. Если все входные сигналы независимы друг от друга, то оценка весовой функции по-прежнему остается несмещенной, но рост уровня помехи ведет к ухудшению статистической стабильности оценки. Можно показать, что ковариационная функция погрешности в оценке h в этих условиях дополняется матрицей

$$\Delta C_{\Delta h_1} = M\{DH_1 H_1' D'\} \quad (10.40),$$

где D имеет вид

$$D = \begin{bmatrix} 0 & R_{x_1 x_1}^{-1} R_{x_1 x_2} & R_{x_1 x_1}^{-1} R_{x_1 x_3} & \cdots & R_{x_1 x_1}^{-1} R_{x_1 x_q} \\ R_{x_2 x_2}^{-1} R_{x_2 x_1} & 0 & R_{x_2 x_2}^{-1} R_{x_2 x_3} & \cdots & R_{x_2 x_2}^{-1} R_{x_2 x_q} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ R_{x_q x_q}^{-1} R_{x_q x_1} & R_{x_q x_q}^{-1} R_{x_q x_2} & R_{x_q x_q}^{-1} R_{x_q x_3} & \cdots & 0 \end{bmatrix} \quad (10.41)$$

Нетрудно видеть, что в данном случае вариация оценок дополнительно зависит от положения рабочей точки и характеристик сигналов по всем остальным каналам.

Однако наиболее существенным выводом является то, что оценка по-прежнему является состоятельной и её точность возрастает при увеличении интервала наблюдения. Платой за использование упрощенного подхода является необходимость удлинения этого интервала по сравнению со случаем идентификации полного вектора весовых функций по всем каналам. Однако при этом резко снижается и размерность промежуточного массива данных, что позволяет ставить и решать задачу идентификации характеристик динамического объекта полностью на «Электронике-100» с использованием одного из известных методов [43] численного решения системы линейных уравнений.

ЗАКЛЮЧЕНИЕ

«Электроника-100» как и любая малая ЭВМ, представляет собой лишь основу (базу) для построения все новых и новых систем. Есть, однако, целый ряд задач, решение которых заметно расширило бы круг возможных применений и повысило эффективность использования машины.

С точки зрения развития аппаратной части самой машины представляется первоочередным скорее всего наращивание оперативной и долговременной памяти. Во многих практических задачах АСУ ТП объём памяти является наиболее узким местом, препятствующим увеличению мощности и развитию функций системы. Целесообразным представляется введение аппаратной защиты участков МОЗУ. Комплекс внешних устройств машины, по-видимому, нуждается в расширении для обеспечения возможности прямого ввода в машину более широкого класса стандартных сигналов от промышленных датчиков.

Программное обеспечение «Электроники-100» достаточно разнообразно и есть несколько важных самостоятельных направлений его освоения и развития. Работы здесь настолько много, что назревает настоятельная необходимость специализации отдельных организаций по тем или иным направлениям с последующим обменом результатами, например в рамках ведомственных объединений или целой Ассоциации пользователей, подобной созданным у пользователей ЭВМ типа «Минск» и

«М-20». Такое объединение должно координировать планы работ, обеспечивать обмен информацией и иметь возможность публиковать наиболее интересные результаты по разработке и освоению математического обеспечения для «Электроники-100». При накапливаемой библиотеке программ для машины целесообразно иметь вместе с описанием программ и инструкциями по их использованию алгоритмы построения и структурные схемы программ, по крайней мере, на уровне крупных блоков.

Установление единой формы описания текста программ с пояснениями облегчается наличием в системе математического обеспечения машины языка символического кодирования. В связи с этим одной из первоочередных задач каждого пользователя является овладение техникой программирования на ассемблере с применением содержательных примечаний (комментариев) как средства не только повышения общей культуры программирования, но и общения с другими пользователями.

Своеобразной проблемой, решение которой открыло бы простой доступ к ассемблеру широкому кругу отечественных пользователей, является переработка входного языка в русской интерпретации.

Возможность подобного развития заложена в самом исходном языке (за счёт наличия псевдокоманд EXPUNGE и FIXMRI).

Крупной задачей программного обеспечения задач, решаемых в реальном масштабе времени, и в первую очередь задач АСУ ТП, является разработка специализированного языка программирования и транслятора с него для «Электроники-100». Решение этой сложной задачи должно, очевидно, пройти через ряд этапов, связанных с выделением состава задач, их описания и решения. В идеальном случае разработанное математическое обеспечение должно бы было позволять компилировать рабочие программы системы для решения задач, которые сформулированы в исходном описании (задании на проектирование системы) вместе с условиями её функционирования. Рациональной формой заполнения входного описания функций системы и условий ее работы является применение вопросника, в котором заготовлено достаточное множество вопросов, предлагаемых проектировщику в режиме диалога по мере ответа на предыдущие вопросы. Единый подход позволил бы здесь в определенной мере автоматизировать создание математического обеспечения системы, удовлетворяющей заданным требованиям и имеющей возможности и Перспективу развития.

Несомненную помощь в создании соответствующих программ для «Электроники-100» оказали бы простейшие языки программирования. Что касается возможного развития для этой цели одного из разработанных для «Электроники-100» языков высокого уровня, то с учётом ограниченного объёма памяти здесь целесообразно ограничиться лишь заимствованием наиболее эффективных приемов программной реализации отдельных функций, а это, в свою очередь, требует ещё значительной работы по их выявлению.

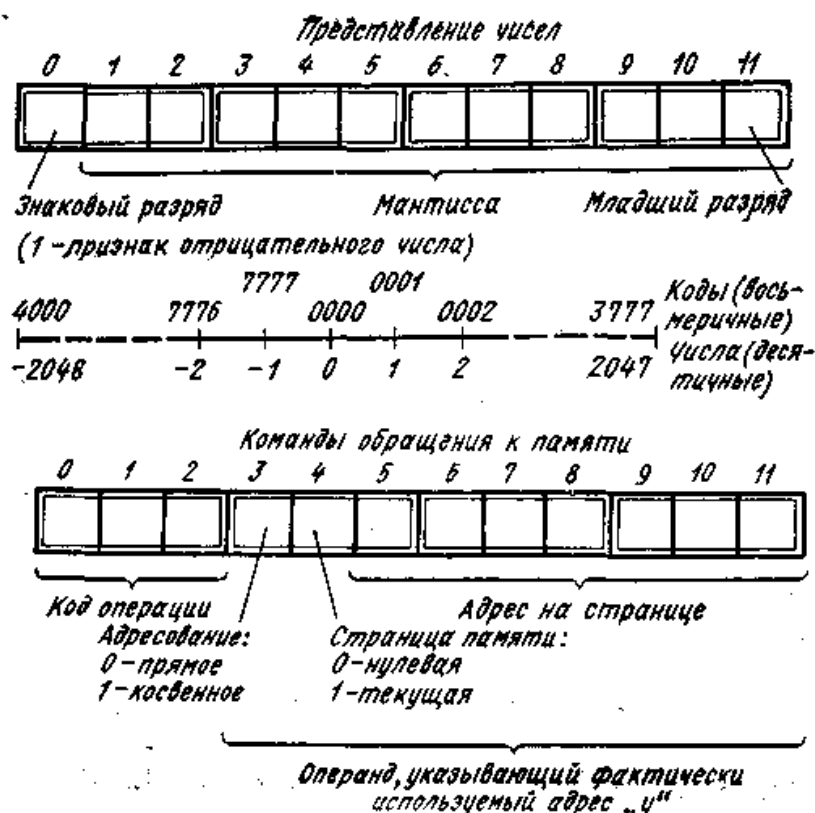
Из других задач большой практический интерес представляет рациональная организация библиотеки наиболее употребительных стандартных программ в кодах машины. При использовании языков программирования эта задача решается применением загрузчика в перемещаемом формате и соответствующим образом оформленных подпрограмм в библиотеке данного языка.

Большие перспективы могут иметь работы, связанные с созданием системы на базе нескольких машин, с разделением работы для повышения быстродействия, а также надежности систем управления длительными непрерывными технологическими процессами (за счет системной организации аппаратных средств и операционной системы).

Повидимому, перечень задач и перспектив развития «Электроники-100» мог бы быть дополнен и расширен. Однако всякое познание и выдвигаемые им задачи бесконечны, и если хотя бы на часть из первоочередных вопросов читатель нашел в книге ответ (или встречный вопрос), авторы считают свою задачу выполненной и в данном объёме.

Приложение 1

СИСТЕМА КОМАНД ЭВМ «ЭЛЕКТРОНИКА-100»



Код операции	Команда	Пояснение
0	ЛУ (логическое умножение)	$ HP \wedge Y \rightarrow HP$
1	СЛ (сложение)	$ HP + Y \rightarrow HP$
2	УП (увеличение и пропуск)	$ Y + 1 \rightarrow Y$; если новое $ Y = 0$, то $ СК + 1 \rightarrow СК$
3	ПС (посылка и сброс)	$ HP \rightarrow Y$; $0 \rightarrow HP$
4	ОП (обращение к подпрограмме)	$ СК \rightarrow Y$; $Y + 1 \rightarrow СК$
5	БП (безусловная передача управления)	$Y \rightarrow СК$

Примечания:

1. $|X|$ обозначает «содержимое (регистра или ячейки) X».
2. A_{i-j} обозначает «разряды $i-j$ регистра A».



6001	Разрешить прерывание	Выполняется еще одна команда и разрешается прерывание
6002	Запретить прерывание	
6031	Опрос флага ввода	Если флаг ввода в «1», то $ СК + 1 \rightarrow СК$
6032 ¹⁾	Очистка флага ввода и НР	$0 \rightarrow \text{флаг ввода};$ $0 \rightarrow НР$
6034	Считывание по правилам логического сложения	$\{ \text{Буфер ввода} \vee НР_{4-11} \rightarrow НР_{4-11}; НР_{0-3} \text{ не изменяется}$
6036 ¹⁾	Считывание с предварительной очисткой флага ввода и НР	$0 \rightarrow НР; 0 \rightarrow \text{флаг ввода};$ $ \text{Буфер ввода} \rightarrow НР_{4-11}$

¹⁾ И начало движения перфоленты при работе со считывателем: после считывания очередной строки п/д $1 \rightarrow \text{флаг ввода}$.

6041	Опрос флага вывода	Если флаг вывода в «1», то $ СК + 1 \rightarrow СК$
6042	Очистка флага вывода	$0 \rightarrow \text{флаг вывода}$
6046	Вывод на печать или перфорацию (тумблер ПЧ/ПР) с предварительной очисткой флага вывода	$ НР_{4-11} \rightarrow \text{буфер вывода и начало собственно вывода};$ по окончании вывода $1 \rightarrow \text{флаг вывода}$

При расширении комплектности машины:

6004	Запуск АЦП с предварительной очисткой НР	Код преобразованного аналогового сигнала $\rightarrow НР_{0-9}$
6133	Опрос и очистка флага таймера	Если флаг таймера в «1», то $ СК + 1 \rightarrow СК;$ $0 \rightarrow \text{флаг таймера}$
6137	Запуск таймера на очередной цикл	$ НР \rightarrow \text{регистр таймера, запуск таймера; когда регистр таймера } = 7777_8,$ $1 \rightarrow \text{флаг таймера}$
6313	Опрос и очистка флага входного коммутатора дискретных сигналов (ВКДС)	Если флаг ВКДС в «1», то $ СК + 1 \rightarrow СК;$ $0 \rightarrow \text{флаг ВКДС}$

Команды обращения к расширенной памяти

62N1	Установка поля данных	$N \rightarrow \text{регистр поля данных}$
62N2	Установка поля команд	$N \rightarrow \text{регистр поля команд — после первой (последующей) команды ОП или БП; до выполнения команд ОП или БП разрешенное ранее прерывание блокируется}$
6214	Считывание поля данных	$ РПД \vee НР_{6-8} \rightarrow НР_{6-8}^{1)}$
6224	Считывание поля команд	$ РПК \vee НР_{6-8} \rightarrow НР_{6-8}^{1)}$
6234	Считывание регистра запоминания поля	$ РЗП \rightarrow НР;$ код поля команд $\rightarrow НР_{6-8}$ код поля данных $\rightarrow НР_{9-11}$
6244	Восстановление полей памяти	$ РЗП \rightarrow РПД$ $\rightarrow РПК$

¹⁾ $НР_{6-8}$ и $НР_{9-11}$ не изменяются.

Коды «Консула-254»

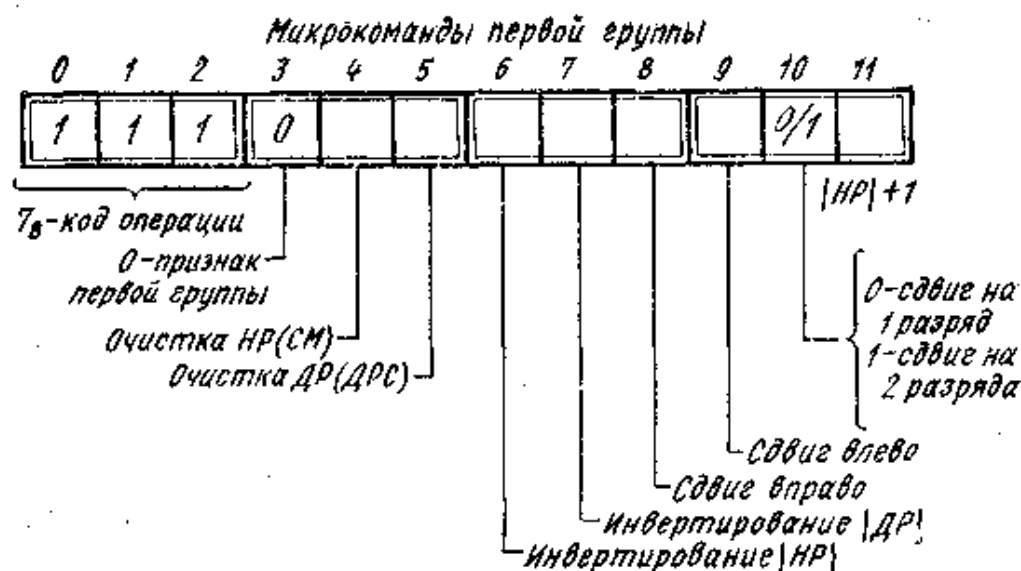
Старшие разряды	Младшие разряды							
	0	1	2	3	4	5	6	7
0 1	0] 8;	1, 9[	2. +—	3 ₁₀ — =	4↑ />	5()	6)	7×
2 3	Возврат каретки — *		Красная Черная лента лента		Верх- Табули- ний рование регистр		Нижний регистр	Про- бел
4 5	A— И	B% Й	BW K≤	ГG ЛL	ДD M≡	Е: НN	ЖV О	ЗZ П≡
6 7	PR Ш'	CS Щ'	T√ Ы	УU Ь^	ФF Э≥	X< Ю◇	Ц≠ ЯQ	Ч÷

1. На пересечении строки и столбца, номера которых образуют код символа, указан вначале символ верхнего регистра, а затем — нижнего.

2. Символы на НР имеют дополнительную пробивку по 7-й дорожке.

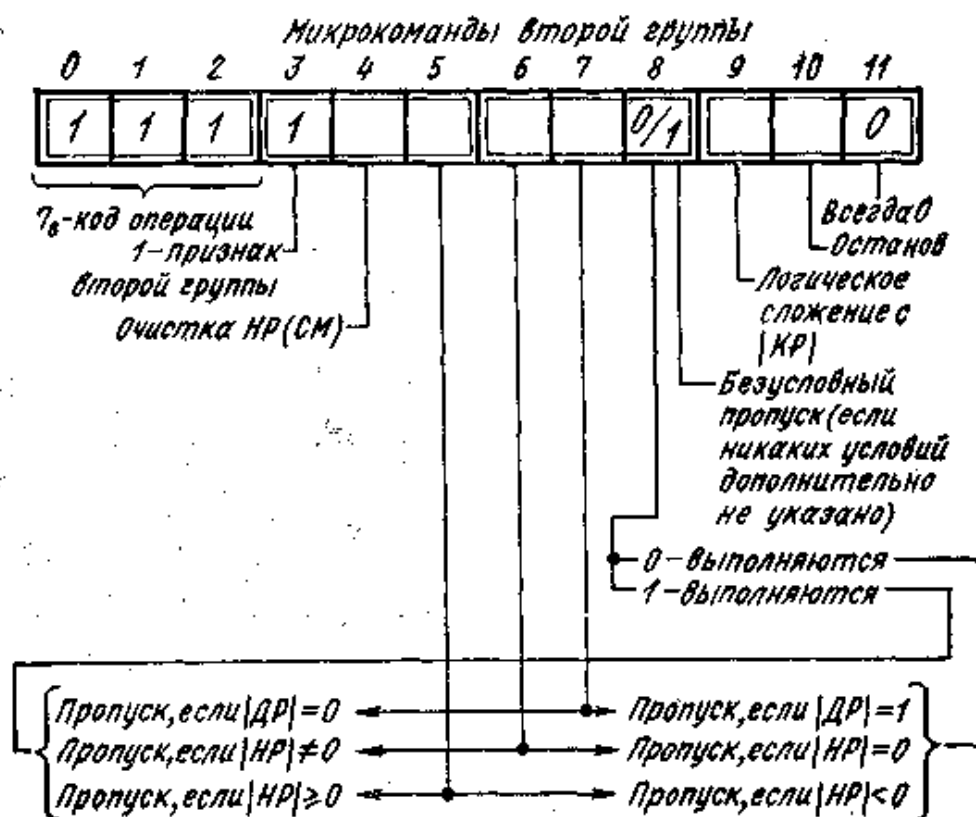
3. При контроле на четность используется 8-я дорожка (при четном числе единичных разрядов дополнительно пробивается 8-я дорожка).

Программа «ВВОД 1»							
7700	6032	7710	7006				
1	6031	1	6031				
2	5301	2	5311				
3	6036	3	6034				
4	7106	4	7420				
5	7006	5	3720				
6	7510	6	3320				
7	5301	7	5300				
		7720	0000				



7000	Нет операции	Задержка на один машинный цикл
7001	Увеличение содержимого НР (СМ) на 1	$ \text{НР} + 1 \rightarrow \text{НР}$
7004	Циклический сдвиг влево на 1 разряд	$ \text{НР}_i \rightarrow \text{НР}_{i-1}; \text{НР}_0 \rightarrow \text{ДР};$ $ \text{ДР} \rightarrow \text{НР}_{11}$
7006	Циклический сдвиг влево на 2 разряда	$ \text{НР}_i \rightarrow \text{НР}_{i-2};$ $ \text{НР}_0 \rightarrow \text{НР}_{11}; \text{ДР} \rightarrow \text{НР}_{10};$ $ \text{НР}_1 \rightarrow \text{ДР}$
7010	Циклический сдвиг вправо на 1 разряд	$ \text{НР}_i \rightarrow \text{НР}_{i+1}; \text{ДР} \rightarrow \text{НР}_0;$ $ \text{НР}_{11} \rightarrow \text{ДР}$
7012	Циклический сдвиг вправо на 2 разряда	$ \text{НР}_i \rightarrow \text{НР}_{i+2}; \text{НР}_{10} \rightarrow \text{ДР};$ $ \text{ДР} \rightarrow \text{НР}_1; \text{НР}_{11} \rightarrow \text{НР}_0$
7020	Инвертирование ДР (ДРС)	$ \overline{\text{ДР}} \rightarrow \text{ДР}$
7021	Инвертирование ДР (ДРС) и увеличение НР (СМ) на 1	$ \overline{\text{ДР}} \rightarrow \text{ДР}; \text{НР} + 1 \rightarrow \text{НР}$
7040	Инвертирование НР (СМ)	$ \overline{\text{НР}} \rightarrow \text{НР}$
7041	Образование дополнительного кода	$ \overline{\text{НР}} + 1 \rightarrow \text{НР}$
7060	Инвертирование НР и ДР	$ \overline{\text{НР}} \rightarrow \text{НР}; \overline{\text{ДР}} \rightarrow \text{ДР}$
7100	Очистка ДР (ДРС)	$0 \rightarrow \text{ДР}$
7101	Очистка ДР (ДРС) и увеличение содержимого НР (СМ) на 1	$0 \rightarrow \text{ДР}; \text{НР} + 1 \rightarrow \text{НР}$
7104	Циклический сдвиг влево на 1 разряд с предварительной очисткой ДР	$0 \rightarrow \text{ДР}; \text{НР}_i \rightarrow \text{НР}_{i-1};$ $ \text{НР}_0 \rightarrow \text{ДР}; \text{ДР} \rightarrow \text{НР}_{11}$
7106	Циклический сдвиг влево на 2 разряда с очисткой ДР (ДРС)	$0 \rightarrow \text{ДР}; \text{НР}_i \rightarrow \text{НР}_{i-2};$ $ \text{НР}_0 \rightarrow \text{НР}_{11}; \text{ДР} \rightarrow \text{НР}_{10}$

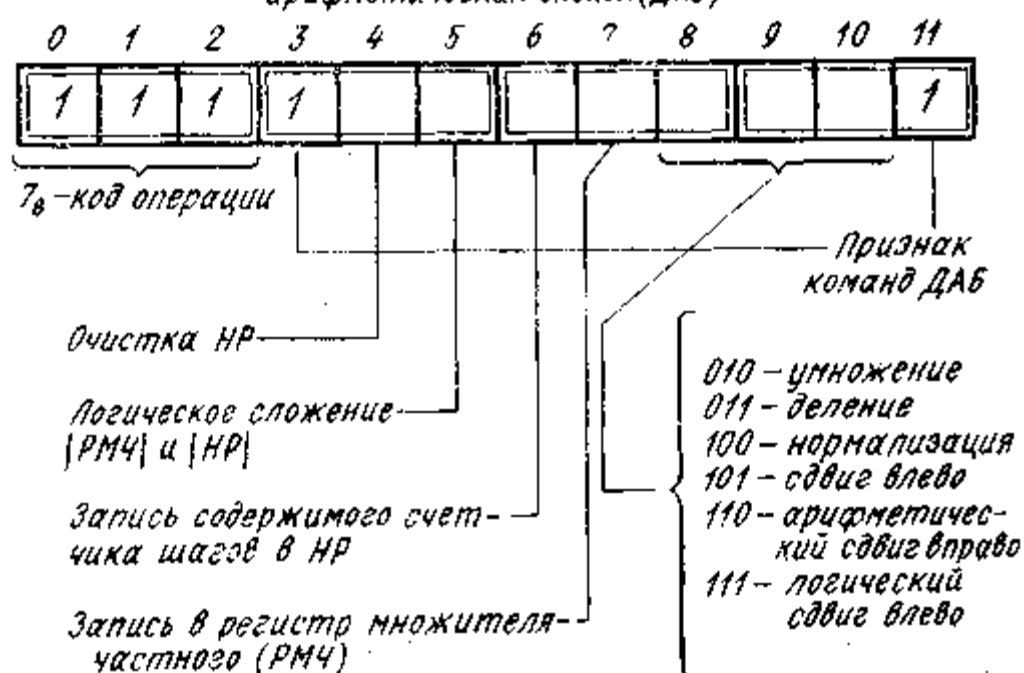
7110	Циклический сдвиг вправо на 1 разряд с предварительной очисткой ДР	$0 \rightarrow \text{ДР}; \text{НР}_i \rightarrow \text{НР}_{i+1}; \text{НР}_{11} \rightarrow \text{ДР}; \text{ДР} \rightarrow \text{НР}_0$
7112	Циклический сдвиг вправо на 2 разряда с очисткой ДР	$0 \rightarrow \text{ДР}; \text{НР}_i \rightarrow \text{НР}_{i+2}; \text{НР}_{11} \rightarrow \text{НР}_0; \text{ДР} \rightarrow \text{НР}_1$
7120	Запись «1» в ДР (ДРС)	$1 \rightarrow \text{ДР}$
7121	Запись «1» в ДР (ДРС) и увеличение содержимого НР (СМ) на 1	$1 \rightarrow \text{ДР}; \text{НР} + 1 \rightarrow \text{НР}$
7140	Очистка ДР и инвертирование НР	$0 \rightarrow \text{ДР}; \overline{ \text{НР} } \rightarrow \text{НР}$
7200	Очистка НР (СМ)	$0 \rightarrow \text{НР}$
7201	Запись кода 0001 в НР (СМ)	$0001 \rightarrow \text{НР}$
7204	Засылка ДР в 11-й разряд НР (СМ)	$0 \rightarrow \text{НР}; \text{ДР} \rightarrow \text{НР}_{11}; 0 \rightarrow \text{ДР}$
7206	Засылка ДР в 10-й разряд НР (СМ)	$0 \rightarrow \text{НР}; \text{ДР} \rightarrow \text{НР}_{10}; 0 \rightarrow \text{ДР}$
7210	Засылка ДР в 0-й разряд НР (СМ)	$0 \rightarrow \text{НР}; \text{ДР} \rightarrow \text{НР}_0; 0 \rightarrow \text{ДР}$
7212	Засылка ДР в 1-й разряд НР (СМ)	$0 \rightarrow \text{НР}; \text{ДР} \rightarrow \text{НР}_1; 0 \rightarrow \text{ДР}$
7240	Засылка «-1» в НР (СМ)	$7777 \rightarrow \text{НР}$
7300	Очистка НР (СМ) и ДР (ДРС)	$0 \rightarrow \text{НР}; 0 \rightarrow \text{ДР}$
7320	Очистка НР и запись «1» в ДР	$0 \rightarrow \text{НР}; 1 \rightarrow \text{ДР}$
7340	Очистка ДР и запись «-1» в НР	$0 \rightarrow \text{ДР}; 7777 \rightarrow \text{НР}$



7402	Останов	
7404	Логическое сложение $ НР $ с кодом, набран- ным на КР	$ НР + КР \rightarrow НР$
7410	Безусловный пропуск	$ СК + 1 \rightarrow СК$
7420	Пропуск, если $ ДР =1$	Если $ ДР =1$, то $ СК + 1 \rightarrow СК$
7430	Пропуск, если $ ДР =0$	Если $ ДР =0$, то $ СК + 1 \rightarrow СК$
7440	Пропуск, если $ НР =0$	Если $ НР =0$, то $ СК + 1 \rightarrow СК$
7450	Пропуск, если $ НР \neq 0$	Если $ НР \neq 0$, то $ СК + 1 \rightarrow СК$
7460	Пропуск, если $ НР =0$ и (или) $ ДР =1$	Если $ НР =0$ и $ ДР =0$, то $ СК + 1 \rightarrow СК$; если $ НР =0$ или $ ДР =1$ то $ СК + 1 \rightarrow СК$
7470	Пропуск, если $ НР =0$ и $ ДР =0$	Если $ НР =0$ и $ ДР =0$, то $ СК + 1 \rightarrow СК$
7500	Пропуск, если $ НР < 0$	Если $ НР < 0$, то $ СК + 1 \rightarrow СК$
7510	Пропуск, если $ НР \geq 0$	Если $ НР \geq 0$, то $ СК + 1 \rightarrow СК$
7520	Пропуск, если $ НР < 0$ и (или) $ ДР =1$	Если $ НР < 0$ и $ДР=1$, то $ СК + 1 \rightarrow СК$; если $ НР < 0$ или $ ДР =1$, то $ СК + 1 \rightarrow СК$
7530	Пропуск, если $ НР \geq 0$ и $ ДР =0$	Если $ НР \geq 0$ и $ ДР =0$, то $ СК + 1 \rightarrow СК$

7540	Пропуск, если $ HP \leq 0$	Если $ HP \leq 0$, то $ СК + 1 \rightarrow СК$
7550	Пропуск, если $ HP > 0$	Если $ HP > 0$, то $ СК + 1 \rightarrow СК$
7600	Очистка HP (CM)	$0 \rightarrow HP$
7604	Снятие информации с клавишного регистра	$ КР \rightarrow HP$
7610	Безусловный пропуск и очистка HP (CM)	$ СК + 1 \rightarrow СК; 0 \rightarrow HP$
7620	Пропуск, если $ ДР = 1$ и очистка HP (CM)	Если $ ДР = 1$, то $ СК + 1 \rightarrow СК$; всегда $0 \rightarrow HP$
7630	Пропуск, если $ ДР = 0$ и очистка HP (CM)	Если $ ДР = 0$, то $ СК + 1 \rightarrow СК$; всегда $0 \rightarrow HP$
7640	Пропуск, если $ HP = 0$ и очистка HP (CM)	Если $ HP = 0$, то $ СК + 1 \rightarrow СК$; всегда $0 \rightarrow HP$
7650	Пропуск, если $ HP \neq 0$ и очистка HP	Если $ HP \neq 0$, то $ СК + 1 \rightarrow СК$; всегда $0 \rightarrow HP$
7700	Пропуск, если $ HP < 0$ и очистка HP	Если $ HP < 0$, то $ СК + 1 \rightarrow СК$; всегда $0 \rightarrow HP$
7710	Пропуск, если $ HP \geq 0$ и очистка HP	Если $ HP \geq 0$, то $ СК + 1 \rightarrow СК$; всегда $0 \rightarrow HP$
7750	Пропуск, если $ HP > 0$ и очистка HP	Если $ HP > 0$, то $ СК + 1 \rightarrow СК$; всегда $0 \rightarrow HP$

Команды управления добавляемым арифметическим блоком (ДЛБ)



МАЛАЯ ОПЕРАЦИОННАЯ СИСТЕМА ДЛЯ ЭВМ «ЭЛЕКТРОНИКА-100»

Настоящая операционная система для ЭВМ «Электроника-100» с одним кубом МОЗУ, таймером и набором периферийных устройств обеспечивает мультипрограммную работу ЭВМ в реальном времени, ввод — вывод информации через пишущую машинку «Консул-254» параллельно с основными вычислениями, работу с периферийными устройствами, организующими запросы в ЭВМ по каналу прерывания, а также предоставляет пользователям эффективный способ кодирования печатных сообщений и бланков.

Собственно операционная система включает две части:

- программу-диспетчер;
- блок программного обеспечения ввода — вывода.

ПРОГРАММА-ДИСПЕТЧЕР

Отдельные функционально законченные участки общей программы рассматриваются диспетчером как самостоятельные задачи, имеющие свой начальный адрес и приоритет. Приоритеты нумеруются натуральными числами, начиная с 0 (высший приоритет). Любая задача, (кроме привилегированных¹) может начинаться только после её постановки в общую очередь и передачи ей управления при попадании её на первое место в основной очереди. По окончании задачи управление должно быть возвращено диспетчеру.

Заявки на решение той или иной задачи могут возникать в случайные моменты времени по сигналам прерывания извне и по требованиям служб диспетчера.

Обращение к программе-диспетчеру

Адреса для обращения в различные блоки диспетчера вынесены на нулевую страницу, что обеспечивает простоту обращения к диспетчеру из любого места основных функциональных программ системы.

1. По окончании решения задачи управление передается диспетчеру по команде «5576».
2. Для задержки дальнейшего выполнения задачи на $(m)_8$ интервалов² циклической работы таймера в основной программе необходимо выполнить следующие операции:

- вызов $(m)_8$ на СМ;
- обращение к диспетчеру по команде «4575».

По истечении задержки управление передается в следующую (за командой «4575») ячейку программы.

Для периодического выполнения задачи достаточно в конце её программы выполнить следующие операции:

- вызов $(m)_8$ на СМ;
- обращение к блоку службы времени по команде «4575»;
- передача управления в начало задачи.

3. Для вывода на печать обращение к диспетчеру имеет вид:

- команда «4574»;
- рабочая ячейка, содержащая адрес кодограммы³ печати.

По окончании вывода управление возвращается в ячейку, следующую за рабочей с адресом кодограммы.

Настройка диспетчера

В программе-диспетчер предусмотрены следующие настройки, важные при реализации конкретных систем:

- а) настройка службы времени;
- б) настройка блока обслуживания прерываний.

1. Настройка службы времени. Наименьший элементарный отрезок времени, который отсчитывает таймер, составляет 40 мс. В соответствии с этим машинную константу, соответствующую требуемому интервалу периодичности T_s прерываний от таймера, и константу пересчета минут определяют по следующим правилам:

- а) определяют целую часть частного T_s [мс] / 40 (обозначим её через n);
- б) переводят n в восьмеричный код;
- в) образуют от полученного кода обратный код и запасают его в ячейке 0577;
- г) корректируют константу счетчика минут;
- рассчитывают значение константы P по формуле

$$P = 1500/n;$$

¹ Привилегированными считаются задачи обслуживания очередных фрагментов ведущегося вывода (ввода) и внутренние задачи диспетчера.

² На m наложено ограничение $0 < m_8 \leq 7777$, что соответствует (при длительности интервала 80 мс) задержкам от 0 до 327,6 с = 5.46 мин. О настройках службы времени см. ниже.

³ Вид и способ составления кодограммы определяется используемым программным обеспечением ввода—вывода через «Консул-254» (см. ниже).

- переводят P в восьмеричный код;
- образуют от полученного кода дополнительный код и засылают его в ячейку 0571.

Код часов и код минут, накопленных к текущему моменту времени, хранятся в ячейках 0572 и 0573 соответственно; ячейку 0574 занимает счетчик числа интервалов T_s , прошедших после отсчета последней минуты; в ячейках 0575, 0576 располагается счетчик двухъячеичный общего числа прерываний от таймера. Адреса этих ячеек при настройках не изменяются.

Значение текущего времени (часы, минуты) при использовании средств программного обеспечения диалога (см. ниже) может по запросу оператора вызываться на печать в десятичной форме. Там же имеется команда установки текущего состояния счетчиков службы времени в соответствии с указываемым оператором текущим значением времени.

2. Настройка блока обслуживания прерываний состоит в уточнении номеров подключенных внешних устройств, присылающих свои запросы по каналу прерывания, и заполнении таблицы характеристик запрашиваемых задач. В исходной редакции программа-диспетчер после прерывания опрашивает флаги таймера, входного коммутатора дискретных сигналов, устройств вывода и ввода, а затем флаги внешних устройств с номерами от 14₈ до 17₈. Таким образом, кроме таймера и устройств ввода — вывода, по каналу прерывания могут присылать свои запросы еще пять внешних устройств. Номера соответствующих селекторов, используемых в общей системе, и адреса констант, характеризующих задачи, которые вызываются по запросам этих устройств, сведены в таблицу.

Номер внешнего устройства, присылающего запрос по каналу	Адрес расположения характеристик запрашиваемой задачи	
	начальный адрес	приоритет
34 ₈ (входной коммутатор дискретных сигналов)	0357	0352
14 ₈	0360	0353
15 ₈	0361	0354
16 ₈	0362	0355
17 ₈	0363	0355

Когда подключены не все из названных селекторов, соответствующую команду типа «бпм3» (где nm — восьмеричный номер внешнего устройства) в ячейках 0205 — 0240 следует заменить на «7000». Если подключены внешние устройства с другими номерами селекторов, следует заменить одну из команд типа «бпм3» на аналогичную, но с номером подключенного селектора. Начальный адрес и приоритет задачи, инициируемой этим внешним устройством, помещают в ячейки, соответствующие (по таблице) внешнему устройству, номер которого изменяется.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВВОДА - ВЫВОДА ¹

Программное обеспечение ввода — вывода реализует связь оператора с ЭВМ «Электроника-100» через «Консул-254» в процессе функционирования системы управления. Оно состоит из двух частей:

- а) программное обеспечение диалога (ПОД);

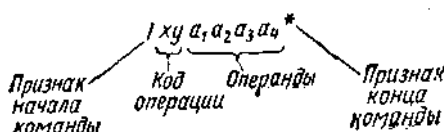
- б) программы печати бланков (ППБ).

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДИАЛОГА (ПОД)

ПОД позволяет производить следующие операции:

- а) запись восьмеричных и десятичных кодов в ячейки, указываемые абсолютными или символическими адресами;
- б) вывод на печать содержимого отдельных ячеек, указываемых абсолютными или символическими адресами в восьмеричном или десятичном виде;
- в) установку и индикацию текущего состояния таймера;
- г) инициирование печати бланков.

Каждая операция реализуется путем набора на клавиатуре «Консула-254» текста нужной команды. Общая структура команд оператора имеет вид:



В настоящем варианте операционной системы реализовано 12 команд:

- /C1* — выдать сводку 1,
- /C2* — выдать сводку 2,
- /IT* — индикация таймера,
- /UT $a_1 a_2 - a_3 a_4$ * — установить таймер ($a_1 a_2$ — часы; $a_3 a_4$ — минуты),
- /ЛВ $a_1 \dots a_n$ * — печать восьмеричная содержимого ячеек, указанных операндами $a_1, \dots, a_n$; всего может быть указано до трех адресов,
- /ПД $a_1 \dots a_n$ * — печать десятичная (роль операндов та же, что у предыдущей команды),
- /ЗВ $a_1 \dots a_4$ В $b_1 \dots b_4$ * — запись восьмеричная кода $a_1 a_2 a_3 a_4$ в ячейку $b_1 b_2 b_3 b_4$,
- /ЗД $a_1 \dots a_4$ В $b_1 \dots b_4$ * — запись десятичная кода $a_1 a_2 a_3 a_4$ в ячейку $b_1 b_2 b_3 b_4$,
- /ЧВ $a_1 \dots a_n$ * — чтение восьмеричное параметров $a_1 \dots a_n$ (параметр указывается буквой с двумя последующими цифрами); одновременно может быть указано до четырех параметров,
- /ЧД $a_1 \dots a_n$ * — чтение десятичное параметров (роль операндов та же, что у предыдущей команды),
- /УВ $a_1 a_2 a_3 - b_1 \dots b_4$ * — установка восьмеричная параметра $a_1 a_2 a_3$ в значение $b_1 b_2 b_3 b_4$
- /УД $a_1 \dots a_4 - b_1 \dots b_4$ * — установка десятичная параметра $a_1 a_2 a_3 a_4$ в значение $b_1 b_2 b_3 b_4$,

Примечания. 1. В любом месте команды допустимо введение произвольного числа пробелов. 2. Команда может быть разбита на части возвратом каретки. 3. В ответ на неправильный набор команды машина печатает «!». 4. Информация, вводимая без признака начала команды, игнорируется. 5. При вводе признака начала команды ранее введенный массив

¹ Разработано А. С. Ждановым.

ПРОГРАММЫ ПЕЧАТИ БЛАНКОВ (ППБ)

ППБ обеспечивает печать информационных бланков и сообщений. Каждому бланку и сообщению соответствует определенный упорядоченный набор кодов (кодограмма), который однозначно определяет подлежащий печати текст.

В состав ППБ входит программа управления печатью (УП) и исполнительные программы, реализующие различные виды печати: программы печати дробно-десятичных одно- и двухъязычных чисел, программы печати группы упакованных символов, программа вывода отдельного символа, программа печати даты и времени, программа перехода на другой сегмент кодограммы (расположенный в ином участке памяти) и программа, осуществляющая включение другой кодограммы в данную.

Программа управления печатью осуществляет просмотр специфицированной кодограммы и последовательно, в зависимости от кодов кодограммы, передает управление исполнительным программам, которые после реализации необходимых операций возвращают управление УП. Данный процесс завершается, когда УП сталкивается с управляющим кодом 3700, принадлежащим исходной кодограмме: при этом УП передает управление в ту точку, откуда была инициирована печать. Если код 3700 относится к кодограмме, включенной в рассматриваемую, то осуществляется возврат в исходную кодограмму.

Кодограмма состоит из одного или нескольких массивов кодов, исчерпывающим образом определяющих для УП последовательность и начальные условия запуска исполнительных программ, необходимых для печати соответствующего текста. Каждый массив кодограммы представляет собой последовательность смежных ячеек, содержащих упорядоченный набор управляющих кодов, включающих одну или несколько ячеек. Первая половина первой ячейки каждого такого кода определяет тип соответствующей операции, а остальные разряды кода служат начальной информацией для связанной с данной операцией исполнительной программы.

Восьмеричные управляющие коды, используемые в кодограммах, имеют вид:

а) однаждыязычные коды:

03 *ab* — вывод на «Консул» символа с кодом *ab*,

07 *xy* — печать *xy* пробелов, времени и даты ($xy < 37_8$).

3700 — признак окончания кодограммы.

б) двухъязычные коды:

00 xy 0000 — печать xy пробелов.

$a_1a_2a_3a_4$ ($1 \leq a \leq 4$; $a > b$) — печать дробно-десятичного числа, записанного в ячейке с адресом $a_1a_2a_3a_4$ в виде a цифр, из которых b цифр справа отделены запятой, и xy пробелов.

$a(b+4)xy$ $a_1a_2a_3a_4$ ($a = 4$; $0 \leq b \leq 3$) — печать дробно-десятичного числа, записанного в двух ячейках, начиная с адреса $a_1a_2a_3a_4$, в виде a цифр, из которых b цифр справа отделены запятой, и xy пробелов.

0500 $a_1a_2a_3a_4$ — переход на другую кодограмму по адресу $a_1a_2a_3a_4$ с возвратом после её обработки в исходную кодограмму.

0600 $a_1a_2a_3a_4$ — переход на другой сегмент кодограммы по адресу $a_1a_2a_3a_4$.

0400 $a_1a_2a_3a_4$ — печать группы упакованных символов, записанных, начиная с адреса $a_1a_2a_3a_4$.

в) трехъязычные коды:

01 xy $a_1a_2a_3a_4$ $abcd$ — печать строки cd дробнодесятичных (однаждыязычных) чисел с xy пробелами между ними, записанных в ячейках, начиная с адреса $a_1a_2a_3a_4$ в виде a цифр, из которых b цифр справа отделены запятой.

02 xy $a_1a_2a_3a_4$ $abcd$ — печать строки cd дробнодесятичных двухъязычных чисел с xy пробелами между ними, записанных в ячейках, начиная с адреса $a_1a_2a_3a_4$, в виде a цифр, из которых b цифр справа отделены запятой.

г) код переменной длины.

Имеется операция печати упакованных символов (по два в одну ячейку), помещенных непосредственно за управляющим кодом (17 xy). По окончании печати делается xy пробелов. Конец последовательности упакованных символов определяется признаком (37). Следующая за этим признаком ячейка рассматривается УП как первая ячейка очередной группы управляющих кодов;

д) печать чисел со знаком.

Программы печати учитывают знак числа, если старший разряд второй половины первой управляющей ячейки соответствующей группы находится в состоянии «1», т. е. (xy) = 40 + a , где $a \geq 0$. Таким образом, число пробелов, задаваемое цифрами (xy), не может превышать 37₈.

Составление кодограмм. Для написания кодограммы необходимо разбить соответствующий бланк на части, каждая из которых может быть напечатана с помощью одного из описанных управляющих кодов. Вначале следует закодировать первую строку бланка, затем вторую и т. д., двигаясь по строкам слева направо. Соответствующие управляющие коды должны помещаться в последовательные ячейки памяти в том же порядке, в каком осуществляется кодирование бланков. Если возникает необходимость расположить кодограмму в нескольких несмежных областях памяти, то в конце каждого отдельного сегмента кодограммы (за исключением последнего) нужно поместить управляющий код «0600 $a_1a_2a_3a_4$ », реализующий переход (без возврата) на следующий сегмент. Если целесообразно включить в данную кодограмму другую (когда последняя описывает печать, являющуюся составной частью рассматриваемого бланка), то следует использовать управляющий код «0500 $a_1a_2a_3a_4$ », обеспечивающий переход с возвратом. Каждая кодограмма должна завершаться признаком конца — кодом «3700».

Пример. Предположим, что необходимо обеспечить печать бланка следующего вида:

Показатели

xxx, xppxxx, xppxxx, xppp±xxxppppx, xhpx, xx



$Q = \pm XX, XXpppA, = XX XXpppppppppppppppp$ ч/мин — день/мес.

Здесь П означает пробел, Х представляет произвольную десятичную цифру, а прямоугольник обозначает некоторый текст, кодограмма которого уже записана и имеет начальный адрес 3500. Допустим, что первые три числа представлены в памяти двухъязычными кодами, занимающими ячейки 4010 — 4011, 4012 — 4013 и 4014 — 4015 соответственно; четвертое число формируется из одноязычного кода, расположенного по адресу 7501; пятому и шестому числам соответствуют одноязычные коды с адресами 7603 и 7604; седьмое число определяется по одноязычному коду, записываемому в ячейку 3567, а восьмое — по двухъязычному коду с адресом 5020. Такой бланк может быть описан следующей кодограммой:

Адрес	Содержимое	Адрес	Содержимое
3400	0320	4	1700
1	0017	5	2026
2	0000	6	7613
3	0400	7	2437
4	3770	3430	4243
5	0320	1	3567
5	0320	2	0340
7	0202	3	0326
3410	4010	4	0313
1	4103	5	0324
2	0001	6	4411
3	0000	7	5020
4	3044	3440	0700
5	7501	1	0320
6	0101	2	3700
7	7603	3770	5756
3420	3202	1	5240
1	0320	2	4740
2	0500	3	6245
3	3500	4	5350
		5	3700

Настраиваемые адреса программы диалога:

6124	Содержит адрес кодограммы, соответствующей сводке 1.
6164	Содержит адрес кодограммы, соответствующей сводке 2.
6100—6137	Таблица символических обозначений параметров.
6140—6177	Каждая ячейка таблицы хранит код символа параметра в младших разрядах. Таблица адресов исходных кодов параметров. Каждый адрес таблицы указывает местоположение исходных кодов соответствующей группы параметров, обозначенных общим символом.
6567	Содержит восьмеричный код количества параметров, используемых в конкретной системе.

ОГРАНИЧЕНИЯ ОПЕРАЦИОННОЙ СИСТЕМЫ, ИНДИКАЦИЯ НАРУШЕНИЙ, ЗАПУСК

1. Программа-диспетчер занимает ячейки 0000, 0001, 0173 — 1377, 3200 — 3377; программное обеспечение ввода — вывода — ячейки 5200 — 7377.
2. Число мест:
 - в очереди № 1 — 15₁₀; в очереди № 2 — 5;
 - в очереди задержек — 12₁₀; в очереди вывода — 11₁₀;
 - число внешних устройств (помимо таймера и «Консула-254»), которые могут обращаться к диспетчеру по каналу прерывания — 5.
3. При переполнении очередей диспетчер теряет заявку и печатает в экстренном порядке (по окончании очередной печати):
 - ОТКАЗ 1 — xxxx — при отказе в очереди № 1;
 - ОТКАЗ 2 — xxxx — при отказе в задержке;
 - ОТКАЗ 3 — xxxx — при отказе на вывод, где xxxx — адрес задачи, получившей отказ.
4. Стартовый адрес программы-диспетчера 0400.
5. Предусмотренный останов в ячейке 0241 происходит, если оказалось, что после прерывания ни один из флагов внешних устройств не установлен в единичное состояние.
6. При передаче управления задачам из очереди № 1 СМ и ДРС содержат случайные коды.
7. Перед обращением к службе времени или к службе вывода программы основного функционального обеспечения должны сами запомнить содержимое регистров и индексных ячеек, необходимых для продолжения нормальной работы после возвращения им управления от соответствующей службы диспетчера.
8. Если две или более задач с различными приоритетами используют некоторые общие подпрограммы или внешние устройства (помимо пишущей машинки), то необходимо дополнять операционную систему блоком «узкого места», подобного по своей структуре блоку вывода диспетчера, либо (что менее предпочтительно) запрещать прерывание (или повышать приоритет своей задачи — первой задачи из очереди № 1) на время прохождения такого «узкого места» с последующим разрешением прерывания (восстановлением исходного приоритета своей задачи).

ТЕКСТ ПРОГРАММЫ

Список дополнительных сокращенных обозначений, принятых в пояснениях к тексту программы:

- ГД — главный диспетчер (программные блоки диспетчера, входящие в настоящую операционную систему);
- ВК — возврат каретки;
- ВР — верхний регистр;
- ППГЧ — программа печати группы чисел.

Адрес	Содержимое	Пояснения
0000	0752	Рабочая ячейка адреса продолжения прерванной задачи
1	5377	БП в блок обнаружения источника прерывания
0173	0000	Рабочая ячейка вида печати
4	1000	Адрес блока печати
5	0505	Адрес блока установки задержки
6	0642	Адрес блока окончания задач
7	7000	Константа

Опрос флагов и рост очереди № 1					
0200	3377	Запоминание [СМ] и [ДРС]	1	1747	
1	7004		2	7041	
2	3376		3	1350	
3	5321	На обнуление признаков	4	7700	
4	6133	Опрос флага таймера	5	5271	
5	5207		6	1746	
6	5737		7	7650	Контроль наличия места в очереди № 2
7	3375	0→счетчик пройденных позиций	0270	5745	
0210	6343	Опрос флага ВКДС	1	1744	
1	5213		2	7640	Возвращение заявки в конец очереди № 1, если очередь № 2 заполнена
2	5242		3	5303	
3	6041	Опрос флага вывода	4	2342	
4	5216		5	1351	
5	5770		6	3742	
6	6031	Опрос флага ввода	7	2341	
7	5222		0300	1350	Обращение к блоку экстренного вывода
0220	6032		1	3741	
1	5771		2	5305	
2	2375		3	4743	
3	6143		4	0351	
4	5216	Опрос флагов других ВУ и счет пройденных позиций	5	7200	
5	5242		6	1366	Анализ изменений в очередях № 1 и № 2
6	2375		7	7650	
7	6153		0310	5314	
0230	5232		1	1365	
1	5242		2	7640	
2	2375		3	5333	
3	6163		4	1740	
4	5236		5	7640	
5	5242		6	1000	
6	2375		7	3740	
7	6173		0320	5330	
0240	7402	Останов — неизвестное прерывание	1	3366	Обнуление признаков роста очередей
1	7000		2	3365	
2	1373		3	5204	Пересылка адреса запроса в ячейку 0000, если очередь № 1 пуста
3	1375		4	1351	
4	3372		5	3000	
5	1772	Выборка из таблицы исходных данных $A_{нач}$ и приоритета заявки	6	5274	
6	3350		7	7000	
7	1374		0330	1376	Восстановление [СМ] и [ДРС]
0250	1375		1	7110	
1	3372		2	1377	Изменяемые ячейки блока
2	1772		3	7000	
3	3351	Сравнение приоритетов и размещение запроса в очереди № 1	4	7000	Разрешение прерывания и выход из ГД
4	2366		5	6001	
5	7200		6	5400	Адрес службы времени
6	1740		7	0400	
7	7650				
0260	5324				

0340	3244		7	0000	
1	3263		0360	0000	Таблица начальных
2	3243		1	0000	адресов обслужи-
3	1212	Адреса таблиц и кон-	2	0000	вания запросов по
4	3262	станты очередей № 1	3	0000	прерываниям извне
5	0600	и № 2	4	0000	
6	3376		5	0000	Рабочие ячейки при-
7	3264		6	0000	знаков
			7	0000	
0350	0000	Рабочие ячейки за-	0370	6247	
1	0000	проса	1	5630	
2	0000		2	0000	
3	0000	Таблица приорите-	3	0352	Константы и рабочие
4	0000	тов запросов	4	0357	ячеек
5	0000		5	0000	
6	0000		6	0000	
			7	0000	

Служба времени

0400	6771	Запуск таймера на	0440	1356	
1	1377	очередной цикл	1	3754	
2	6137		2	1762	
3	2376		3	7650	
4	5206		4	5255	Цикл опроса регист-
5	2375		5	2762	ров поля задержек,
6	7000		6	5255	досчета ненулевых ре-
7	2374		7	1761	гистров, восстановле-
0410	7200	Счет импульсов, ми-	0450	3752	ния задач в очереди
1	1374	нут, часов	1	1760	№ 1, если необходи-
2	1371		2	3751	мая задержка истек-
3	7640		3	3761	ла
4	5225		4	5750	
5	3374		5	2360	
6	2373		6	2361	
7	1373		7	2362	
0420	1370		0460	7200	
1	7640		1	2366	
2	5225		2	5242	
3	3373		3	1177	Восстановление бло-
4	2372		4	3754	ков ГД
5	1367		5	1177	
6	3366		6	3755	
7	1365	Подготовка цикла	7	5747	БП в ГД
0430	3362		0470	1704	
1	1364		1	3000	Уменьшение очере-
2	3361		2	7240	ди № 1
3	1363		3	1702	
4	3360		4	3702	
5	7000		5	7240	
6	1357	Изменения блоков ГД	6	1703	
7	3755		7	3703	

Обращение к службе задержек

0500	5701		Запрос на за-	→ 5	0000	← Адрес возврата
1	0731		держку	→ 6	6002	
2	0341			→ 7	7041	
3	0342					
4	3245			0510	3346	
				1	1367	

2	3366		5	3264	
3	1365		6	0000	
4	3362		7	0305	
5	1364	Исходные операции в			
6	3361	подготовка цикла	0550	0254	
7	1363		1	0350	
			2	0351	
0520	3360		3	0000	Адреса, константы и
1	1761		4	0334	рабочие ячейки
2	7650		5	0333	
3	5334	Цикл поиска свобод-	6	0455	
4	2360	ного места	7	5734	
5	2361				
6	2362		0560	0000	
7	2366		1	0000	
			2	0000	
0530	5321		3	3230	
1	4744	Обращение к блоку	4	3214	
2	3244	экстренного вывода	5	3200	
3	5342		6	0000	
4	1346		7	7764	
5	3762	Занесение запроса в	0570	7704	
6	1305	таблицы	1	7603	
7	3761		2	0000	Счетчик часов
			3	0000	Счетчик минут
0540	1745		4	0000	Счетчик 0,5 с
1	3760		5	0000	Счетчик импульсов
2	5576	«Конец задачи»	6	0000	таймера (младшие
3	0773				разряды)
4	1212		7	7763	Константа цикла
					таймера
Рост очереди № 2					
0600	2373		2	1000	
1	1777		3	3773	Запоминание адреса
2	3773	Запоминание JCM и	4	2373	продолжения и прио-
3	2373	IDPC в очереди № 2	5	1767	ритета прерываемой
4	1776		6	3773	задачи
5	3773		7	2754	Засылка признака
6	2373				роста очереди № 2
7	1372		0630	1774	Помещение запроса в
			1	3767	начало очереди № 1
0610	3371	Подготовка цикла	2	1775	и уход на его обслу-
1	1372		3	4235	живание
2	7041		4	5765	
3	3370		5	0000	
4	1770		6	3766	Подпрограмма за-
5	3773		7	1766	сылки адреса продол-
6	2373	Цикл запоминания			жения задачи из оче-
7	2370	содержимого вв-			реды № 1 в ячейку
0620	2371	дексных ячеек			0000
1	5214				
Уменьшение очередей					
0640	3000		5	7650	
1	5635		6	5321	Анализ наличия не-
2	6002		7	1763	решенных задач в
3	7200		0650	7650	очередях № 1 и № 2
4	1764		1	5257	

2	1762	Сравнение приори-	4	3360	
3	7041	тетов второй задачи	5	1362	
4	1773	из очереди № 1 и по-	6	3370	
5	7740	следней из очереди	7	5730	
6	5321	№ 2			Уход на участок бло-
7	7000		0730	0470	ка уменьшения очере-
			1	7240	ди № 1
			2	1360	
0660	7000		3	3355	Цикл сдвига очереди
1	1373		4	1760	№ 1
2	1361	Переадресация оче-	5	3755	
3	3373	реди № 2 и предва-	6	7240	
4	1373	рительная пересыл-	7	1370	
5	3360	ка [СМ] в [ДРС]			
6	2360		0740	3355	Подготовка цикла
7	1760		1	1770	
			2	3755	
0670	3777		3	2360	
1	2360		4	2370	
2	1760	Подготовка цикла	5	2371	
3	3776		6	5331	
4	1372		7	1766	
5	3371				
6	1372		0750	7640	Анализ необходи-
7	7011		1	5234	мости холостого хода
			2	5753	
			3	1364	
0700	3370		4	0365	
1	7000		5	0000	
2	7000		6	7761	
3	2360		7	0330	
4	1760	Цикл восстано-	0760	0000	Адреса, константы в
5	3770	вления содержимого ин-	1	7764	рабочие ячейки
6	2370	дексных ячеек	2	3265	
7	2371		3	3245	
			4	3316	
0710	5303		5	0333	
1	2360		6	3244	
2	1760	Пересылка адреса	7	3264	
3	4235	продолжения и прио-	0770	0000	
4	3760	ритета продолжзе-	1	0000	
5	2360	мой задачи в начало	2	7770	
6	1760	очереди № 1 и уход	3	3303	
7	3767	на ее обслуживание	4	0350	
			5	0351	
0720	5757		6	0376	
1	1356		7	0377	
2	3371	Подготовка цикла			
3	1363				
Блок вывода					
← 1000	0000	← Адрес возврата	1	2200	Возвращение управ-
1	6002		2	5600	ления, минуя вывод
запрос	2	Анализ наличия сво-	3	7000	Рабочая ячейка
на вы-	7200	бодного места в оче-	4	1377	
вод	3	реди вывода	5	1375	
	4		6	3372	
	5		7	1200	
	6				
	7		1020	3772	
	4773		1	1377	
1010	3244				

2	1374		1110	1374	
3	3372	Запоминание за-	1	3356	
4	1771	проса и рост очереди	2	1355	
5	3772	вывода	3	3372	
6	2377		4	2372	
7	1173		5	1772	
			6	3755	
			7	1356	
1030	7640	Анализ:			Цикл сдвига очереди
1	5576	«канал свободен»?,	1120	3372	вывода
2	2370	начало вывода, свя-	1	2372	
3	7201	тые задачи из очере-	2	1772	
4	3173	ди № 1	3	3756	
5	1775		4	2355	
6	3372		5	2356	
7	5252		6	2354	
			7	6312	
1040	7200		1130	7240	Уменьшение счет-
1	1370		1	1377	чика занятых мест
2	7650		2	3377	на 1
3	5767	(с использованием	3	1753	
4	3370	признака вида выво-	4	7640	Нужен экстренный
5	7240	да и признака выво-	5	5752	вывод?
6	1173	да первого символа)	6	1377	Очередь вывода пу-
7	7640		7	7640	ста?
1050	5766				
1	5576		1140	5233	
2	1772		1	3173	Обнуление призна-
3	4751		2	5767	ка «канал занят»
4	6042				БП на прод. задачи
5	7240				из оч. № 1
6	1173	Анализ вида вывода	3	7201	
7	7640		4	1173	Окончен вывод по
			5	7650	запросу оператора?
1060	5343		6	5333	Обнуление призна-
1	1774		7	3753	ка «нужен экстрен-
2	3765				ный вывод»
3	1775		1150	5336	БП на обслуживание
4	7001		1	7210	обычных запросов
5	3764				
6	1360	Восстановление за-	2	1312	
7	3762	дачи, окончившей	3	1200	
		вывод, в очереди № 1	4	0000	
1070	1361		5	0000	
1	3763		6	0000	
2	5757		7	0254	
3	7000		1160	5734	
4	7200		1	1074	
5	1177		2	0333	Адреса, константы в
6	3762		3	0334	рабочие ячейки
7	1177		4	0351	
			5	0350	
1100	3763		6	1306	
1	7000		7	0305	
2	7000		1170	0000	
3	7201		1	3264	
4	1376		2	0000	
5	3354	Подготовка цикла	3	1212	
6	1375				
7	3355				

4 1330
5 1346

6 7762
7 0000

Блок экстренного вывода

1200	0000	Признак «нужен экстренный вывод»	1260	1777	Выявление адреса отказанной задачи и формирование текста
1	2024		1	0374	
2	5662		2	7104	
3	5240	Кодограмма упакованного текста печати при отказах	3	7006	
4	4727		4	1210	
5	0000		5	3210	
6	1327		6	1777	
7	0000		7	0373	
1210	0000		1270	7112	
1	2037		1	7012	
2	0000	Адрес возврата	2	7012	
3	7200		3	3207	
4	1200	Нет ли уже запроса?	4	1777	
5	7640		5	0177	
6	5306		6	7112	
7	2200	+1 в признак	7	7010	
1220	7201		1300	1207	
1	3377		1	3207	
2	1212		2	7000	
3	7041		3	1173	
4	1175		4	7650	
5	7700	Анализ вида (№) отказа и формирование текста	5	5310	
6	5243		6	2212	
7	2377		7	5612	
1230	1212		1310	2772	
1	7041		1	7000	
2	1174		2	7201	Анализ: «канал свободен»? и начало экстренного вывода
3	7700		3	7001	
4	5243		4	3173	
5	2377		5	1371	
6	1212		6	4770	
7	7041		7	5720	
1240	1214		1320	1054	
1	7750		1	0000	
2	2377		2	0000	
3	1377		3	0000	Резерв
4	7106		4	0000	
5	7006		5	0000	
6	7006		6	0000	
7	1376		7	0000	
1250	3205		1330	0000	
1	7000		1	0000	
2	7000		2	0000	
3	1612		3	0000	
4	3377		4	0000	Таблица приоритетов в очереди вывода
5	1777		5	0000	
6	0375		6	0000	
7	3210		7	0000	

1340	0000
1	0000
2	0000
3	0000
4	0000
5	0000
6	0000
7	0000

1360	0000
1	0000
2	0000
3	7200
4	1767
5	6001
6	5365
7	0000

Холостой ход с индикацией содержания ячейки, адрес которой запасен в 1367

1350	0000
1	0000
2	0000
3	0000
4	0000
5	0000
6	0000
7	0000

Таблица адресов в очереди вывода

1370	6253
1	1201
2	1170
3	0700
4	0070
5	0007
6	0027
7	0000

Адреса, константы, рабочие ячейки

Очередь задержек (3200—3243)

Очередь № 1 (3244—3303)

3200	0000
1	0000
2	0000
3	0000
4	0000
5	0000
6	0000
7	0000

3240	0000
1	0000
2	0000
3	0000
4	0000
5	0000
6	0000
7	0000

3210	0000
1	0000
2	0000
3	0000
4	0000
5	0000
6	0000
7	0000

Регистры задержек

3250	0000
1	0000
2	0000
3	0000
4	0000
5	0000
6	0000
7	0000

Адреса задач в очереди № 1

3220	0000
1	0000
2	0000
3	0000
4	0000
5	0000
6	0000
7	0000

Адреса продолжения

3260	0000
1	0000
2	0000
3	0000
4	0100
5	0100
6	0100
7	0100

Константа

3230	0000
1	0000
2	0000
3	0000
4	0000
5	0000
6	0000
7	0000

Приоритеты задач

3270	0100
1	0100
2	0100
3	0100
4	0100
5	0100
6	0100
7	0100

Приоритеты задач в очереди № 1

Очередь № 2 (3303—3377)

3300	0100 ↑		3340	0000 ↑	
1	0100		1	0000	
2	0100		2	0000	
3	0100	Константа	3	0000	Место для информа-
4	0000	[СМ]	4	0000	ции о третьей пре-
5	0000	[ДРС]	5	0000	рванной задаче
6	0000		6	0000	
7	0000		7	0000	
3310	0000	Содержимое	3350	0000	
1	0000	индексных	1	0000	
2	0000	ячеек	2	0000	
3	0000		3	0000	
4	0000		4	0000	
5	0000		5	0000	
6	0000	Адрес продолжения	6	0000	Запоминаемая ин-
7	0000	Приоритет	7	0000	формация о четвер-
3320	0000		3360	0000	той задаче
1	0000		1	0000	
2	0000		2	0000	
3	0000		3	0000	
4	0000		4	0000	
5	0000		5	0000	
6	0000	Место для информа-	6	0000	
7	0000	ции о второй пре-	7	0000	
3330	0000	рванной задаче	3370	0000	
1	0000		1	0000	Информация для пя-
2	0000		2	0000	той задачи
3	0000		3	0000	
4	0000		4	0000	
5	0000		5	0000	
6	0000		6	0000	
7	0000		7	0000	

Исполнительные программы блока печати

5200	0014		5220	7200	
1	7671		1	1204	
2	7212		2	3606	
3	0000	Константы и рабочие	3	1203	
4	0000	ячейки	4	3605	
5	7210		5	5602	
6	7207		6	3201	
7	0000		7	2201	Программа перехода
5210	3207				на другой сегмент
1	1606		5230	1601	кодограммы
2	3204		1	3606	
3	1605		2	5602	
4	3203		3	6341	
5	2207		4	0000	
6	1607	Программа для вклю-	5	3234	
7	4605	чения другой кодо-	6	7240	
		граммы в данную	7	1606	

5240	3606		5320	0000	
1	1634		1	0000	Рабочие ячейки
2	4303		2	0000	
3	4633		3	0000	
4	1200		4	0000	
5	4665		5	0000	
6	1266		6	1724	
7	3670		7	3321	
5250	1267	Программа печати времени и даты	5330	2324	Программа для печат- та группы одно- или двухъядерных чисел (ППГЧ)
1	3671		1	1724	
2	4633		2	3322	
3	1272		3	2324	
4	3670		4	1724	
5	1273		5	3323	
6	3671		6	1323	
7	1274		7	0300	
5260	4665		5340	7041	
1	1274		1	3316	
2	4665		2	0000	
3	5602		3	1722	
4	7000		4	4715	
5	6243		5	2322	
6	6127		6	1323	
7	6130		7	4675	
5270	6334	Рабочие ячейки и константы	5350	1321	
1	6335		1	4303	
2	0572		2	2316	
3	0573		3	5342	
4	0027		4	2606	
5	6717		5	5725	
6	6317		6	1322	
7	0000		7	4775	
5300	0037		5360	2322	
1	6206		1	2322	
2	0027		2	5346	
3	0000	Программа печати за- данного числа пробелов	3	3324	Вход в ППГЧ для однойъядерных чисел
4	0300		4	1376	
5	7450		5	3342	
6	5703		6	5372	
7	3277		7	3324	
5310	1302		5370	1377	
1	3701		1	3342	Вход в ППГЧ для двухъядерных чисел
2	1277		2	4325	
3	4676		3	5774	
4	5703		4	7212	
5	6404	Адрес	5	5500	
6	0000		6	7000	
7	0000		7	5356	Константы

Исполнительные программы диалога

5400	0013		5440	5603	
1	1230		1	1206	Десятичная запись по
2	6510		2	4602	символическому ад-
3	1054	Адреса, рабочие ячей-	3	3204	ресу
4	4230	ка, константы	4	5374	
5	5617		5	4252	
6	5612		6	3604	
7	7011		7	5603	
5410	1205		5450	6020	Вспомогательная под-
1	4607		1	5615	программа для деся-
2	3204	Восьмеричная запись	2	5437	тичной записи (пре-
3	1206	по восьмеричному ад-	3	3251	образует в одноячееч-
4	4607	ресу	4	1651	ный двоичный код 4-
5	3604		5	7106	разрядное десятичное
6	5603		6	7106	число, записанное в
7	5616		7	2251	четырёх ячейках)
5420	1206		5460	1651	
1	4602	Восьмеричная запись	1	7106	
2	3201	по символическому	2	7106	
3	1271	адресу	3	2251	
4	4607		4	1651	
5	3601		5	4650	
6	5603		6	0000	
7	5613		7	5652	
5430	1205	Десятичная запись по	5470	0000	Рабочие ячейки и
1	4607	восьмеричному адресу	1	0000	константа
2	3204		2	0000	
3	1606		3	0000	
4	3266		4	6030	
5	1227		5	0000	
6	4252		6	0000	
7	3604		7	0000	

Программа для преобразования 2 — 10 для чисел в диапазоне 0—9999 (5500—5544) и исполнительные программы блока печати

5500	0000		5520	1272	
1	3277		1	7510	
2	1677		2	5332	
3	3276		3	2275	
4	2277		4	3276	
5	1677		5	1273	
6	3277		6	3277	
7	3275		7	5310	
5510	7300	Программа для пре-	5530	6600	
1	1274	образования двухъ-	1	6626	
2	1277	ячеечных двоичных	2	7200	
3	3273	кодов, лежащих в	3	1277	
4	7004	диапазоне от 0 до	4	4730	
5	3272	9999, в десятичную	5	1275	
6	7040	форму	6	7106	
7	1276		7	7106	

5540	7106		5560	4747	Программа печати
1	1731		1	2751	группы упакован-
2	3731		2	1751	ных символов, вклю-
3	5700		3	3750	ченных в управляю-
4	7000		4	5746	щий код
5	0000	Рабочая ячейка и	5	0000	Вспомогательная про-
6	7212	константы	6	3345	грамма для программ
7	6253		7	1745	печати упакован-
5550	7207		5570	4773	ных символов
1	6252		1	2345	
2	4365	Программа печати	2	5765	
3	1745	группы упакованных	3	5303	Адрес
4	4747	символов, указан-	4	1617	Десятичная запись по
5	5746	ных адресом	5	3266	символическому ад-
6	4365		6	1205	ресу
7	1345		7	5245	

Блок приема и обработки информации с «Консула» (5600—5777)

5600	0077		5640	7650	Признак начала уста-
1	0000		1	5627	новлен?
2	7771		2	1207	
3	5744	Константы и рабочие	3	7041	Символ есть признак
4	7077	ячейки	4	1376	конца?
5	7767		5	7650	
6	0000		6	5702	
7	0031		7	1205	Поле команды за-
5610	0057		5650	7650	полнено?
1	0044		1	5356	
2	0004		2	1374	
3	0002		3	1205	
4	0003		4	3204	
5	0000		5	1207	Запись поступившего
6	0003		6	3604	символа в поле
7	0007	Поле для записи	7	7000	команды и корректи-
5620	0007	команды оператора	5660	2205	ровка счетчика сим-
1	0007		1	7000	волов
2	0004		2	5627	
3	0000		3	5404	
4	0006		4	1375	
5	0002		5	3205	Установка счетчика
6	0000		6	7240	символов и признака
7	1040		7	3206	начала
5630	7000		5670	5627	
1	5271		1	6036	
2	1207		2	0200	Прем символа
3	7041		3	3207	
4	1377	Символ есть признак	4	1207	
5	7650	начала?	5	7041	Символ есть код про-
6	5264		6	1372	бела?
7	1206		7	7640	

5700	5350		5740	7000	
1	5627		1	1373	Задержка
2	6072	Адрес подпрограммы	2	4766	
3	3206		3	7000	
4	1205		4	1604	Выход на исполнительную программу
5	1371	Длина команды = 2?	5	3204	
6	7750		6	5604	
7	5356		7	1054	Адрес в ГД
5710	7300		5750	1207	
1	1210		1	7041	
2	7006	Упаковка кода операции команды в одну ячейку	2	1365	Символ есть код возврата каретки?
3	7006		3	7650	
4	7006		4	5627	
5	1211		5	5232	
6	3203		6	1373	
7	7000		7	4766	Выдача сигнала ошибки
5720	1370		5760	4766	
1	7041		1	5747	
2	3202		2	6040	
3	1367		3	6442	
4	3010		4	6200	
5	1362		5	0020	Константы
6	3204	Поиск по таблице исполнительной программы, соответствующей принятой команде	6	6243	
7	1410		7	5777	
5730	7041		5770	0015	
1	1203		1	0016	Число команд оператора
2	7650		2	0027	
3	5341		3	0024	
4	2204		4	5627	
5	2202		5	7761	
6	5327		6	0031	
7	5356		7	0014	

Таблица кодов операций и адресов соответствующих программ

6000	6101		6020	0000	
1	6102		1	4677	
2	5062		2	3237	
3	6362		3	1620	
4	4742		4	7041	
5	4744		5	7450	
6	5742	Таблица кодов операций команд оператора	6	5233	Программа для преобразования 4-разрядного десятичного числа в двоичный код
7	5744		7	3236	
6010	6742		6030	1276	
1	6744		1	2236	
2	6342		2	5230	
3	6344		3	1237	
4	4244		4	2220	
5	7000		5	5620	
6	7000	Место для дополнительных команд	6	0000	
7	7000		7	0000	

6040	6061		6060	1054	Адрес в ГД
1	6061		1	1324	
2	6330		2	4723	Вызов сводки 1
3	6370		3	5660	
4	5410		4	1364	
5	5430	Таблица адресов исполнителей программ, соответствующих кодам операций	5	4723	Вызов сводки 2
6	7072		6	5660	
7	7077		7	5612	Адреса в поле команд оператора
6050	7162		6070	5615	Адрес подпрограммы
1	6175		1	6357	
2	5420		2	7240	
3	5441		3	3173	Связь с диспетчером
4	6131		4	5675	
5	7000	Места для дополнительных команд	5	5703	
6	7000		6	1750	Константы
7	7000		7	6661	

Таблица обозначений параметров и их адресов

6100	0000		6140	0000	
1	0000		1	0000	
2	0000		2	0000	
3	0000		3	0000	
4	0000		4	0000	
5	0000		5	0000	
6	0000		6	0000	
7	0000		7	0000	
6110	0000		6150	0000	
1	0000		1	0000	
2	0000	Таблица обозначений параметров	2	0000	
3	0000		3	0000	Таблица адресов параметров
4	0000		4	0000	
5	0000		5	0000	
6	0000		6	0000	
7	0000		7	0000	
6120	0000		6160	0000	
1	0000		1	0000	
2	0000		2	0000	
3	7210		3	7000	
4	0000		4	0000	
5	7114		5	7154	
6	0000		6	3326	
7	0000		7	1726	Резерв
6130	0000		6170	4725	
1	1267	Программа для ввода даты	1	5765	
2	4671		2	7130	
3	3327		3	7153	
4	1270		4	5773	Вход в десятичную печать параметров
5	4671		5	1374	
6	3330		6	3773	
7	5660		7	5772	

Исполнительные программы блока печати (продолжение)

6200	0037		6240	0210	
1	0037		1	4243	
2	0037		2	5615	
3	5612	Константы и рабочие ячейки	3	6307	
4	5615		4	6046	
5	0077		5	5646	Программа для вывода на печать отдельного символа
6	0013		6	1040	
7	0000		7	7200	
210	0007		6250	5643	
1	0070		1	7000	
2	0700		2	7104	
3	7000		3	7113	
4	4230		4	3252	
5	7056		5	1652	
6	3214		6	7012	
7	1214		7	7012	
6220	0213		6260	7012	
1	7106	Программа для вывода содержимого ячейки в восьмеричном виде	1	4266	Программа для печати группы упакованных символов
2	7006		2	1652	
3	4243		3	4266	
4	1214		4	2252	
5	0212		5	5255	
6	7112		6	6262	
7	7112		7	0205	
6230	7112		6270	3200	
1	4243		1	1200	
2	1214		2	7041	
3	0211		3	1205	
4	7112		4	7650	
5	7010		5	5310	
6	4243		6	1200	
7	1214		7	3201	

Исполнительные программы диалога и блока печати (продолжение)

6300	1201		6320	7041	
1	7041		1	3207	
2	1202		2	1206	
3	7650		3	4243	
4	5653		4	2207	
5	1201		5	5322	
6	4243		6	5717	
7	5666	Программа для печати группы упакованных символов	7	7000	
6310	2252		6330	4341	
1	1201		1	5732	Программа для индикации таймера
2	3206		2	1054	
3	7240		3	6627	
4	1652		4	0572	
5	4317		5	0573	
6	5264		6	0013	
7	6316		7	7107	

6340	6600		6420	7510	
1	6331		1	7041	
2	1734		2	4600	
3	4740		3	5604	
4	1733		4	1203	
5	4737		5	5222	
6	1336		6	0012	
7	4243	Программа для индикации таймера	7	5756	
6350	1735		6430	7060	
1	4740		1	7200	
2	1733		2	1351	
3	4737		3	4754	Программа для печати трех пробелов
4	5741		4	1351	
5	6661		5	4754	
6	5616		6	1351	
7	6375		7	4754	
6360	3356		6440	5630	
1	1756	Программа для преобразования 2-рядного десятичного числа, записанного в двух ячейках, в двоичный код	1	0031	
2	7106		2	6424	
3	7106		3	3241	
4	2356		4	1352	
5	1756		5	4754	Программа для печати символа с нижнего регистра
6	4755		6	1241	
7	5757		7	4754	
6370	1203		6450	1353	
1	4357		1	4754	
2	3734		2	5642	
3	1204	Программа для установки таймера	3	6505	
4	4357		4	4230	Программа для печати трех пробелов символа «=» и еще трех пробелов
5	3735		5	1350	
6	5777		6	4754	
7	6570		7	4230	
6400	6600		6460	5653	
1	5321		1	5622	
2	0040		2	6504	
3	0000		3	3261	
4	0000		4	1661	
5	3203		5	4754	Программа для печати трех символов, расположенных в трех последовательных ячейках
6	1601		6	2261	
7	0202		7	1661	
6410	7650		6470	4754	
1	5224	Программа для осуществления печати одноклеточных чисел со знаком	1	2261	
2	1203		2	1661	
3	7710		3	4754	
4	7201		4	5662	
5	1226		5	5620	
6	4754		6	7150	
7	1203		7	3275	

Исполнительные программы диалога (продолжение)

6500	1355		6510	7153	
1	4754		1	3306	
2	1275	Программа для печати ВК, трех символов, трех пробелов, «=» и трех пробелов	2	1367	
3	4262		3	7041	
4	4253		4	3366	
5	5676		5	1365	
6	5622		6	3010	
7	0000		7	1364	

6520	3363
1	1410
2	7041
3	1706
4	7650
5	5332
6	2363
7	2366

6530	5321
1	5627
2	2306
3	1706
4	7450
5	5343
6	7041
7	3361

Программа для поиска адреса параметра, указанного буквенным и двумя цифровыми символами

6540	1362
1	2361
2	5340
3	2306
4	1706
5	1763
6	5710
7	7154

6550	0013
1	0027
2	0026
3	0024
4	6243
5	0020
6	7153
7	0000

Константы в рабочие ячейки

6560	7114
1	0000
2	0012
3	6141
4	6140
5	6077
6	7777
7	0001

6570	7000
1	7000
2	3777
3	5774
4	1054
5	7000
6	7000
7	0574

Установка таймера (продолжение)

Стандартные программы перевода 2 → 10 и 10 → 2

6600	7116
1	3221
2	1223
3	3234
4	1224
5	4230
6	7106
7	4230

6630	6613
1	7100
2	3222
3	1221
4	1261
5	7430
6	3221
7	7200

6610	3226
1	1225
2	4230
3	7106
4	7006
5	7006
6	1221
7	3227

Стандартная программа для преобразования в десятичную форму одноячеечного двоичного кода

6640	1222
1	7004
2	2234
3	7430
4	5630
5	5232
6	0140
7	4060

6620	5600
1	0000
2	4002
3	1246
4	1010
5	0400
6	0000
7	0500

6650	6030
1	6340
2	7160
3	7470
4	7634
5	7660
6	7730
7	7754

Стандартная программа для преобразования 3-разрядного десятичного числа в двоичную форму

6660	7766	
1	7516	
2	3304	
3	1304	
4	0306	
5	7110	
6	3305	
7	1305	

6670	7012	
1	1305	
2	1304	
3	0307	
4	3305	
5	1305	
6	7110	
7	1305	

*Программа печати десятичного числа с плавающей запятой
(6710—6777)*

6700	7012	
1	7041	
2	1304	
3	5661	
4	0120	
5	0120	
6	7400	
7	7760	

10→2

6740	7112	
1	7112	
2	4357	
3	1226	
4	4357	
5	1227	
6	7112	
7	7112	

6710	0700	
1	6442	
2	0001	
3	6243	
4	0000	
5	0000	
6	0000	
7	0000	

6750	7112	
1	4357	
2	1227	
3	4357	
4	5717	
5	7774	
6	0000	
7	0000	

6720	3316	
1	1316	
2	0376	
3	7106	
4	7006	
5	1355	
6	3314	
7	1316	

Программа для печати десятичного числа с заданным расположением запятой

6760	3356	
1	1314	
2	7710	
3	5373	
4	1315	
5	7640	
6	5371	
7	1312	

6730	0310	
1	7112	
2	7112	
3	7112	
4	1355	
5	3315	
6	1226	
7	7112	

6770	4711	
1	1356	
2	4713	
3	2314	
4	7000	
5	2315	
6	7000	
7	5757	

Программа диалога

7000	6430	
1	6243	
2	6215	
3	0020	
4	4230	
5	7763	
6	7774	
7	5612	

Константы и рабочие ячейки

7010	5615	
1	7051	
2	3210	
3	1225	
4	3226	
5	1610	
6	7106	
7	7004	

7020	2210	Программа для упаковки в одну ячейку 4 восьмеричных цифр, расположенных в четырех ячейках	7100	3261	Константы в рабочие ячейки
1	2226		1	5232	
2	5215		2	6600	
3	1610		3	0500	
4	5611		4	3700	
5	7775		5	6253	
6	0000		6	7103	
7	5605		7	7122	
7030	7757	Программа для печати двух символов, расположенных в одной ячейке	7110	3303	Программа для десятичной печати одноячеечного кода
1	5616		1	1306	
2	1207		2	4705	
3	3231		3	5707	
4	1627		4	7062	
5	3230		5	4702	
6	1230		6	1723	
7	1206		7	4307	
7040	3230	Константы	7120	1724	Константы
1	1230		1	4307	
2	7041		2	5714	
3	1205		3	6626	
4	7540		4	6627	
5	5667		5	7000	
6	7200		6	7000	
7	1231		7	7775	
7050	4211	Программа для реализации восьмеричной или десятичной печати, заданной оператором по машинным адресам	7130	7200	Программа для восьмеричной или десятичной печати параметров
1	3204		1	1207	
2	1203		2	3231	
3	4601		3	1627	
4	1204		4	3230	
5	4602		5	1230	
6	4600		6	1327	
7	4600		7	3230	
7060	1604	Вход в восьмеричную печать	7140	1230	Вход в десятичную печать
1	4314		1	7041	
2	1231		2	1205	
3	1266		3	7540	
4	3231		4	5667	
5	5236		5	7200	
6	0004		6	1231	
7	1054		7	4777	
7070	0000	Константы	7150	7200	Константы
1	4602		1	1231	
2	1271		2	4776	
3	3261		3	5773	
4	5232		4	7200	
5	7776		5	1231	
6	4314		6	1375	
7	1276		7	3231	

7160	5335	
1	0221	
2	1374	
3	3353	Вход в восьмеричную
4	5330	печать
5	3361	
6	1761	Вставка
7	4602	

7170	5354	
1	6403	
2	7000	
3	6166	
4	5365	Константы
5	0003	
6	6510	
7	6476	

Программа управления печатью кодограмм (7200—7324)

7200	7776	
1	7700	
2	0000	
3	7274	Константы и рабочие
4	7242	ячейки
5	0000	
6	0031	
7	0000	

7250	3100	
1	3200	
2	4100	
3	4200	
4	4300	
5	7700	
6	0500	
7	0600	

Таблица кодов операций управляющих кодов

7210	0000	Адрес возврата
1	3207	
2	1206	
3	7041	
4	3205	
5	1204	Установка начальных
6	3010	условий
7	1203	

7260	4400	
1	0700	
2	0000	
3	3600	
4	4500	
5	4600	
6	4700	
7	0100	

7220	3202	
1	1607	
2	0201	
3	7041	
4	1410	
5	7650	
6	5234	
7	2202	Поиск исполнитель-
		ной программы

7270	0200	
1	0300	
2	0400	
3	1700	
4	7354	
5	7354	
6	7354	
7	7354	

7230	2205	
1	5221	
2	5610	
3	7000	
4	2207	
5	2207	
6	1602	
7	3202	

7300	7354	
1	7354	
2	7354	
3	7354	
4	7354	
5	7354	
6	7354	
7	5210	

Таблица адресов исполнительных программ

7240	1207	
1	1200	Выход на исполня-
2	5602	тельную программу
3	1000	
4	2000	
5	3000	
6	4000	
7	2100	

310	5226	
1	7357	
2	5235	
3	7357	
4	7357	
5	7357	
6	7357	
7	7357	

7320	5363		7350	3207	
1	5367		1	5733	
2	7364		2	7000	Резерв
3	5552		3	7330	
4	5556		4	4333	Вход в печать одно-
5	0000		5	1353	ячеечного числа
6	7300		6	5775	
7	0000		7	4333	
7330	0000	Константа и рабочие	7360	1353	Вход в печать двухъ-
1	0000	ячейки	1	5762	ячеечного числа
2	0000		2	5367	
3	0000		3	0000	
4	3327		4	3363	
5	1727		5	7240	
6	0326		6	1207	
7	7001		7	3207	
7340	3332		7370	1763	Программа для вы-
1	1727		1	4773	вода на печать от-
2	3330	Программа для печат-	2	5212	дельного символа
3	2327	ти отдельного одно-	3	6213	
4	1727	или двухъячеечного	4	5610	
5	3331	числа	5	5363	
6	7240		6	7000	
7	1207		7	7000	

Приложение 3а

ОПЕРАЦИОННАЯ СИСТЕМА ДЛЯ «ЭЛЕКТРОНИКИ-100» С РАСШИРЕННЫМ ОБЪЁМОМ МОЗУ

Настоящая операционная система для ЭВМ «Электроника-100» с блоком МОЗУ от одного до восьми кубов (по 4096 слов), таймером и набором периферийных устройств обеспечивает мультипрограммную работу ЭВМ в реальном времени, ввод — вывод информации через электрическую пишущую машинку (ЭПМ) «Консул-254» параллельно с основными вычислениями, работу с периферийными устройствами, организующими запросы в ЭВМ по каналу прерывания, а также предоставляет пользователям эффективный способ кодирования печатных сообщений и бланков. По сравнению с малой операционной системой здесь реализованы более широкие возможности и несколько изменены команды обращения к различным блокам диспетчера (что связано в основном со спецификой работы с расширенной памятью) и к интерпретатору программного обеспечения ввода — вывода.

ПРОГРАММА-ДИСПЕТЧЕР

Блок установления очередности развит до уровня блока внутреннего прерывания, имеющего вход со стороны программ основного функционального обеспечения и позволяющего им, таким образом, предъявлять диспетчеру для обслуживания новые задачи по мере необходимости, возникающей при решении любой задачи.

В службе времени добавлен блок заказанного времени, что обеспечивает возможность продолжения задачи в запрашиваемое ею время с точностью до часов и минут.

В ряде случаев при переполнении очереди диспетчера программа основного функционального обеспечения получает об этом информацию, что позволяет программным путем учитывать и исправлять некоторые ситуации (например, с помощью повторения запроса через некоторое время). Ввод обслуживается как обычная задача с высоким приоритетом.

Обслуживание новых задач начинается при $|HP|=0$ и $|DP|=0$.

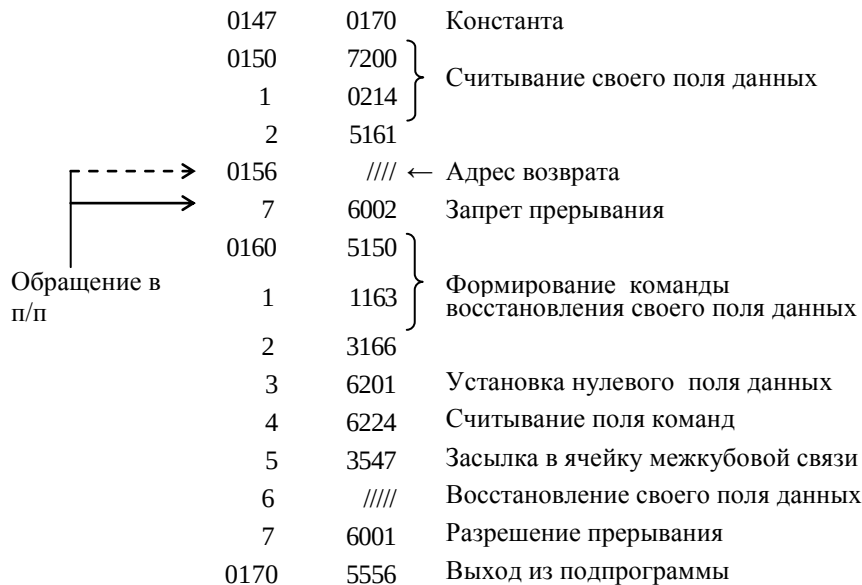
В качестве дополнительного блока в состав диспетчера введен блок «узкого места» (см. ниже). В остальном функции и возможности диспетчера настоящей операционной системы не отличаются от таковых у диспетчера малой операционной системы.

ОБРАЩЕНИЯ К ПРОГРАММЕ-ДИСПЕТЧЕРУ

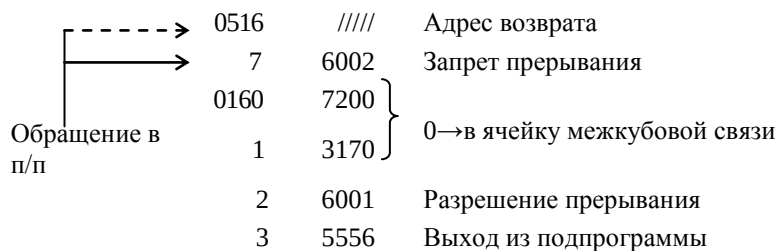
Дополнительными особенностями здесь являются необходимость введения в качестве дополнительной характеристики каждой задачи её поля данных и поля команд и передача диспетчеру информации о поле команд, откуда к нему произошло обращение, в специальную ячейку межкубковой связи. Приоритет каждой задачи, её поле данных и поле команд упаковываются в одно машинное слово «приоритет + поля», имеющее следующий формат:

0	1	2	3	4	5	6	7	8	9	10	11
0	Приоритет					Поле			Поле		

Перед обращением в службы диспетчера обязательна засылка номера своего поля команд в ячейку 0170 0-го куба (во вторую триаду справа — как при считывании поля команд). Целесообразно во всех кубах (кроме 0-го) иметь соответствующую подпрограмму, занимающую с учётом возможных расширений диспетчера ячейки 0147 — 0152 и 0156 — 0170:

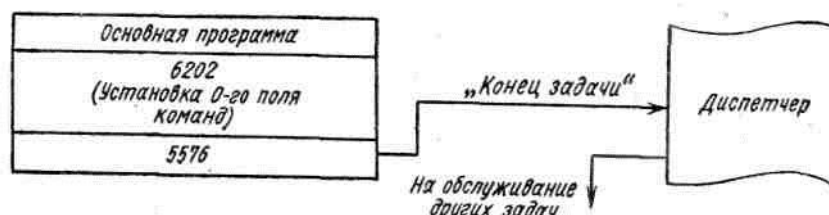


Соответствующая подпрограмма в нулевом кубе МОЗУ, вводимая исключительно с целью унификации команд обращения к диспетчеру, имеет вид:



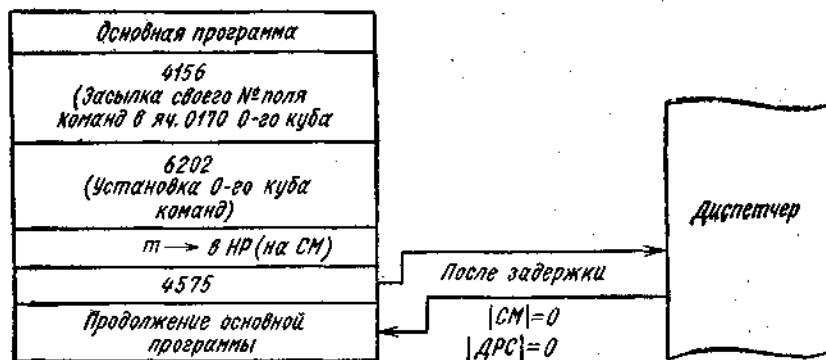
Адреса для обращения в различные блоки диспетчера вынесены на нулевую страницу каждого куба МОЗУ (дублируется содержание ячеек 0154, 0155, 0171 — 0176 нулевого куба МОЗУ). При этом собственно команды обращения к диспетчеру из любого места основных функциональных программ приобретают следующий вид:

1. Окончание задач.



2. Обращение к службе времени:

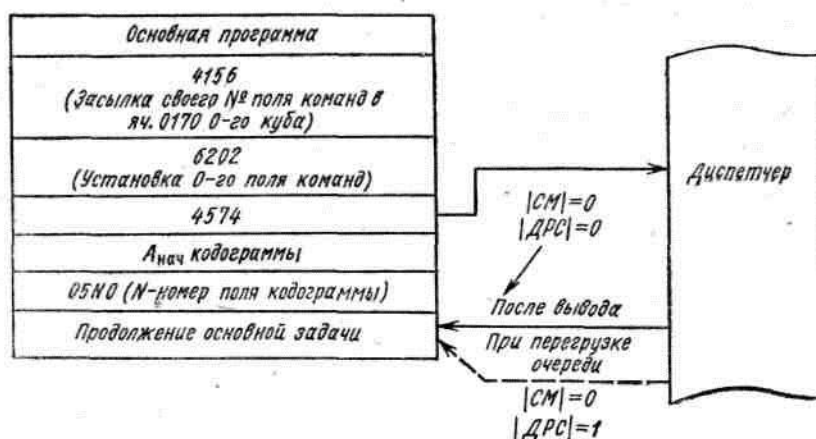
а) задержка на m_8 интервалов (в исходном состоянии программа диспетчера настроена на интервал в 0,48 с)



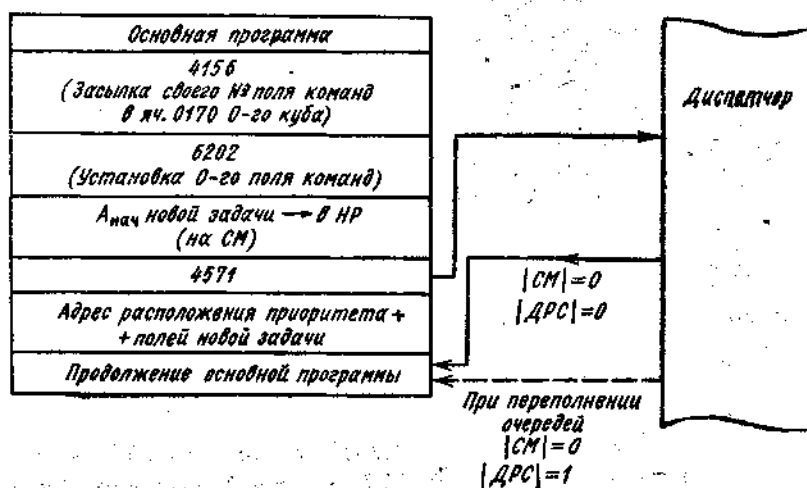
б) запрос на продолжение задачи в точно назначенное время: обращение такое же, как в п. а), но вместо m_8 на СМ вызывается код в виде

0	1	2	3	4	5	6	7	8	9	10	11
Код часов						Код минут					

и затем вместо команды 4575 выполняется команда 4573. 3. Обращение к службе вывода:



4. Вызов на обслуживание другой задачи:



Если приоритет новой задачи не выше, чем у исходной (находящейся на обслуживании), то заявка попадает в общую очередь и управление возвращается обслуживавшейся ранее задаче, в ином случае новая задача начинает обслуживаться тотчас же, а исходная снимается с обслуживания (и попадает в очередь прерванных задач).

Настройки диспетчера

Предусмотрены следующие настройки, важные при реализации конкретных систем:

- настройка службы времени;
- настройка блока обслуживания прерываний;
- вызов ячейки на индикацию при холостом ходе.

1. Настройка службы времени практически ничем не отличается от таковой для малой операционной системы, лишь константа исходной редакции здесь обеспечивает $T_s = 0,48$ с.

2. Настройка блока обслуживания прерываний состоит в уточнении номеров подключенных внешних устройств, присылающих свои запросы по каналу прерывания, и заполнении таблицы характеристик запрашиваемых задач. В исходной редакции программа-диспетчер после прерывания опрашивает флаги таймера, устройства ввода, внешних устройств о номерами 14₈ и 15₈, входного коммутатора дискретных сигналов и устройств вывода. Таким образом, кроме таймера и устройств ввода — вывода, по селекторным каналам могут присылать свои запросы ещё два внешних устройства и 12 устройств имеют вход в машину через групповой флаг входного коммутатора дискретных сигналов. Номера соответствующих селекторов, предусмотренных настоящим вариантом операционной системы, и адреса констант, характеризующие задачи, которые вызываются по запросам этих устройств, сведены в таблицу.

Номер внешнего устройства	Адрес расположения соответствующих		Номер внешнего устройства	Адрес расположения соответствующих	
	начальный адрес	приоритет		адрес начальный	приоритет + поля
03 ₈ (ввод)	2542	2661	2	2547	2566
14 ₈	2543	2562	3	2550	2567
15 ₈	2544	2563	4	2551	2570
31 ₈ (входной коммутатор дискретных сигналов); разряды выделенного канала:	2545	2564	5	2552	2571
			6	2553	2672
			7	2554	2573
			8	2555	2574
			9	2556	2575
			10	2557	2576
			11	2560	2577

Когда подключены не все из названных селекторов, соответствующую команду типа «*6nm3*» (где *nm* — восьмеричный номер внешнего устройства) в ячейках 0207—243 следует заменить на «7000». Если подключены внешние устройства с другими номерами селекторов, следует заменить одну из команд типа «*6nm3*» на аналогичную, но с номером подключенного селектора. Начальный адрес и приоритет задачи, инициируемой этим внешним устройством, помещают в ячейки, соответствующие (по таблице) внешнему устройству, номер которого изменяется.

Код номера канала, выделенного для ввода слова флагов, запасается (располагается) в ячейке 0372 — в исходной редакции 0000.

Команды считывания слова флагов через входной коммутатор дискретных сигналов располагаются в ячейках 0274—0277.

3. Данные о ячейке, содержимое которой вызывается на СМ при холостом ходе, заносятся в следующие ячейки 0-го куба МОЗУ:

1612 — номер куба (в разряды 6 — 8);

1613 — абсолютный адрес ячейки внутри куба.

В исходной редакции ячейки 1612 и 1613 содержат нули, индицируя, таким образом, адрес последнего места прерывания.

Расширение диспетчерских функций при появлении «узкого места»

В тех случаях, когда две задачи (или более двух) с различными приоритетами используют некоторые общие подпрограммы или внешние устройства (помимо ЭПМ), возникает необходимость дополнять операционную систему блоком «узкого места» или местным диспетчером (МД). В настоящем варианте операционной системы данный блок размещается на дополнительно отводимом месте в 1-м кубе МОЗУ.

Для функциональных задач системы при работе с МД устанавливается следующая дисциплина обслуживания.

Блок МД имеет два фиксированных входа (адреса).

Если задаче требуется использовать подпрограмму или внешнее устройство, по условию снабжаемое блоком МД, то задача должна обратиться сначала по первому входу в МД, как за разрешением. Если соответствующая подпрограмма или устройство свободны, то программа, обратившейся в МД, немедленно возвращается управление; в ином случае МД «консервирует» задачу в своей очереди до момента, пока устройство не освободится, и лишь после этого возвращает управление обратившейся к нему задаче. В любом случае для задачи, обратившейся к МД по первому входу, разрешением для занятия внешнего устройства является возвращение управления.

По окончании использования внешнего устройства или подпрограммы функциональная задача должна немедленно освободить МД, обратившись к нему по второму входу.

При обращении к МД по первому входу диспетчер производит анализ состояния очереди к рассматриваемому внешнему устройству (очереди № 3). Задача, уже захватившая МД (стоящая на первом месте в очереди № 3), всегда обслуживается до конца. Далее расстановка задач определяется соотношением приоритетов новой заявки и задачи, стоящей на втором месте в очереди № 3: если приоритет заявки не выше, то заявка размещается в конце очереди № 3, в ином случае вся очередь сдвигается назад на одну позицию, освобождая второе место для заявки.

Если хотя бы одно место в очереди № 3 занято, задача, обратившаяся к МД по первому входу, с обслуживания у процессора снимается. В очереди № 3 для нее сохраняются следующие данные: адрес продолжения, приоритет + поля и содержимое СМ в момент обращения к МД.

Если очередь № 3 была пуста, то заявка немедленно принимается к обслуживанию. Содержимое СМ при этом переписывается в ячейку 0153 1-го куба МОЗУ, и задача возвращается управление, как после обращения к подпрограмме.

При обращении к МД по второму входу диспетчер сдвигает всю очередь № 3 вперёд на одну позицию, переписывает хранимое в очереди № 3 содержимое СМ для задачи, попавшей на первое место, в ячейку 0153 1-го куба МОЗУ и обращается к блоку

внутреннего прерывания для разрешения вопроса: кому предоставить право на занятие процессора — задаче, освободившей МД, или задаче, которая теперь могла бы занимать МД, если ей будет передано управление (как обычно, блок внутреннего прерывания решает этот вопрос на приоритетной основе).

Для размещения программ МД используются ячейки 3400 — 3777 в 1-м кубе МОЗУ. Кроме того, ячейки 0154, 0155 в каждом кубе МОЗУ занимают под адреса обращения к МД и ячейка 0153 в первом кубе МОЗУ — для занесения в нее содержимого СМ в момент обращения к МД по первому входу.

Блок МД не является автономным и может использоваться только вместе с общей программой настоящей операционной системы.

В настоящем варианте операционной системы очередь № 3 содержит 7 мест.

Обращение к МД осуществляется с помощью следующих цепочек команд.

Обращение по первому входу:

- обращение к подпрограмме засылки своего поля команд в ячейку межкубовой связи по команде «4156»;
- вызов на СМ кода, который требуется запасти в ячейке 0153 1-го куба МОЗУ;
- установка первого поля команд — по команде «6212»;
- собственно обращение к МД по команде «4554».

Обращение по второму входу:

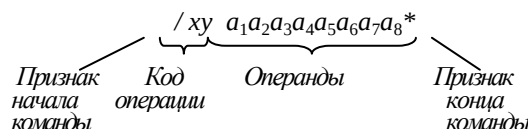
- обращение к подпрограмме засылки своего поля команд в ячейку межкубовой связи по команде «4156»;
- установка первого поля команд — по команде «6212»;
- собственно обращение к МД по команде «4555».

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВВОДА - ВЫВОДА

Ниже приводятся сведения, отличающиеся от тех, которые используются в аналогичных программах малой операционной системы, либо дополняющие их.

Программное обеспечение диалога

Общая структура команд оператора имеет вид:



Приняв команду оператора, ЭВМ вначале организует печать самой команды в виде, «понятном» машине, с добавлением информации о номере куба, с которым в настоящий момент происходит работа, а затем производится автоматическое исполнение команды.

Если машина не может опознать команду в результате её неправильного набора или сбоев при вводе, печатается слово ОШИБКА.

В настоящем варианте операционной системы реализовано 9 команд (и предусмотрена возможность дополнительного введения ещё 6 команд, см. ниже):

- /УВ $a_1a_2a_3a_4a_5a_6a_7a_8^*$ — установка времени: a_1a_2 — часы, a_3a_4 — минуты, a_5a_6 — день, a_7a_8 — месяц;
- /ИВ* — индикация времени;
- /УК a^* — установка куба a (дальнейшие команды печати и записи будут относиться к кубу a);
- /ПВ $a_1a_2a_3a_4b_1b_2^*$ — печать восьмеричная $(b_1, b_2 + 1)_8$ ячеек, начиная с адреса $a_1a_2a_3a_4$;
- /ПД $a_1a_2a_3a_4b^*$ — печать десятичная $(b + 1)_{10}$ ячеек, начиная с адреса $a_1a_2a_3a_4$;
- /ЗВ $a_1a_2a_3a_4b_1b_2b_3b_4^*$ — запоминание восьмеричных кодов $a_1a_2a_3a_4$ и $b_1b_2b_3b_4$ в промежуточных ячейках для записываемого кода (ЗК) и адреса записи (АЗ) соответственно (команда предварительная);
- /ЗД $a_1a_2a_3a_4, b_1b_2b_3b_4^*$ — запоминание с предварительным переводом $10 \rightarrow 2$ чисел $a_1a_2a_3a_4$ и $b_1b_2b_3b_4$ в промежуточных ячейках для записываемого кода (ЗК) и адреса записи (АЗ) соответственно (команда предварительная);
- / * — запись запятого ранее кода (ЗК) в ячейку по запятому адресу (АЗ) — команда исполнительная;
- /СВ i^* — вызов на печать сводки i , где i — десятичная цифра ($0 \leq i \leq 9$).

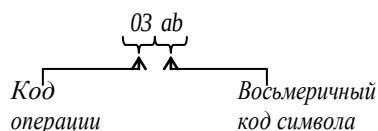
ПРОГРАММА ПЕЧАТИ БЛАНКОВ (ППБ)

ППБ обеспечивает печать информационных бланков сообщений. Каждому бланку и сообщению соответствует определенный упорядоченный набор кодов (кодограмма), который однозначно определяет подлежащий печати текст.

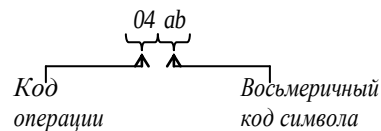
В состав ППБ входит программа управления печатью (УП) и исполнительные программы, реализующие различные виды печати: программы печати дробно-десятичных одно- и двухъязычных чисел, программы печати группы упакованных символов, программа вывода отдельного символа, программа печати даты и времени, программа перехода на другой сегмент кодограммы (расположенный в ином участке памяти) и программа, осуществляющая включение другой кодограммы в данную.

В интерпретаторе кодограмм настоящей операционной системы реализовано 16_{10} операций, имеющих следующий вид.

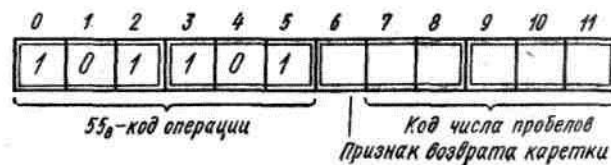
1. Печать символа в верхнего регистра



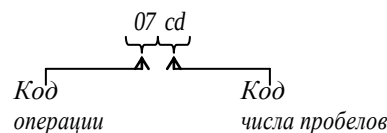
2. Печать символа с нижнего регистра



3. Печать пробелов с указанием необходимости предварительного возврата каретки



4. Печать времени



По этой команде осуществляется печать $(cd)_8$ пробелов и затем печать текущего времени и даты в виде

$$a_1a_2 — a_3a_4/a_5a_6 — a_7a_8,$$

где a_1a_2 — часы; a_3a_4 — минуты; a_5a_6 — день месяца; a_7a_8 — месяц.

После печати дополнительно делается один пробел.

Примечание. Эта команда позволяет программным путем осуществлять при распечатке бланков те же функции, что и директива оператора /ИВ.

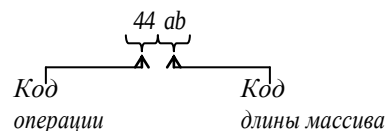
5. Установка поля данных



Здесь N — код требуемого поля данных (куба МОЗУ), к которому будут относиться дальнейшие команды печати чисел и неупакованных символов.

6. Печать группы символов, коды которых расположены в последовательных ячейках памяти (в правых половинах ячеек). Команда двусловная.

Первое слово имеет вид



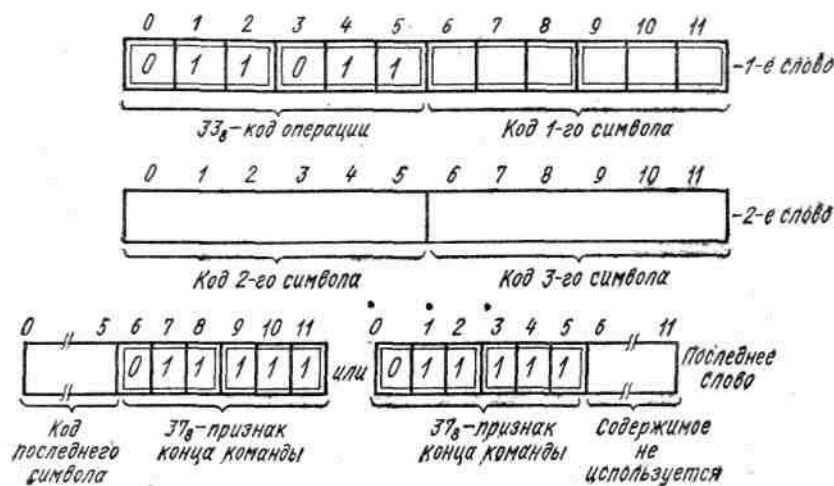
Во втором слове указывается начальный адрес массива ячеек (в пределах куба МОЗУ, установленного ранее при исполнении команды п. 5).

Если $(ab)_8 = n$, то печатается n символов.

7. Печать упакованной группы символов, расположенной непосредственно за командой.

Команда, включающая вместе с выводимой информацией произвольное число машинных слов. Коды символов упаковываются по два в одну ячейку. Коды 37_8 и 77_8 , не используемые в качестве информационных, являются служебными. Код 37_8 , встреченный в левой или правой половине служебного слова, служит признаком конца команды (и группы символов).

Команда, таким образом, имеет вид:



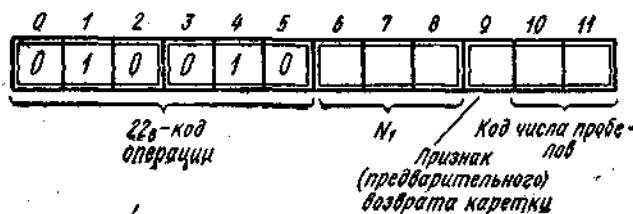
Код 77_8 применяется как признак организации многократного (до 63_{10}) ввода одного и того же символа. Для этого используется три полуслова (в произвольном месте рассматриваемой команды).



Здесь $^{120} 77_8$ — признак; $(ab)_8$ — код необходимого числа повторений печати символа с кодом $(cd)_8$.

8. Печать упакованной группы символов, расположенных по указываемому адресу. Команда двусловная.

Первое слово имеет вид

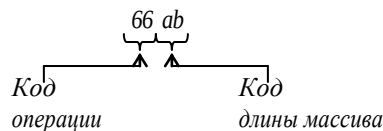


Здесь N_1 указывает (на время действия данной команды) номер куба, где располагается упакованная группа символов; её начальный адрес указывается во втором слове команды.

Как и в п. 7, выполнение команды заканчивается при встрече с кодом 37_8 .

9. Восьмеричная печать группы кодов из последовательно расположенных ячеек. Команда двусловная.

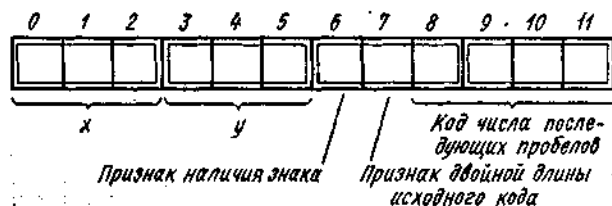
Первое слово имеет вид



Во втором слове располагается начальный адрес выводимого массива.

Если $(ab)_8 = n$, то осуществляется вывод $n + 1$ кодов.

10. Печать дробно-десятичного числа. Команда двусловная. Первое слово имеет вид



Во второй ячейке указывается адрес исходного кода (куб МОЗУ определяется последним полем данных, установленным по

¹²⁰ Пример приведен для случая, когда первое полуслово расположено в старших разрядах ячейки

команде п. 6).

По этой команде код по указанному адресу переводится из двоичной формы в десятичную и на печать выдается результат в виде x цифр, из которых y цифр справа отделены запятой. Исходный код может занимать одну или две ячейки (в последнем случае в признак двойной длины исходного кода заносится 1). Если исходный код должен рассматриваться как положительное число, то признак наличия знака следует устанавливать равным 0; если признак наличия знака в состоянии «1», то полагается, что кодировка чисел соответствует общепринятой для «Электроники-100» и число печатается со знаком (при условии, что печать знака не заблокирована командой «установка поля данных»),

11. Печать последовательности дробно-десятичных чисел. Команда трехсловная.

Первое слово имеет вид



Второе слово определяет формат выводимых чисел в соответствии с п. 10.

Третье слово указывает начальный адрес массива исходных кодов, занимающих последовательные ячейки памяти.

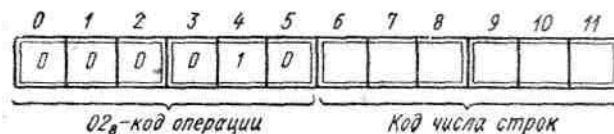
12. Печать группы чисел из произвольных мест памяти.

Если требуется вывести n чисел, то команда занимает $n + 2$ слова.

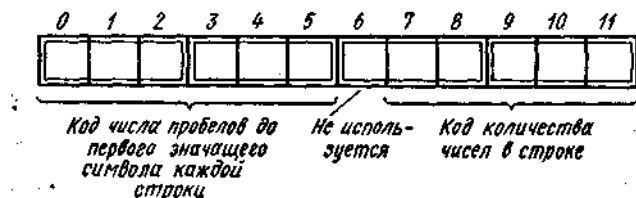
Код операции (указываемый в первой половине первого слова) 77₈. В остальном отличие от п. 11 состоит в том, что, начиная с 3-го слова рассматриваемой команды, указывается адрес каждого исходного кода (всего n адресов).

13. Печать нескольких строчек чисел. Команда содержит 4 машинных слова и является дальнейшим развитием команд п. п. 10 и 11.

Первое слово имеет вид



Второе слово имеет вид



Два последних слова команды по своему назначению и структуре полностью идентичны двум последним словам команды п. 11.

14. Переход к новому сегменту кодограммы. Команда двусловная.

Первое слово имеет вид



Здесь N — номер куба, из которого будут выбираться дальнейшие команды печати.

Второе слово указывает начальный адрес нового сегмента кодограммы.

15. Переход к новому сегменту кодограммы с возвратом. Команда двусловная.

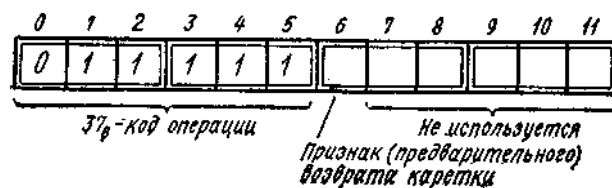
Первое слово имеет вид



Здесь N — номер куба, из которого будут выбираться дальнейшие команды печати.

Второе слово содержит начальный адрес нового сегмента кодограммы. При выборке из нового сегмента команды «Конец кодограммы» окончания печати не произойдет, а будут обрабатываться команды печати, располагающиеся вслед за командой перехода с возвратом.

16. Конец кодограммы.



По этой команде завершается выполнение кодограммы.

Пример. Предположим, что необходимо обеспечить печать бланка следующего вида:

Показатели

XXX, ХППХХ, ХППХХХ, ХППП+ХХХППППХ, ХХПХ, ХХ

$Q = \pm XX$, ХХПППА = ХХХХХППППППППП ч-мин/день-мес.

Здесь П обозначает пробел; Х представляет произвольную десятичную цифру, а прямоугольник обозначает некоторый текст, кодограмма которого уже записана и имеет начальный адрес 3500 (куб 1). Допустим, что первые три числа представлены в памяти двухъядерными кодами, занимающими в 0-м кубе ячейки 4010—4011, 4012—4013 и 4014—4015 соответственно; четвертое число формируется из одноядерного кода, расположенного по адресу 7501 в 1-м кубе; пятому и шестому числам соответствуют одноядерные коды с адресами 7603 и 7604 (0-й куб); седьмое число определяется по одноядерному коду, записываемому в ячейку 3567 (1-й куб), а восьмое — по двухъядерному коду с адресом 5020 (1-й куб).

Такой бланк может быть описан следующей кодограммой:

Адрес	Содержимое	Пояснения
00		
34	5557	Возврат каретки в печать 15 пробелов
1	3357	} Печать слова «Показатели» в пропуск двух строк
2	5652	
3	4047	
4	4062	
5	4553	
6	5020	
7	2037	
3410	1100	Установка 0-го куба
1	0103	} Печать первых трех чисел
2	4123	
3	4010	} Печать пробела и установка 1-го куба
4	1111	
5	3044	} Печать четвертого числа
6	7501	
7	1100	Установка 1-го куба
3420	0102	} Печать пятого и шестого чисел
1	3202	
2	7603	} Возврат каретки
3	5540	
4	0610	} Печать кодограммы, определяющей содержимое прямо- угольника
5	3500	
6	3320	} Возврат каретки и печать «Q—»
7	2676	
3430	1337	} Установка 1-го куба
1	1110	
2	4243	} Печать седьмого числа
3	3567	
4	0340	Печать символа «A»
5	0413	Печать символа «=»

ДИАГНОСТИКА СБОЕВ И ОТКАЗОВ, ЗАПУСК, ОГРАНИЧЕНИЯ

При сбоях и отказах происходит следующая печать (в экстренном порядке, но после завершения уже начавшейся печати):

КТО ТАМ! — неизвестное прерывание.

ЧЕЙ ФЛАГ — после прерывания от входного коммутатора дискретных сигналов «слово» флагов не содержит ни одной 1.

СБОЙ 1 — при ненулевом «слове» флагов не обнаружено ни одной 1 при сдвигах.

ОТКАЗ 1 — $N\ abcd$ — отказ новой заявке от одного из блоков диспетчера; N — номер куба, $abcd$ — начальный адрес (или адрес продолжения) задачи, получившей отказ (и потерянной).

ОТКАЗ 1* — $N\ abcd$ — тоже самое, но по заявке от решаемой задачи.

ОТКАЗ 2 — $N\ abcd$ — отказ задержке в результате переполнения очередей; N и $abcd$ — номер куба и адрес задачи, получившей отказ (и потерянной).

ОТКАЗ 2* — $N\ abcd$ — то же самое, но в результате сбоя внутри цикла формирования адресов.

ОТКАЗ 3 — $N\ abcd$ — отказ выводу в результате переполнения очередей; N и $abcd$ — номер куба и адрес задачи, получившей отказ (управление возвращается задаче—заказчику вывода, но вывод игнорируется).

ОТКАЗ 4 — $N\ abcd$ — отказ в приеме заявки на обслуживание МД в результате переполнения очереди № 3; N и $abcd$ — номер куба и адрес задачи, получившей отказ (заявка теряется).

Стартовый адрес программы-диспетчера—0400. Примерно через 1с после первоначального пуска (и в 00 ч 00 мин) машина печатает: «Установите дату и время» — соответствующие операции могут быть произведены по командам оператора.

Перед обращением к службе времени или к службе вывода диспетчера программы основного функционального обеспечения должны сами запоминать содержимое регистров и индексных ячеек, необходимое для продолжения нормальной работы после возвращения им управления от

соответствующей службы диспетчера.

Число мест в очередях диспетчера:

— в очереди № 1 — 10_{10} ;

— в очереди № 2 — 5;

— в очереди задержек — 10_{10} ;

— в службе заказанного времени—3;

— в очереди вывода — 15_{10} ;

— число внешних устройств (помимо таймера и ЭПМ), которые могут обращаться к диспетчеру по каналу прерывания через индивидуальные селекторные каналы — 2; через специальный канал входного коммутатора дискретных сигналов — 12_{10} .

Программа настоящей операционной системы занимает в 0-м кубе МОЗУ $4507_8 = 2311_{10}$ ячеек с адресами 0000, 0001, 0156 — 0163, 0165 — 4663; ячейки 0010, ООП, 0040, 0041 используются при выводе.

Кроме того, во всех других кубах МОЗУ занимают ячейки 0147 — 0152, 0156 — 0176.

При использовании блока «узкого места» (местного диспетчера) дополнительно занимают в 1-м кубе МОЗУ ячейки 0153 — 0155 и 3400 — 3777, а во всех остальных кубах—ячейки 0154, 0155.

ТЕКСТ ПРОГРАММЫ

Адрес	Содержимое	Пояснения
0000	1600	— Адрес возврата при прерываниях
1	5377	БП в начало блока обслуживания прерываний
0153	0000	Содержимое СМ
4	3400	Адрес первого обращения
5	3613	Адрес второго обращения
6	0000	} при работе в МД
7	6002	
0160	7200	} Подпрограмма заполнения ячейки межкубовой связи при работе в 0-м кубе МОЗУ
1	3170	
2	6001	
3	5556	
4	0000	Резерв
5	0000	Рабочая ячейка блока вывода
6	0000	Поле данных
7	0000	Адрес
0170	0000	Рабочая ячейка межкубовой связи
1	1000	Адрес блока внутреннего прерывания
2	0000	Резерв для расширения диспетчера
3	0674	Адрес службы заказанного времени
4	1614	Адрес службы вывода
5	0600	Адрес службы задержек
6	1420	Адрес блока окончания задач
7	7000	Константа и незначащая команда

Блок обслуживания прерываний

(с адреса 0246 — Блок выхода из прерываний)

0200	3376	Запоминание со-	0230	5312	
1	7004	держимого СМ и	1	7000	
2	3375	ДРС	2	6313	Запрос от ВКДС?
3	1170	Запоминание содер-	3	5241	
4	3361	жимого ячейки 0170	4	5272	Подготовка обра-
5	6234	Запоминание содер-	5	3170	
6	3342	жимого РЗП	6	1766	
7	6133		7	3366	
0210	5212	Запрос от таймера?	0240	5316	
1	5762		1	6041	
2	3377		2	5245	Запрос от вывода?
3	6031		3	6042	
4	5217	Запрос от ввода?	4	5774	
5	6032		5	4763	В блок экстренного вывода
6	5312		6	1342	
7	2377		7	0341	Подготовка команды
0220	6143	Запрос от ВУ № 14?	0250	1340	восстановления по-
1	5224		1	3264	ля команд
2	7000		2	1342	
3	5312		3	7106	Подготовка команды
4	2377	Запрос от ВУ № 15?	4	7004	
5	6153		5	0341	
6	5231		6	1337	
7	7000		7	3263	восстановления по-

0260	1361	Восстановление ячейки межкубовой связи	7	5246	Выход на обслуживание первой задачи
1	3170	0 → признак «адрес... заносился»	0330	7201	
2	3743		1	4763	
3	0000	Восстановление полей	2	5246	Обращения в блок экстренного вывода
4	0000		3	7201	
5	1375	Восстановление содержимого СМ и ДРС	4	7001	
6	7110		5	4763	
7	1376		6	5246	
			7	6201	Константа
0270	6001	Разрешение прерывания и выхода на продолжение прерывной задачи	0340	6202	Константы
1	5400		1	0070	
2	1365	Подготовка цикла	2	0000	Рабочая ячейка
3	3364		3	1162	Адрес признака
4	1372		4	2024	
5	6314	Считывание слова флагов	5	6745	
6	7200		6	5127	
7	6312		7	6453	Кодограмма текста «Чей флаг»
0300	7450	Флаги ВУ есть?	0350	4043	
1	5330		1	2037	
2	2377	Счет пройденных позиций и сдвиг слова флагов	2	2024	
3	7104		3	5262	
4	2364	Сдвиги все?	4	5627	
5	5307		5	6240	Кодограмма текста «Кто там!»
6	5333		6	5426	
7	7420	Этот флаг? ($ ДРС = 1$)?	7	7224	
0310	5302	Запоминание остатка слова	0360	2037	
1	3371		1	0000	
2	1377		2	0400	
3	1367		3	2041	
4	3366	Формирование адреса в таблицах	4	0000	
5	5235		5	7763	
6	1370		6	0000	
7	1377		7	2561	Адреса, константы в рабочих ячейках
0320	3373	Вызов адреса новой задачи	0370	2542	
1	1773		1	0000	
2	4571	Обращение в блок внутреннего прерыв.	2	0000	
3	0366		3	0000	
4	1371		4	3412	
5	7440	Остались флаги?	5	0000	
6	5302		6	0000	
			7	0000	

Служба времени

0400	6771	Запуск таймера на 0,48с	0410	5233	
1	1377		1	3374	Установка минут и контроль: «час истек?»
2	6137		2	2373	
3	7200		3	1373	
4	2374	Минута истекла?	4	1370	
5	1374		5	7640	
6	1371		6	5233	
7	7640		7	3373	

0420	2372		0510	1754	
1	1372		1	7640	
2	1376	Установка часов и	2	5316	
3	7640	контроль: «сутки	3	2354	
4	5233	истекли?»	4	2353	
5	3372	Иницирование об-	5	5310	
6	3170	ращения к оператору:	6	1372	
7	1375	«Установите да-	7	7106	
0430	4571	ту и время»	0520	7006	
1	0362		1	7006	Цикл сравнения теку-
2	7200		2	1373	щего времени с за-
3	1357	Есть задача на об-	3	1754	данным и предъявле-
4	7450	служивание задер-	4	7640	ния соответствующих
5	5277	жек?	5	5346	задач блоку внутрен-
6	7041		6	3754	него прерывания
7	3355		7	1360	
0440	1365		0530	1353	
1	3354	Организация цикла	1	3352	
2	3353		2	1752	
3	1754		3	3352	
4	7640		4	3170	
5	5251		5	1361	
6	2354		6	1353	
7	2353		7	3366	
0450	5243		0540	1766	
1	2754		1	4571	
2	5274		2	0552	
3	1363		3	7240	
4	1353		4	1356	
5	3352		5	3356	
6	1752		6	2355	
7	3352		7	5313	
0460	3170	Цикл отсчета обслу-	0550	5767	БП в блок выхода
1	1364	живаемых задержек	1	7000	из прерываний
2	1353	и предъявления за-	2	0000	
3	3366	дач, чья задержка	3	0000	
4	1766	истекла, блоку внут-	4	0000	
5	4571	реннего прерывания	5	0000	
6	0552		6	0000	
7	7000		7	0000	
0470	7000		0560	2354	
1	7240		1	2351	
2	1357		2	2346	
3	3357		3	2403	
4	2355		4	2371	Рабочие ячейки, ад-
5	5246		5	2357	реса и константы
6	7000		6	0000	
7	1356		7	0246	
0500	7450	Есть задачи с занесенным	0570	7704	
1	5767	временем начала?	1	7603	
2	7041		2	0027	
3	3355		3	0073	
4	1362	Организация цикла	4	0172	
5	3354		5	2300	
6	3353		6	7750	
7	7000		7	7763	

Блок приема временных заявок

0600	0000	← Адрес возврата	0660	6214	
1	6002	Запрет прерыв.	1	7112	
2	4255	На формир. полей	2	7010	
3	7200		3	1170	
4	1777		4	3374	Подпрограмма формир.
5	1373		5	6201	мирования полей
6	7710		6	1760	
7	5214		7	0357	
		Анализ наличия мест			
0610	1372		0670	1374	
1	4765		1	3374	
2	7000		2	7000	
3	5576		3	5655	
4	2777	+1 к числу занятых мест	4	0000	← Адрес возврата
5	1200		5	6002	Запрет прерываний
6	3376		6	4255	На формир. полей
7	1371		7	3254	
0620	3366		0700	1724	
1	1777		1	1325	
2	7041		2	7710	Анализ наличия свободного места
3	3364	Подготовка цикла	3	5307	
4	1370		4	1306	
5	3362		5	5211	
6	1367		6	0010	
7	3361		7	2724	+1 к числу занятых мест
0630	7240		0710	1274	
1	3254		1	3376	
2	1766		2	1330	Подготовка цикла поиска свободного места в размещении заявки
3	7650		3	3366	
4	5245	Цикл поиска свободного места	4	1724	
5	2366		5	7041	
6	2361		6	3364	
7	2362		7	1327	
0640	2364		0720	3362	
1	5232		1	1326	
2	1363		2	3361	
3	1254		3	5232	БП на начало цикла
4	5211		4	0556	
5	1375		5	7775	
6	3766	Засылка кода задержки, адреса продолжения задачи и ее приоритета + полей в таблицы службы задержек	6	2354	Адреса и константа
7	1376		7	2351	
0650	3762		0730	2346	
1	1374		1	2024	
2	3761		2	6361	
3	5576		3	6240	Текст сообщения
4	0000	Рабочая ячейка	4	5556	
5	0000		5	4250	
6	7041		6	6245	
7	3375		7	2744	

0740	4062		0760	2530	
1	6327		1	0000	
2	5027	Текст сообщения	2	0000	
3	4260	«Установите дату и время»	3	0016	
4	4554		4	0000	
5	7620		5	2041	
6	3700		6	0000	
7	0000		7	2403	
0750	2024		0770	2371	
1	6141	Текст сообщения	1	2357	
2	5651	«Сбой 1»	2	0007	
3	2701		3	7766	
4	2037		4	0000	
5	0000		5	0000	
6	0000		6	0000	
7	7700		7	0557	

Рабочие ячейки, адреса и константы

Блок внутреннего прерывания

→1000	0000	←Адрес возврата	1040	1374	
→1	6002	Запрет прерывания	1	7110	
2	3376	Адрес заявки→р. я.	2	7012	
3	6214	Считывание и запо-	3	1170	
4	3374	минание поля данных	4	3742	
5	1215	Формирование	5	3746	
6	1170	команды 62N ₁	6	3747	
7	3210		7	2367	

Имитация прерывания и запасание признака «Заявка не от диспетчера»

1010	0000	←62N ₁ (команд)1	1050	5260	
1	1600	Пересылка приорите-	1	7000	
2	3375	тов + полей заявки	2	1361	
3	1775	в р. я.	3	7650	
4	3375		4	5317	
5	6201		5	1362	
6	2200	Коррекция адреса возврата	6	7640	
7	1375		7	5270	

Очередь № 1 пуста?

Адрес места прерывания в очереди № 1 фиксировался?

1020	0366	Выделение (допол-	1060	7201	
1	7041	нительного кода)	1	3362	
2	3365	приоритета заявки	2	1000	
3	3367		3	3755	
4	1170		4	1764	
5	7640		5	0366	
6	5236		6	1742	
7	1200		7	3764	

Засылка признака «Адрес заносился»
Адрес продолжения→
→в очередь № 1
Коррекция приорите-
тов + полей прерыва-
нием задачи по
текущему состоянию

1030	7750	Анализ: заявка от диспетчера?	1070	1361	
1	5236		1	1360	
2	1200		2	7710	
3	1343		3	5332	
4	7750		4	1353	
5	5252		5	1354	
6	1200	Имитация прерыва-	6	7710	
7	3000	ния	7	5314	

Есть место в очереди № 1?

Есть место в очереди № 2?

1100	1367	Заявка от диспетчера? 1→в р. я. для [ДРС] Обращение в блок экстренного вывода	1160	7766	
1	7450		1	0000	
2	5305		2	0000	
3	3746		3	0077	
4	1356		4	2530	
5	1357		5	0000	
6	4751		6	7700	
7	7320		7	0000	
1110	1367	Заявка от диспетчера? На выход из прерыв. БП по адресу возвр. Размещение заявки в конце очереди № 2 Заявку → в конец очереди № 1	1170	0246	Резерв
1	7640		1	0000	
2	5770		2	0000	
3	5600		3	0000	
4	4752		4	0000	
5	7000		5	0000	
6	5310		6	0000	
7	4750		7	0000	Рабочие ячейки
1120	7200	Обнуление р. я. для [СМ] и [ДРС] Данные первой задачи из очереди № 1 → в рабочие ячейки	1200	7650	Второе место в очереди № 1 свободно?
1	3747		1	5242	
2	3746		2	1776	
3	1755		3	0375	
4	3000		4	1774	
5	1764		5	7710	
6	0363		6	5242	
7	3742		7	7240	
1130	7300	БП на выход из прерывания Приоритет заявки выше, чем у первой задачи из очереди № 1?	1210	1773	Подготовка цикла
1	5310		1	7041	
2	1764		2	3372	
3	0366		3	1372	
4	1365		4	7041	
5	7740		5	1354	
6	5745		6	3371	
7	7240		7	1771	
1140	1361	Второе место в очереди № 2 свободно?	1220	2371	Цикл формирования адресов и сдвига очереди № 1 на одну позицию (со второго места до конца — освобождение второго места)
1	5744		1	3771	
2	0342		2	1372	
3	5500		3	7041	
4	1200		4	1353	
5	1244		5	3371	
6	0375		6	1771	
7	0376		7	2371	
1150	1400	Адреса, константы и рабочие ячейки	1230	3771	Константа
1	2011		1	2372	
2	1332		2	5213	
3	7773		3	0070	
4	0000		4	1767	
5	2516		5	3776	
6	0006		6	1766	
7	0006		7	3770	Размещение заявки на втором месте в очереди № 1

1240	2773	+1 к длине очереди 1	1320	5330	Подпрограмма формирования начального адреса свободного места в очереди № 2
1	5765		1	7041	
2	4746	Заявку → в конец оч. 1	2	3372	
3	5765		3	1360	
4	1753		4	1361	
5	0233	Формирование команды 82N1 в ячейке 1276	5	2372	
6	1300		6	5324	
7	3276		7	3360	
1250	1763		1330	2763	
1	1364	Есть ли то в очереди № 2?	1	5713	
2	7700		2	0000	
3	5242		3	4313	Подпрограмма размещения заявок в конце очереди № 2
4	4313		4	1767	
5	1351		5	3760	
6	3372		6	2360	
7	1352		7	1766	
1260	3371		1340	3760	
1	1771		1	2360	
2	3350	Размещение в очереди № 2 приоритета + поля, адреса продолжения прерываемой задачи, содержимого ячейки межкубковой связи, [СМ], [ДРС]	2	3760	
3	1750		3	2360	
4	3760		4	3760	
5	2360		5	5732	
6	2371		6	1400	
7	2372		7	1120	
1270	5261		1350	0000	
1	1377		1	7773	
2	7041		2	1353	
3	3372	Запоминание содержимого индексных ячеек из поля команд	3	2530	
4	1377		4	2516	
5	3371		5	0375	
6	0000		6	0376	
7	1771		7	0361	
1300	6201		1360	0000	Адреса, константы в рабочие ячейки
1	3760		1	0015	
2	2360		2	2415	
3	2371		3	1154	
4	2372		4	7773	
5	5276		5	1130	
6	1767		6	1176	
7	3753	Размещение заявки на первом месте в очереди № 1	7	1175	
1310	1766		1370	2517	
1	3754		1	0000	
2	5747		2	0000	
3	0000		3	1161	
4	1362		4	1165	
5	3360		5	3700	
6	1763		6	2531	
7	7450		7	0010	

Блок окончания задач (с адреса 1420)

1400	0000		1	1760	
1	1776		2	3375	
2	1354		3	1773	Цикл восстановления
3	3373		4	3775	в промежуточных
4	1771		5	2373	ячейках и в таблицах
5	3773		6	2360	очереди № 1 данных
6	1776	Подпрограмма раз-	7	2363	задачи из очереди
7	1353	мещения заявки в			№ 2
		конце очереди № 1			
1410	3373		1470	5261	
1	1772		1	1374	
2	3773		2	7041	Цикл восстановления
3	2776		3	3363	содержимого ин-
4	5600		4	1374	дексных ячеек в сво-
5	0070	Константа	5	3360	ем поле команд
6	1313	Адреса	6	1773	
7	1360		7	0000	
1420	6002	Запрет прерыв.	1500	3760	
1	6201		1	6201	
2	7200		2	2373	
3	1767	Очередь № 2 пуста?	3	2360	
4	7650		4	2363	
5	5313		5	5276	
6	7240		6	7240	Уменьшение длины
7	1767	Определение и пере-	7	1767	очереди № 2 на 1
		сылка адреса послед-			
		ней задачи из очере-	1510	3767	
		ди № 2	1	2777	1→в признак «За-
1430	3767		2	5751	прос не от диспетче-
1	4616				ра» и БП на выход
2	1617		3	7240	Уменьшение длины
3	3373	Очередь № 1 исчер-	4	1776	очереди № 1 на 1
4	7240	пана?	5	3776	
5	1776		6	1776	
6	7650		7	7450	Очередь № 1 исчер-
7	5251				пана?
1440	1765		1520	5750	
1	0362	Приоритет последней	1	7041	
2	7041	задачи из очереди	2	3363	Организация цикла
3	3363	№ 2 выше (или ра-	3	1365	
4	1773	вен) приоритета вто-	4	3360	
5	0362	рой задачи из очере-	5	1366	
6	1363	ди 1?	6	3373	
7	7740		7	7240	
1450	5313		1530	1360	
1	1773	Формирование	1	3375	Цикл сдвига очереди
2	0215	команды 62N ₁ в	2	1760	№ 1 на одну пози-
3	1301	ячейке 1477	3	3775	цию вперед
4	3277		4	7240	
5	1352		5	1373	
6	3360	Подготовка цикла	6	3375	
7	1361		7	1773	
1460	3363				

1540	3775		1560	0000
1	2360	Цикл сдвига очереди № 1	1	7773
2	2373		2	3700
3	2363		3	0000
4	5327		4	2415
5	3756	Обнуление рабочих ячеек для [СМ] и [ДРС]	5	2531
6	3755		6	2517
7	5311		7	1154
		БП на окончание пе- ресылок	1570	1075
1550	1600			
1	1123	Адреса, константы и рабочие ячейки	1	1176
2	1553		2	1175
3	2530		3	0000
4	2516		4	0010
5	0375		5	0000
6	0376		6	1161
7	0361		7	1167

Служба вывода (с адреса 1614)

1600	7200		1640	7120	
1	1212	←62N1	1	0000	Возвращение управ- ления функциональ- ной задачей
2	1222		2	0000	
3	3204		3	6001	
4	0000		4	5614	Вставка
5	1613		5	3363	
6	6001		6	5277	
7	2204		7	1371	
1610	5207		1650	1377	
1	5200	поле индицируемой адрес ячейки	1	3370	Формирование те- кущих адресов в таб- лицах очереди выво- да и засылка в них адреса продолжения и приоритета + по- лей
2	0000		2	1214	
3	0000		3	3770	
4	0000		4	1241	
5	6002		5	0365	
6	7200		6	7112	
7	6214		7	7010	
1620	1222	Запоминание поля данных	1660	7000	
1	3241		1	1170	+1 в числу занятых мест Канал вывода свобо- ден?
2	6201		2	3364	
3	1377		3	1372	
4	1376		4	1377	
5	7710		5	3370	
6	5247		6	1767	
7	7000		7	0366	
1630	1375		1670	1364	
1	4774	Возвращения управ- ления функциональ- ной задачи с призна- ком отказа	1	3770	
2	2214		2	7000	
3	7200		3	2377	
4	1373		4	1363	
5	1170		5	7640	
6	3242		6	5576	
7	2214		7	1771	

1700	3362		1740	5753	
1	1772		1	7200	
2	0365		2	1377	Очередь на вывод ис-
3	1222	Вызов и пересылка	3	7640	черпана?
4	3305	поля данных и на-	4	5351	
5	6201	чального адреса ко-	5	3363	Обнуление призна-
6	1762	дограммы первой за-	6	5760	ка «канал занят» и
7	3167	дачи из очереди вы-			БП на выход из пре-
		вода в ячейки 0166,	7	7000	рывания
		0167			
1710	2362		1750	7000	
1	1762		1	7201	Засылка 1 в признак
2	3166		2	5245	«канал занят»
3	6201		3	2062	
4	7000		4	2014	
5	4756	Обращение в блок	5	2163	
		собственно вывода	6	3032	
6	7201		7	6253	
7	1363	Шел экстренный вы-			
		вод?	1760	0246	
1720	7650		1	7707	
1	5332		2	0000	
2	3170		3	0000	
3	1771	Восстановление за-	4	0000	
4	7001	дачи, закончившей	5	0070	Адреса, константы и
5	4571	вывод, в очереди	6	3700	рабочие ячейки
6	2327	№ 1	7	2530	
7	5754	БП на сдвиг очереди			
		вывода			
1730	7240		1770	0000	
1	1377	Уменьшение длины	1	2310	
2	3377	очереди вывода на 1	2	2327	
3	1755		3	6202	
4	7450	Нужен экстренный	4	2041	
5	5341	вывод?	5	0011	
6	3363	Коррекция призна-	6	7761	
7	3755	ков и БП в блок	7	0000	
		экстренного вывода			

Блок экстренного вывода (с адреса 2041)

2000	7200		2020	1371	
1	1776	Канал вывода свобо-	1	3367	
2	7640	ден?	2	1370	
3	5777		3	3366	
4	2776	1→в признак «канал	4	2366	
		занят»	5	1766	
5	1363		6	3770	
6	7450	Начинается экстрен-	7	1367	
7	5576	ный вывод?			
2010	3776	Модификация при-	2030	3366	Цикл сдвига таблиц
1	3363	знака канала и 0→в	1	2366	очереди вывода
		признак «нужен	2	1766	
2	6202	экстренный вывод»	3	3767	
3	5641		4	2367	
4	1775	Формирование счет-	5	2370	
5	3373	чика цикла	6	2373	
6	1372		7	5222	
7	3370				

2040	5765	1		
1	0000	← Адрес возврата	7	2516
2	3364	Запоминание № запроса	2130	0000
3	1363	Есть нереализован-	1	1176
4	7640	ный запрос на экст-	2	0600
5	5212	ренный вывод?	3	0674
6	7240	Засылка признака	4	2024
7	3363	«нужен экстренный	5	5662
		вывод»	6	5240
			7	4727
2050	1364		2140	0000
1	1362	Кодограмму надо до-	1	1327
2	7500	полнять?	2	0000
3	5266		3	0000
4	1361	Формирование ад-	4	0000
5	3360	реса в таблице	5	2037
6	7000		6	2147
7	1776	Канал вывода свобо-	7	2134
		ден?	2150	2260
2060	7640		1	0027
1	5212	Засылка исходных	2	0070
2	1760	данных в ячейки	3	0000
3	3167	0166, 0167 и БП в	4	2246
4	1374	блок вывода	5	2200
5	5756		6	2244
6	1361	Подготовка адреса	7	1764
7	3366	для кодограммы экст-	2160	0000
		ренного вывода	1	2123
2070	1766		2	7772
1	4755		3	0000
2	1364	Подготовка № поля	4	0000
3	1354	для кодограммы экст-	5	1730
4	3353	ренного вывода	6	0000
5	1753		7	0000
6	3353		2170	0000
7	1753		1	2327
2100	0352		2	2310
1	7106		3	0000
2	7004		4	0500
3	1351		5	1777
4	3342		6	1763
5	1364	Засылка № отказа в	7	0246
6	1350	кодограмму	2200	0000
7	3353		1	3236
2110	1753		2	1636
1	3340	Подготовка началь-	3	0230
2	1346	ного адреса	4	3635
3	3360	Начальные адреса	5	1636
4	5257	неформируемых со-	6	0231
5	0352	общений	7	7104
6	0344		2210	7006
7	0750		1	1635
2120	0000	Резерв	2	3635
1	0000		3	1636
2	0000		4	0232
3	1176		5	7112
4	0600		6	7012
5	0674		7	7012
6	1614			

Местоположение ад-
ресов задач, получа-
ющих отказ

Текст сообщения об
отказе в виде: ОТКАЗ
XX—X XXXX

Адреса, константы и
рабочие ячейки

Подпрограмма

2220	3634		2310	0000		
1	1636		1	0000		
2	0233		2	0000		
3	7112	Подпрограмма подго-	3	0000		
4	7010	товки для кодограм-	4	0000		
5	1634	мы кода из ячейки с	5	0000		
6	3634	заданным (на СМ)	6	0000		
7	5600	адресом	7	0000	Адреса	
2230	0007		2320	0000		
1	0070	Константы	1	0000		
2	0700		2	0000		
3	7000		3	0000		
4	2143		4	0000		
5	2144		5	0000		
6	0000	Адреса	6	0000		
7	0000	Рабочая ячейка	7	0000		
2240	2200	Стандартная кодо-	2330	0000		
1	0000	грамма блока экст-	1	0000	Приори-	
2	3700	ренного вывода	2	0000	теты +	
3	1714	Адрес	3	0000	+ поля	
4	3166	Блок стыковки с про-	4	0000		
5	1167	граммным обеспече-	5	0000		
6	3241	нием собственно вы-	6	0000		
7	1252	вода	7	0000		
2250	3167		2340	0000		
1	5643		1	0000		
2	2240	Адрес	2	0000		
3	7000		3	0000		
4	1175		4	0000		
5	0774	Таблица адресов рас-	5	0000		
6	0774	положения полей для	6	0000	Регистры	
7	0170	блока экстренного вы-	7	0000		
2260	2530	вода	2350	0000		
1	0000		1	0000	Адреса	
2	1175		2	0000		
3	0774		3	0000		
4	0774		4	0000	Приори-	
5	0000		5	0000	теты +	
6	0127		6	0000	+ поля	
7	0227		7	0000		
2270	0227	Таблица упакован-	2360	0000		
1	0327	ных кодов для форм-	1	0000		
2	0427	мирования кодограмм	2	0000		
3	0000		3	0000	Регистры	
4	0131		4	0000		
5	0231		5	0000		
6	0231		6	0000		
7	0000		7	0000		
2300	3170	Задача обращения к	2370	0000		
1	4574	оператору: «Устано-	1	0000		
2	2305	вите дату и время»	2	0000		
3	0500		3	0000		
4	5576		4	0000	Адреса	
5	2200	Кодограмма для за-	5	0000		
6	0731	дачи обращения к опе-	6	0000		
7	3700	ратору	7	0000		

Таблицы
службы
вывода

Таблицы
службы
заказан-
ного
времени

Таблицы
службы
задержек

2400	0000	Адреса	Таблицы службы задержек	2	0000	Четвертое место очереди № 2
1	0000			3	0000	
2	0000			4	0000	
3	0000			5	0000	
4	0000			6	0000	
5	0000			7	0000	
6	0000					
7	0000	Приори- теты + + поля		2470	0000	
2410	0000			1	0000	
1	0000			2	0000	
2	0000			3	0000	
3	0000			4	0000	
4	0000			5	0000	
5	0000			6	0000	
6	0000	Приор. + + поля Адрес прод. [ДРС] [СМ] [Яч. 0170]	Данные первой задачи в оч. № 2	7	0000	Пятое место в очере- ди № 2
7	0000			2500	0000	
2420	0000			1	0000	
1	0000			2	0000	
2	0000			3	0000	
3	0000			4	0000	
4	0000			5	0000	
5	0000	Содержи- мое индекс- ных ячеек	6	0000		
6	0000		7	0000		
7	0000		2510	0000		
2430	0000		1	0000		
1	0000		2	0000		
2	0000		3	0000		
3	0000		4	0000		
4	0000	Данные второй зада- чи из очереди № 2	5	0000	Адреса	
5	0000		6	0000		
6	0000		7	0000		
7	0000		2520	0000		
2440	0000		1	0000		
1	0000		2	0000		
2	0000		3	0000		
3	0000	4	0000			
4	0000	5	0000			
5	0000	6	0000			
6	0000	7	0000	Очередь № 1		
7	0000	2530	0000			
2450	0000	1	0000			
1	0000	2	0000			
2	0000	3	0000			
3	0000	4	0000			
4	0000	5	0000			
5	0000	Третье место в очере- ди № 2	6	0000	Приори- теты + + поля	
6	0000		7	0000		
7	0000		2540	0000		
2460	0000		1	0000		
1	0000					

Службы ввода		Таблица адресов исходного набора задач		Службы ввода		Таблица приоритетов + + полей исходного набора задач
2	4047	2560	0000	1	0100	
3	1652	1	0100	2	0011	
4	0000	2	0011	3	0000	
5	0000	3	0000	4	0000	
6	0000	4	0000	5	0000	
7	0000	5	0000	6	0000	
		6	0000	7	0000	
2550	0000	2570	0000			
1	0000	1	0000			
2	0000	2	0000			
3	0000	3	0000			
4	0000	4	0000			
5	0000	5	0000			
6	0000	6	0000			
7	0000	7	0000			

Программы печати бланков

→ 2600 7777		2640 3221		Подпрограмма последовательной упаковки 6-разрядных кодов по два в одну ячейку	2650 3200	Подпрограмма преобразования двоичных чисел в десятичную упакованную форму из семи цифр
1	0000	1	1223			
2	0000	2	3222		1	3224
3	2201	3	1227		2	7300
4	5218	4	3225		3	1221
5	1600	5	1230		4	1626
6	3600	6	3226		5	3221
7	5217	7	1231		6	7004
2610 2200					7	1220
1	7106				2660 1625	
2	7006				1	7510
3	7006				2	5266
4	3600				3	3220
5	7240				4	2224
6	3201				5	5252
7	5602				6	7200
2620 0000					7	1626
1	0000				2670 7041	
2	0000				1	1221
3	7772				2	3221
4	0000				3	1224
5	0000				4	4202
6	0000				5	2225
7	2704				6	2226
2630 2712					7	2222
1	2717					
2	0000					
3	3201					
4	6201					
5	1724					
6	3220					
7	1725					

2700	5251		→ 1	0000	
1	1221		2	7012	
2	4202		3	7012	Подпрограмма вы-
3	5632		4	7012	деления левого сим-
4	7413		5	0326	вола
5	7747		6	5771	
6	7775		7	3754	
7	7777				
2710	7777		→ 3000	0000	Подпрограмма уста-
1	7777		1	6201	новки номера куба
2	6700		2	5600	кодограммы
3	4540		→ 3	0000	
4	4360		4	4625	
5	6030		5	0213	
6	7634		6	1265	Подпрограмма пе-
7	7766		7	3201	рехода на другой сег-
2720	0000		3010	1167	мент кодограммы
1	0000		1	3165	
2	0000		2	5603	
3	0000		3	0070	
4	3207		→ 4	0000	
5	3210		5	4200	
6	0077		6	1565	
7	7700		7	3166	Подпрограмма вы-
2730	0000		3020	2165	борки двух текущих
1	2737	Подпрограмма пе-	1	1565	ячеек кодограммы
2	2727	чати заданного чис-	2	3167	
3	0000	ла пробелов	3	6201	
4	0326		4	5614	
5	1327		5	3722	
6	3330		6	3155	
7	6201		7	3457	
2740	1332		3030	0007	
1	4743		1	2771	
2	5733		→ 2	0000	
3	3457		3	1166	
4	3014		4	4631	
5	3240		5	0230	
6	0000		6	3200	
7	0000		7	1166	
→ 2750	0000		3040	7006	
1	4777		1	7006	
2	7041		2	0230	
3	7450		3	7041	
4	5367	Подпрограмма печат-	4	1200	
5	3346	ти группы чисел одина-	5	7440	
6	1167	кового формата,	6	5251	
7	3347	расположенных в па-	7	1322	
2760	4744	мяти произвольно	3050	5263	
1	1347		1	7500	Подпрограмма уп-
2	3166		2	5256	равления печатью ко-
3	1166		3	7200	дограмм (выделение
4	4745		4	1226	команд кодограмм и
5	2346		5	5264	их исполнение)
6	5360		6	7041	
7	4744		7	1200	
2770	5750				

3060	7640		3150	3201	
1	5375		1	1328	
2	1226		2	3165	
3	1200		3	5730	
4	3200		4	3773	
5	6201		5	3232	
6	1600		6	3343	
7	3200		7	3353	
3070	4754		3160	3403	
1	1166		1	3415	
2	4600		2	3003	
3	4214		3	3130	Таблица адресов испол- нительных про- грамм
4	5233		4	3605	
→ 5	0000		5	3773	
6	4625		6	4011	
7	0213		7	3075	
3100	1265		3170	3111	
1	3304		1	3704	
2	7240	Подпрограмма печат- ти группы упакован- ных символов, ука- занной адресом в ко- дограмме	2	3653	
3	1167		3	3657	
4	0000		4	2750	
5	4627		5	1166	
6	4214		6	4772	Вставка
7	5675		7	5632	
3110	7776		3200	7000	Рабочие ячейки и константы
→ 1	0000		1	3754	
2	7200		2	0000	
3	1310		→ 3	0000	
4	1165	Подпрограмма печат- ти группы упакован- ных символов, вклю- ченных в кодограмму	4	1236	Подпрограмма воз- врата каретки
5	4200		5	4630	
6	4627		6	5603	
7	6201		7	0000	
3120	7201		3210	0000	
1	1724		1	3014	Рабочие ячейки и константы
2	3165		2	2632	
3	5711		3	2733	
4	3576		4	3533	
5	0000		→ 5	0000	
6	0000		6	6201	Подпрограмма уста- новки номера куба чисел
7	0000		7	5615	
→ 3130	0000		3220	6201	
1	4625		1	0070	
2	7201		→ 2	0000	
3	1165		3	0221	Подпрограмма за- дания номера куба чисел
4	3326		4	1220	
5	1201		5	3216	
6	3325		6	5622	
7	1232		7	0040	
3140	3327		3230	3403	
1	1166		1	0012	Константы
2	4203	Подпрограмма вклю- чения другой кодо- граммы в данную ко- дограмму	→ 2	0000	
3	4214		3	2165	Вход в подпрограм- му печати одного чис- ла
4	4232		4	4240	
5	1327		5	5632	
6	3232		6	0020	Константы
7	1325		7	0017	

→ 3240	0000		3330	7041	
1	4215		1	3202	Вспомогательная
2	0227		2	4611	подпрограмма для
3	7650		3	1202	организации печати
4	5257		4	3342	группы десятичных
5	1567		5	1166	чисел
6	7510		6	4240	
7	7041		7	2342	
3250	3210		3340	5335	
1	1567		1	5723	
2	7710		2	0000	
3	7201		→ 3	0000	
4	1231		4	7240	
5	4630		5	3232	Подпрограмма печат
6	5261		6	4323	ти строки чисел
7	1567		7	4611	
3260	3210		3350	5743	
1	1166		1	0000	
2	0236		2	2771	
3	7640		→ 3	0000	
4	5267		4	0321	
5	3207		5	7041	
6	5313		6	7450	
7	4215		7	5723	
3270	1567		3360	3222	
1	7510	Подпрограмма печат	1	4611	
2	5300	ти десятичного чис	2	1166	
3	3207	ла, представленного	3	4752	Подпрограмма ор
4	2167	в памяти одно- или	4	3351	ганизации печати не
5	1567	двухъязычным дво-	5	7240	скольких строк деся
6	3210	ичным числом	6	3232	тичных чисел
7	5313		7	4203	
3300	7300		3370	1351	
1	1567		1	4613	
2	7040		2	4323	
3	3207		3	2222	
4	2167		4	5367	
5	1567		5	4203	
6	7041		6	4611	
7	3210		7	5753	
3310	7004		3400	2771	
1	1207		1	6201	
2	3207		2	3735	
3	4612		→ 3	0000	Подпрограмма вы
4	4614		4	6046	вода на печать от
5	1166		5	7200	дельного символа
6	0237		6	6214	
7	4613		7	1201	
3320	2167		3410	3212	
1	0077		1	5602	
2	5640		2	6201	
→ 3	0000		3	7200	
4	2232		4	5603	
5	5333		→ 5	0000	
6	1166		6	3257	
7	4601		7	1227	

3420	4203		3510	4203	
1	1257		1	2230	
2	4203	Подпрограмма вы-	2	5307	
3	1226	вода на печать от-	3	5262	
4	4203	дельного символа с	4	0000	
5	5615	нижнего регистра	5	1231	
6	0024		6	7640	
7	0026		7	5325	
3430	0000		3520	2376	
→ 1	0000		1	2231	
2	3230		2	1776	
3	1230		3	4600	Подпрограмма рас-
4	4600		4	5330	паковки упакован-
5	4241		5	3231	ных символов
6	1230		6	1776	
7	4241		7	0332	
3440	5631		3530	3241	
1	0000		1	5714	
2	3253	Подпрограмма вось-	2	0077	
3	1253	меричной печати со-	→ 3	0000	
4	5645	держимого сумматор	4	1166	
5	3767	ра	5	7006	
6	4203		6	7006	
7	1253		7	0254	
3450	0254		3540	1374	
1	4203		1	3230	
2	5641		2	1166	
3	0000		3	4600	
4	0007		4	0254	
5	7701		5	1374	
6	7741		6	3375	
→ 7	0000		7	1374	
3460	3376		3550	7001	
1	3231	Подпрограмма печат-	1	3253	
2	4314	ти упакованных сим-	2	1377	
3	1241	волов	3	3376	
4	1256		4	3231	
5	7650		5	4314	
6	5657		6	2375	
7	1241		7	5362	Подпрограмма печат-
3470	1255		3560	1350	ти дробного десятич-
1	7650		1	4215	ного числа
2	5276		2	2230	
3	1241		3	7000	
4	4203		4	1230	
5	5262		5	7710	
6	4314		6	5371	
7	1241		7	1241	
3500	7440		3570	4203	
1	5304		1	2253	
2	4314		2	5355	
3	5262		3	5733	
4	7041		4	7770	
5	3230		5	0000	
6	4314		6	0000	
7	1241		7	2717	

3600	3216		3670	1567	
1	3222		1	4645	
2	3130		2	1251	
3	3624		3	4703	
4	0000		4	2204	
→ 5	0000		5	5277	
6	4703		6	5332	
7	1600		7	2167	
3610	3204		3700	2247	
1	4601		1	5267	
2	1203		2	5264	
3	3167		3	2733	
4	3166		→ 4	0000	
5	4602		5	0250	
6	1204	Подпрограмма печати текущего времени (часы — минуты) и даты (число—месяц)	6	7041	
7	3600		7	7450	
3620	7240		3710	5320	
1	1165		1	3240	
2	3165		2	4641	Подпрограмма печати неупакованной группы символов
3	5605		3	1567	
4	2000		4	4642	
5	0572		5	2167	
6	0313		6	2240	
7	2000		7	5313	
3630	0573		3720	4643	
1	0314		1	5704	
2	2000		→ 2	0000	
3	0040		3	0251	
4	0313		4	7640	
5	2002		5	4644	
6	0041		6	1166	
7	3700		7	0353	
3640	0000		3730	4703	
1	3215		1	5351	
2	3403		2	4643	
3	3014		3	5657	Вспомогательная подпрограмма печати пробелов и возврата каретки
4	3203		4	0007	
5	3431	Рабочие ячейки и константы	5	6201	
6	7770		6	5737	
7	0000		7	2000	
3650	0077		3740	7000	
1	0004		1	7000	
2	3215		2	7000	
→ 3	0000		3	7000	
4	4354		4	7000	Резерв
5	4703		5	7000	
6	5653		6	7000	
7	0000		7	7000	
3660	0250		3750	7000	
1	7001		1	1166	Вставка
2	7041		2	5722	
3	3204	Подпрограмма восьмеричной печати содержимого указанной группы ячеек	3	0003	
4	1246		4	0000	
5	3247		5	0363	Вспомогательная подпрограмма для возврата каретки
6	4644		6	7640	
7	4652		7	4765	

3760	1166		3770	7012	
1	0364		1	0334	
2	5754		2	5766	
3	0040		3	0000	
4	0037		4	1377	Подпрограмма уста-
5	3203	Константы	5	4642	новки верхнего регис-
6	3446		6	5773	тра
7	7010	Сдвиг на 3 разряда вправо	7	0024	

Программное обеспечение диалога

4000	0002		4050	6036	
1	0004		1	0233	
2	0001		2	3245	
3	7200		3	1245	
4	3255		4	1235	
5	4630		5	7650	
6	3222		6	5576	
7	3203		7	1245	
4010	2733		4060	1236	
→ 1	0000		1	7650	
2	0200		2	5576	
3	7650		3	1245	
4	5217	Подпрограмма орга-	4	1237	
5	1203	низации печати моду-	5	7640	
6	5220	ля дробного десятич-	6	5304	
7	1205	ного числа	7	3243	
4020	3604		4070	1241	
1	1166		1	3242	
2	4606		2	1303	Блок приема и реали-
3	1166		3	3245	зации команд опера-
4	0201		4	3642	тора, поступающих с
5	7640		5	2242	«Консула»
6	4607		6	2245	
7	1166		7	5274	
4030	0202		4100	1241	
1	4610		1	3242	
2	5611		2	5576	
3	0077		3	7766	
4	3600		4	1243	
5	7751		5	7640	
6	7760		6	5576	
7	7764		7	1245	
4040	7747		4110	1240	
1	4166		1	7650	
2	0000		2	5323	
3	0001		3	1242	
4	4545		4	1234	
5	0000		5	7650	
6	4525		6	5350	
7	6201		7	1245	

4120	3642		4210	7106	
1	2242		1	2011	
2	5576		2	5206	
3	2243	Блок приема и реализации команд оператора	3	1410	
4	1366		4	5602	
5	7106		5	6201	
6	7006		6	0000	Рабочие ячейки и константы
7	7006		7	0000	
4130	1367		4220	4167	
1	3245		1	4170	
2	1355		2	4173	
3	3010		3	0070	
4	1246		4	1621	
5	3011		5	7106	
6	1244		6	7104	Программа задания номера куба данных при диалоге
7	3354		7	0223	
4140	1411		4230	1215	
1	7041		1	3234	
2	1245		2	5375	
3	7650		3	0000	Подпрограмма установки номера куба
4	5357		4	6201	
5	2354		5	5633	
6	2010		6	4233	
7	5340		7	1217	
4150	4756		4240	3616	Запись по адресу
1	4367		1	5576	
2	0500		2	0000	
3	5576		3	3010	
4	0000		4	1410	
5	7761		5	3256	Подпрограмма организации преобразования в двоичную систему 4-значного десятичного числа
6	4572		6	1410	
7	5760		7	7106	
4160	4564		4250	7106	
1	1754		1	1410	
2	3354		2	7106	
3	4765		3	7106	
4	5754		4	1410	
5	4344		5	4323	
6	0000		6	0000	
7	0000		7	5642	
4170	0000		4260	1220	
1	0000		1	4202	
2	0000		2	3217	
3	0000	Поле команд оператора	3	1222	Подготовка восьмеричной записи
4	0000		4	4202	
5	0000		5	3216	
6	0000		6	5576	
7	0000		7	1220	Подготовка десятичной записи
4200	7000		4270	4242	
1	7775		1	5262	
2	0000	Подпрограмма формирования адреса из 4 восьмеричных цифр (из последовательных ячеек памяти)	2	0000	
3	3010		3	3315	
4	1201		4	1315	
5	3011		5	0317	
6	1410		6	7110	
7	7104		7	3316	

4300	1316		7	3356	
1	7012		4370	7050	Кодограмма сооб-
2	1316		1	4152	щения об ошибке в
3	1315		2	4037	команде
4	0320		3	0501	
5	3316		4	4351	
6	1316		5	1621	
7	7110	Стандартная подпро-	6	3343	Вставка
		грамма перевода 3-	7	5576	
4310	1316	значного десятичного	4400	3216	
1	7012	числа в двоичную си-	1	4234	
2	7041	стему	2	0320	
3	1315		3	0000	
4	5672		4	4002	
5	0000		5	0000	
6	0000		6	3700	
7	7400		→ 7	1601	
4320	7760		4410	3600	
1	1750		1	1230	
→ 2	0000		2	4631	Программа десятич-
3	0000		3	3205	ной печати содержи-
4	4272		4	1632	мого группы ячеек,
5	3322		5	0227	указанных в команде
6	1723	Подпрограмма орга-	6	7001	оператора
7	7041	низации перевода в	7	1226	
		двоичную систему 4-	4420	3203	
4330	7450	значного десятичного	1	4670	
1	5336	числа, размещенно-	2	4402	
2	3342	го в двух ячейках	3	0500	
3	1321		4	5576	
4	2342		5	4175	
5	5333		6	0100	
6	1322		7	0017	
7	2323		4430	4167	
4340	5723		1	4202	
1	4572		2	4174	
2	0000		3	6600	
3	0000		4	0077	
4	0000		5	0000	
5	4741		6	0000	
6	4351		7	3700	
7	0500		→ 4440	1601	
4350	5744		1	3600	
1	1100	Подпрограмма печат-	2	1230	
2	4402	ти команды операто-	3	4631	
3	4166	ра, принятой с «Кон-	4	3236	
4	5501	сула»	5	1272	
5	4404		6	4631	
6	4170		7	0234	
7	5501		4450	1233	
4360	4404		1	3235	Программа восьмери-
1	4174		2	4670	ной печати содержи-
2	5502		3	4435	мого группы ячеек,
3	4401		4	0500	указанных в команде
4	4343		5	5576	оператора
5	5502		6	4173	
6	3700		7	0700	

4460	3700	1		4530	4742	
→ 1	4670			1	4744	
2	4457			2	5757	Таблица кодов операций диалога
3	0500	Программа печати		3	5042	
4	5576	времени и даты по запросу		4	6342	
5	0572			5	0000	
6	0573			6	6352	
7	0574			7	6142	
4470	4572			4540	0000	
1	4165			1	0000	
2	4171			2	0000	Место для включения дополнительных команд оператора
→ 3	0012			3	0000	
4	0000			4	0000	
5	3010			5	4440	
6	1410			6	4407	
7	7041			7	4260	Таблица адресов исполняемых диалоговых программ
4500	7450			4550	4267	
1	5306			1	4150	
2	3011	Подпрограмма перевода в двоичную систему 2-значного десятичного числа		2	4461	
3	1273			3	4510	
4	2011			4	4236	
5	5303			5	4224	
6	1410			6	4633	
7	5674			7	4056	Место для адресов дополнительных программ
→ 4510	1230			4560	4056	
1	4274			1	4056	
2	3665			2	4056	
3	1272			3	4056	
4	4274			4	3170	
5	3666			5	1371	
6	3667	Программа записи времени и даты с «Консула»		6	4575	
7	1256			7	5770	
4520	4274			4570	4161	Связь с программой-диспетчером
1	3040			1	0001	
2	1225			2	0000	
3	4274			3	1372	
4	3041			4	3574	
5	5576			5	3170	
6	5742			6	5777	
7	5744			7	1615	

Программа вызова сводок

4600	0000		4610	0000	
1	0000		1	0000	
2	0000		2	0000	
3	0000	Номера кубов и адреса сводок	3	0000	
4	0000		4	0000	
5	0000		5	0000	
6	0000		6	0000	
7	0000		7	0000	

4620	0000
1	0000
2	0000
3	0000
4	4170
5	0017
6	7766
7	4572

4630	0007
1	0500
2	4577
3	1624
4	0225
5	1226
6	7700
7	5662

4640	1624
1	0225
2	1232
3	3010
4	1410
5	0230
6	7106
7	7004

4650	1231
1	3260
2	1410
3	3257
4	7000
5	7000
6	4627
7	0000

4660	0000
1	5576
2	4150
3	7000

Текст программы вы-
зова сводок

Местный диспетчер (1-й куб МОЗУ)

→ 3400	0000	Адрес возврата
1	6002	Запрет прерывания
2	3376	Запоминание [СМ]
3	7000	
4	6214	Считывание и запо- минание поля дан- ных
5	7112	
6	7010	
7	3374	

3430	1372	
1	6202	
2	4771	
3	1231	Восстановление блока
4	3767	экстренного вывода
5	5261	В блок окончания за- дач
6	7200	Размещение запроса
7	4322	в очереди № 3

3410	7000	Считывание поля команд
1	6201	
2	1775	
3	6211	
4	1374	Формирование и за- поминание полей
5	3374	
6	1377	
7	7450	Выходной комму- татор свободен?

3440	1763	Пересылка [СМ]
1	3153	
2	0000	Константа
3	6201	Пересылка адреса
4	1200	продолжения задачи
5	3642	
6	1374	Пересылка полей в
7	3760	ячейку для [РЗП]

3420	5236	
1	1373	Есть место в очереди № 3?
2	7710	
3	5254	
4	6201	Подготовка обраще- ния и обращение в блок экстренного вы- вода
5	1370	
6	3767	
7	7000	

3450	3757	Обнуление рабочих
1	3756	ячеек для [ДРС] в
		[СМ]
2	6202	В блок выхода из
3	5755	прерывания
4	7240	
5	1377	Второе место в оче-
6	7640	реди № 3 свободно?
7	5263	

3460	4322	Размещение запро-	3550	0000	
1	6202	са в конце очереди	1	0000	
2	5576	№ 3 в БП в блок	2	3561	
		окончания задач	3	0000	
3	1754		4	3772	
4	0364		5	0246	
5	7041		6	0376	
6	3366		7	0375	
7	6201				
3470	1765	Подготовка и срав-	3560	0342	
1	6211	нение приоритетов	1	3762	
2	0364	заявки и второй за-	2	3771	Адреса, константы
3	1366	дачи из очереди № 3	3	3753	и рабочие ячейки
4	7700		4	7700	
5	5260		5	2530	
6	1137		6	0000	
7	3353		7	2011	
3500	1352	Подготовка (двой-	3570	6212	
1	3366	ного) цикла	1	2041	
2	1377		2	0012	
3	7041		3	7771	
4	3351		4	0000	
5	1351		5	0170	
6	7040		6	0000	
7	1766		7	0000	
3510	3360		3600	7201	
1	1750		1	1707	
2	2350	Цикл сдвига таблиц	2	3306	Размещение заявки
3	3750	очереди № 3 на 1 по-	3	7201	на втором месте в
4	2351	зицию навод	4	3707	очереди № 3
5	5305		5	4705	
6	2366		6	1306	
7	2353		7	3707	
3520	5302		3610	6202	В блок окончания за-
1	5377	На следующую страницу	1	5576	дач
→ 2	0000		2	7000	
3	1363		→ 3	0000	Адрес возврата
4	1377		4	6002	Запрет прерывания
5	3366	Подпрограмма раз-	5	7200	Формирование полей
6	1376	мещения заявки в	6	6214	в ячейке 3574
7	3766	конце очереди № 3	7	7112	
3530	1361		3620	7010	
1	1377		1	3704	
2	3366		2	6201	
3	1200		3	1703	
4	3766		4	6211	
5	1362		5	1704	
6	1377		6	3704	
7	3366		7	7000	
3540	6201		3630	7240	Уменьшение длины
1	1765		1	1707	очереди № 3 на 1
2	6211		2	3707	
3	0364		3	1707	Очередь № 3 исчер-
4	1374		4	7640	пана?
5	3766		5	5242	На продолжение за-
6	2377		6	1213	дачи
7	5722		7	6201	

3640	5702		3720	1712	Пересылка [СМ]
1	7775		1	3153	
2	1241		2	1704	Пересылка полей за-
3	3274		3	3315	дача на текущую
4	1276				страницу
5	3275	Подготовка (двой-	4	7000	Резерв
6	1707	ного) цикла	5	7000	
7	7041		6	1713	
			7	6201	
3650	3306		3730	3711	
1	1675		1	6211	
2	3273		2	1714	
3	7201	Цикл сдвига очереди	3	6201	
4	1273	№ 3 на 1 позицию	4	3710	
5	3272	вперед	5	7000	
6	1672		6	1213	
7	3673		7	3671	
3660	2273		3740	1315	
1	2306		1	0316	Пересылка данных за-
2	5253		2	3703	дачи, освободившей
3	2275		3	1315	МД, в блок внутрен-
4	2274		4	0317	него прерывания
5	5246		5	7106	
6	5320	На обход констант	6	7004	
7	1174		7	3667	
3670	1017		3750	6202	В блок внутреннего
1	1000		1	5670	прерывания
2	0000		2	0000	
3	0000		3	0000	
4	0000		4	0000	
5	0000		5	0000	
6	3712		6	0000	
7	3754		7	0000	[СМ]
3700	3763		3760	0000	
1	3772	Адреса в рабочие	1	0000	
2	3445	ячейки	2	0000	
3	0170		3	0000	
4	3574		4	0000	Адреса
5	3522		5	0000	
6	0000		6	0000	
7	3577		7	0000	
3710	1175		3770	0000	
1	1176		1	0000	
2	3753		2	0000	
3	3762		3	0000	
4	3771		4	0000	Приори-
5	0000		5	0000	теты +
6	0070	Константы	6	0000	+ поля
7	0007		7	0000	

Таблицы очереди № 3

СПИСОК ЛИТЕРАТУРЫ

1. Автоматизированные системы управления. Применение вычислительной техники и автоматизированных систем управления на предприятиях и в отраслях промышленности. Материалы Первого Всесоюзного совещания. –М.: «Экономика», 1972.
2. Папернов А.А. Логические основы цифровых машин и программирования –М.: «Наука», 1965.
3. Гутер Р.С. и др. Практика программирования. Справочник. М., «Наука», 1965.
4. Глушков В.М. Об оптимальном объеме оперативных запоминающих устройств электронных цифровых машин. — ДАН УССР, 1960, № 5.
5. Введение в программирование. Вычислительные машины РДР-8 М., Всесоюзная Торговая палата, 1971.
6. Ступин Ю.В. Основы программирования на ФОКАЛе. Отредактированный перевод с англ. Препринт, Физ. ин-т им. Лебедева АН СССР, –М.: 1971.
7. Меррей-Шелли Р. Программирование на Фортране. Пер. с англ. –М.: «Энергия», 1971.
8. Малые вычислительные машины. Тематический указатель литературы, Сер. Контрольно-измерительная аппаратура. Вып. 6 (142), –М.: Ин-т «Электроника», 1970.
9. Мини-ЭВМ (тематическая подборка). Рига. Латв.- республиканский ИНТИП, 1971.
10. Hileman J.R. Justifying a minicomputer for process control. — «Chem. Eng.», 1972, v. 79. № 12, p. 61—66
11. Coughanour D.R. New Algorithms for Control. — «Chem. Eng.», 1969, v. 76, № 12.
12. Цыпкин Я.З. Адаптация и обучение в автоматических системах. –М.: «Наука», 1968.
13. Колин К.К., Липаев В.В. Проектирование алгоритмов управляющих ЦВМ. –М.: «Сов. радио», 1970.
- Колин К.К., Липаев В.В., Серебровский Л.А. Математическое обеспечение управляющих ЦВМ. –М.: «Сов. радио», 1972.
14. Виленкин С.Я., Трахтенгерц Э.А. Математическое обеспечение управляющих вычислительных машин. –М.: «Энергия», 1972.
15. Липаев В.В. Определение количества приоритетов при диспетчеризации управляющих ЦВМ с неограниченной буферной памятью. — «Изв. АН СССР, Техническая кибернетика», 1972, № 5.
16. Кулик В. Т. Алгоритмизация объектов управления. Справочник. Киев, «Наукова думка». 1968.
17. Вычислительная техника для управления производственными процессами. Справочник. –М.: «Энергия», 1971.
18. Белостоцкий А.А., Чухман В.Н. Исследование и организация производства при создании автоматизированных систем управления М., «Металлургия», 1971.
19. Ли Т.Г., Адамс Г.Э., Гейнз У.М. Управление процессами с помощью ЭВМ. Пер. с англ. –М.: «Сов. радио», 1972.
20. Чугани П. Основные правила разработки информационных систем управления», «Data Processing», 1972, v. 14, № 3.
21. Емельянов С.В., Матич Б.П., Костылева Н.Е. Универсальная унифицированная система управления переменной структуры. — «Приборы и системы управления», 1973, № 12.
22. ТИИЭТ, т. 58, № 1, 1970. Тематический выпуск «Автоматизированные системы управления производством».
23. Harrison T. J., Homiak R. L., Merckel G. U. IBM System/7 and plant automation. — «IBM J. Res. and Develop», 1970, v. 14, № 6.
24. «Электронная промышленность», 1974, № 1. Тематический выпуск по АСУ ТП.
25. Илюшин С.В., Виноградов В.П., Зудин А.Г. и др. Автоматизированная система комплексного управления производством цветных кинескопов. — «Электронная промышленность», 1972, № 1.
26. Шершевский А.С. Основные цели и задачи АСУ ТП в электронной промышленности — «Электронная промышленность», 1972, № 5.
27. Котельников Ю.Н., Крочев А.М., Моросанова И.М. и др. Система управления группой вакуумных установок на основе ЭВМ «Электроника-100» — «Электронная промышленность», 1972, № 6.
28. Кон Н., Биддл Мл., Лекс Мл. и др. Применение управляющих вычислительных машин в электроэнергетике. — ТИИЭТ, 1970, т. 58, № 1.
29. Трояновский В.П., Быков Ю.М. Вопросы рационального построения АСУТП на основе системы коллективного пользования. — «Электронная промышленность», 1972, № 4.
30. Пугачев В. С. Теория случайных функций и её применение в задачах автоматического управления, –М.: Физматгиз., 1962.
31. Быков Ю.М. Трояновский В.М. и др. О некоторых информационных принципах в задачах расчета технико-экономических показателей производства — В кн.: Вопросы определения экономической эффективности автоматизации химических производств –М.: НИИТЭХИМ, 1967.
32. Балашов Л.Н., Быков Ю.М, Исаев А.И. и др. Об особенностях приборного и алгоритмического обеспечения информационных систем при управлении динамическими объектами. — «Автоматизация химических производств», 1971, вып. 3.
33. Ицкович Э.Л. О вычислении сводных показателей работы производства в системе автоматического контроля. — «Автоматика и телемеханика», 1965, № 7.
34. Бородюк В. П., Круг Г. К. Некоторые вопросы организации эксперимента для отыскания математического описания технологических процессов. — В кн.: Автоматическое управление и вычислительная техника. Вып. 6. –М.: «Машиностроение», 1964.
35. Hill I. D. Woldn't it be nice if we could write computer programs in ordinary English — or would it? — «The computer Bulletin», 1972, v. 16, № 6, pp. 306-311.
36. Mecklenburgh J.C. May P.A. Protran, a Fortran based computer language for process control «Aulomatica», 1970, v, 6, №4.

37. Schoeffler J.D., Wilmott T., Dedourek J. Programming language for industrial process control. 2-nd IFAC/IFIP International Conference, Menton, France, 1968.
38. Быков Ю.М. О статистической точности восстанавливающих элементов при импульсной передаче случайных сигналов. — «Изв. АН СССР, Техническая кибернетика», №1/1965.
39. Быков Ю.М., Еникеев Ш.Г., Ружников Е.М. Статистическая точность преобразования информации в комбинированной системе. — В кн.: Вычислительная техника в управлении —М.: «Наука», 1966.
40. Ланцош К. Практические методы прикладного анализа. Пер. с англ., —М.: Физматгиз, 1961.
41. Красовский А. А. Динамика непрерывных самонастраивающихся систем. —М.: Физматгиз, 1963.
42. Трояновский В.М., Черкасов Ю.Н. Малые машины и системы коллективного пользования в экспериментальных научных исследованиях. — В кн.: Автоматизация научных исследований в химии Материалы V Всесоюзной школы Рига, «Зинатне», 1975
43. Фаддеев Д.К., Фаддеева В.Н. Вычислительные методы линейной алгебры. Физматгиз, 1964.

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ	3
-------------------	---

Часть 1

ПРОГРАММИРОВАНИЕ НА «ЭЛЕКТРОНИКЕ-100»	3
---	---

1 ХАРАКТЕРИСТИКА МАШИНЫ «ЭЛЕКТРОНИКА-100»	5
1.1. УСТРОЙСТВО И ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ МАШИНЫ	5
1.2. ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА	8
1.3. ПЕРСПЕКТИВЫ РАЗВИТИЯ	9
2. РАЗРЯДНАЯ СЕТКА МАШИНЫ И её ИСПОЛЬЗОВАНИЕ	10
2.1. ОБЩИЕ СВЕДЕНИЯ.....	10
2.2. ЧЕТЫРЕ СТРУКТУРЫ	10
2.3. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В МАШИНЕ	13
3. СИСТЕМА КОМАНД	14
3.1. КОМАНДЫ ОБРАЩЕНИЯ К ПАМЯТИ.....	14
3.2. АДРЕСНАЯ ЧАСТЬ КОМАНД ОБРАЩЕНИЯ К ПАМЯТИ	17
3.3. КОМАНДЫ ОБРАЩЕНИЯ К ВНЕШНИМ И ПЕРИФЕРИЙНЫМ УСТРОЙСТВАМ	19
3.4. ПРОГРАММНОЕ ПРЕРЫВАНИЕ	24
3.5. ГРУППЫ МИКРОКОМАНД	25
4. ПРИЕМЫ ПРОГРАММИРОВАНИЯ	29
4.1. ОБЩИЕ СВЕДЕНИЯ.....	29
4.2. ОСОБЕННОСТИ, СВЯЗАННЫЕ С ОДНОАДРЕСНОСТЬЮ МАШИНЫ И СТРАНИЧНОЙ ОРГАНИЗАЦИЕЙ ПАМЯТИ	29
4.3. ОРГАНИЗАЦИЯ РАЗВЕТВЛЕНИИ	31
4.4. ЦИКЛИЧЕСКИЕ ПРОГРАММЫ	32
4.5. ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ	36
4.6. ОБСЛУЖИВАНИЕ ПРЕРЫВАНИЙ.....	37
4.7. ОРГАНИЗАЦИЯ АРИФМЕТИЧЕСКИХ ВЫЧИСЛЕНИЙ	40
4.8 РАБОТА С ТАБЛИЦАМИ	44
4.9. ПРИНЦИПЫ ОРГАНИЗАЦИИ ПЕЧАТИ	46
4.10. ОСОБЕННОСТИ РАБОТЫ С РАСШИРЕННЫМ МОЗУ	48
4.11. К СИНТЕЗУ — ЧЕРЕЗ АНАЛИЗ	52
5. ПРАКТИЧЕСКАЯ РАБОТА НА МАШИНЕ	57
5.1. ПУЛЬТ ПРОЦЕССОРА И БЛОКА УПРАВЛЕНИЯ УСТРОЙСТВАМИ ВВОДА - ВЫВОДА.....	57
5.2. УСТРОЙСТВА И ПРОГРАММЫ ВВОДА — ВЫВОДА	59
5.3. ПОДГОТОВКА И ВКЛЮЧЕНИЕ МАШИНЫ.....	61
5.4. ЗАПУСК И ОТЛАДКА ПРОГРАММ.	61
5.5. МАЛЕНЬКИЕ ХИТРОСТИ.....	65
6. СРЕДСТВА АВТОМАТИЗАЦИИ ПРОГРАММИРОВАНИЯ.....	67

Часть 2

ПРИМЕНЕНИЕ «ЭЛЕКТРОНИКИ-100»	72
7. ЗАДАЧИ АСУ ТП.....	73
8. ПРОГРАММА-ДИСПЕТЧЕР.....	75
8.1. ОСОБЕННОСТИ ПОСТРОЕНИЯ ДИСПЕТЧЕРА ДЛЯ МИНИМАЛЬНОГО КОМПЛЕКТА ЭВМ	75
8.2. ОБНАРУЖЕНИЕ ИСТОЧНИКА ПРЕРЫВАНИЯ.....	77

8.3.	ОЧЕРЕДИ НА РЕШЕНИЕ ЗАДАЧ	79
8.4.	СЛУЖБЫ ДИСПЕТЧЕРА	81
8.5.	ДИАГНОСТИКА И ФИКСАЦИЯ ОТКАЗОВ	84
8.6.	РЕАЛИЗАЦИЯ ДИСПЕТЧЕРА ДЛЯ МИНИМАЛЬНОГО КОМПЛЕКТА ЭВМ И ДЛЯ МАШИН С РАСШИРЕННОЙ ПАМЯТЬЮ	85
8.7.	ОБМЕН ИНФОРМАЦИЕЙ С ЦВМ, РАБОТАЮЩИМИ В АСУП И В СИСТЕМЕ КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ	87
8.8.	ДРУГИЕ РАСШИРЕНИЯ ФУНКЦИЙ ДИСПЕТЧЕРА	89
9.	ТИПОВАЯ СТРУКТУРА СИСТЕМЫ И БЛОЧНЫЙ ПРИНЦИП ПОСТРОЕНИЯ ПРОГРАММ	92
9.1.	СБОР И ПЕРВИЧНАЯ ОБРАБОТКА ИНФОРМАЦИИ	92
9.2.	ОПРЕДЕЛЕНИЕ НАРУШЕНИЙ НОРМАЛЬНОГО РЕЖИМА ПРОЦЕССА	97
9.3.	ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ СИГНАЛОВ ПРИ РАСЧЕТЕ ТЕКУЩИХ ПОКАЗАТЕЛЕЙ И УПРАВЛЯЮЩИХ ВОЗДЕЙСТВИИ	97
9.4.	ОБ ОБЩЕЙ КОМПОНОВКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ	100
10.	ОПЕРАТИВНОЕ ИЗУЧЕНИЕ СВОЙСТВ ОБЪЕКТА ПО ДАННЫМ НОРМАЛЬНОГО ФУНКЦИОНИРОВАНИЯ	101
10.1.	ПОСТАНОВКА ЗАДАЧИ	102
10.2.	ПОДХОД К РЕШЕНИЮ ЗАДАЧИ В УСЛОВИЯХ ОГРАНИЧЕННОГО ОБЪЕМА ПАМЯТИ	103
10.3.	ОПРЕДЕЛЕНИЕ ХАРАКТЕРИСТИК СИГНАЛОВ	106
10.4.	О ПОЛУЧЕНИИ ОЦЕНОК ВЕСОВОЙ ФУНКЦИИ ОБЪЕКТА ПО ОСНОВНЫМ КАНАЛАМ	109
	ЗАКЛЮЧЕНИЕ	114
	Приложение 1	
	СИСТЕМА КОМАНД ЭВМ «ЭЛЕКТРОНИКА-100»	115
	Приложение 2	
	ПОИСК КОМАНД, ПОРЯЖИХ СОДЕРЖИМОЕ ЯЧЕЙКИ ПО ЗАДАННОМУ АДРЕСУ	123
	Приложение 3	
	МАЛАЯ ОПЕРАЦИОННАЯ СИСТЕМА ДЛЯ ЭВМ «ЭЛЕКТРОНИКА-100»	124
	Приложение 3а	
	ОПЕРАЦИОННАЯ СИСТЕМА ДЛЯ «ЭЛЕКТРОНИКИ-100» С РАСШИРЕННЫМ ОБЪЕМОМ МОЗУ	147
	СПИСОК ЛИТЕРАТУРЫ	184