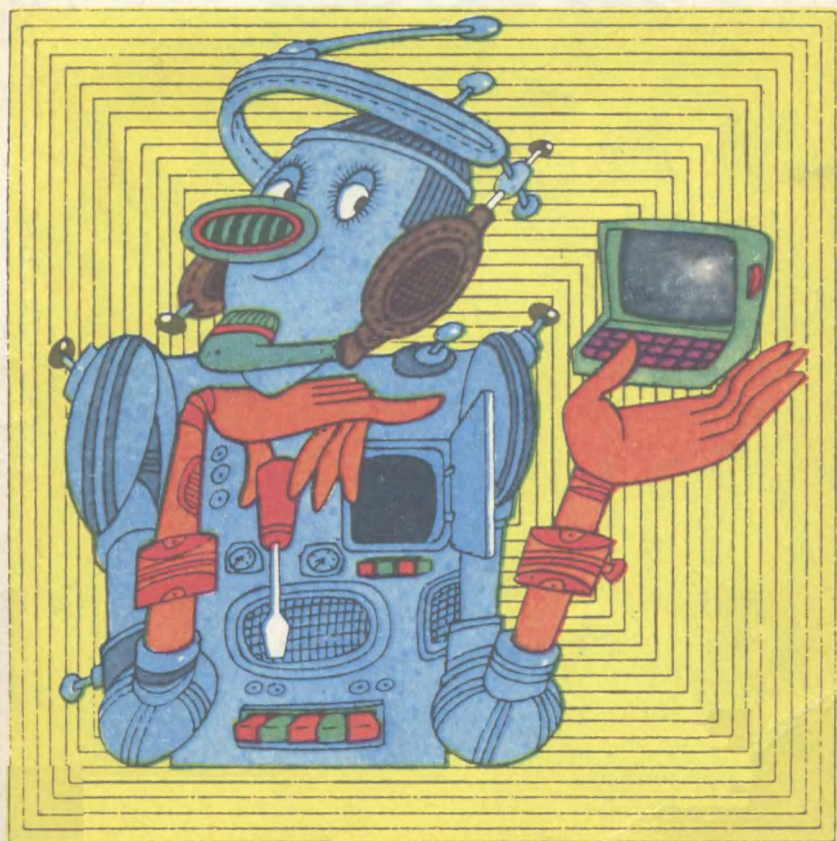


# Мир знаний

А.Н.САЛТОВСКИЙ Ю.А.ПЕРВИН

## Как работает ЭВМ



**МР  
И  
знаний**

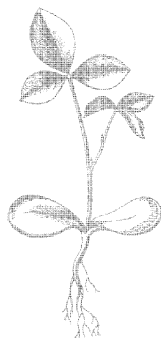
---

**А.Н.САЛТОВСКИЙ Ю.А.ПЕРВИН**

# **Как работает ЭВМ**

*Книга для внеклассного чтения  
учащихся 9—10 классов  
средней школы*

**МОСКВА  
«ПРОСВЕЩЕНИЕ»  
1986**



Scan AAW

ББК 32.973  
С16

Рецензенты:

учитель математики школы № 415 Москвы,  
канд. пед. наук. *Е. Н. Турецкий*;  
преподаватель кафедры математики  
Славянского педагогического института *Е. А. Лодатко*.

**Салтовский А. Н., Первин Ю. А.**

**С16** Как работает ЭВМ: Кн. для внеклас. чтения учащихся 9—10 кл. сред. шк. — М.: Просвещение, 1986. — 159 с.: ил. — (Мир знаний).

Книга является дополнительным пособием для изучения нового предмета «Основы информатики и вычислительной техники». В книге интересно и занимательно раскрывается огромная роль применения ЭВМ в нашей жизни. Книга популярно рассказывает: об архитектуре вычислительных систем, арифметических основах ЭВМ, о перспективах вычислительной техники; знакомит с использованием в курсе программирования средств языков высокого уровня.

С  $\frac{4306020000-704}{103(03)-86}$  КБ—11—19—1986 ББК 32.973

© Издательство «Просвещение», 1986

## ПРЕДИСЛОВИЕ

**Н**ынешние школьники получают достаточно много сведений о программировании на ЭВМ. Они узнают о применениях ЭВМ и способах управления вычислительными машинами не только на школьных уроках, но и на занятиях кружков программирования, в летних школах, в беседах с родителями и шефами.

С 1979 года на страницах журнала «Квант» открылась рубрика «Искусство программирования», где наряду с уроками Заочной школы программирования рассказывалось об устройстве электронных вычислительных машин. На основании материалов «Кванта» и родилась эта книга.

Статьи по устройству ЭВМ не входили в состав обязательных уроков Заочной школы и представляли собою отдельные, по существу, мало связанные друг с другом рассказы. Чтобы из этих рассказов об ЭВМ получилась книга, пришлось написать здесь и об архитектуре вычислительных систем, и об арифметических основах ЭВМ, и о перспективах вычислительной техники. Юным программистам были адресованы и материалы квантовой рубрики «Искусство программирования». Современная методика преподавания этой дисциплины рекомендует в курсе программирования использовать средства языков высокого уровня, которые оставляют в стороне не только конструкцию ЭВМ и отдельных ее устройств, но подчас даже принципы их работы. Так и получается, что школьники, которых интересует вычислительная техника, получают гораздо меньше популярных книг, чем юные программисты. Впрочем, и программистам, по-настоящему увлеченным своей наукой, сведения об устройстве вычис-

лительных машин совершенно необходимы. Тот, кто хочет достичь вершин в программировании, мечтает стать системным программистом. Системный программист должен уметь составлять программы, но в отличие от обычного пользователя ЭВМ он обязан отчетливо представлять, как его программа выполняется на машине. Первое знакомство с миром вычислительных машин не может претендовать на сколь-нибудь подробное описание технических характеристик ЭВМ. На этих страницах излагаются, прежде всего, самые общие черты и свойства элементов, блоков, узлов и устройств машины. В тех случаях, когда общие схематические описания иллюстрируются примерами конкретных ЭВМ, чаще всего упоминаются машины серии ЕС ЭВМ — наиболее распространенные отечественные ЭВМ третьего поколения. Хотя, рассказывая об ЭВМ, трудно не сказать об управляющих ими программах, все же в целом книга построена так, что вопросы программирования остались за ее рамками.

Книга «Как работает ЭВМ» — не учебник. Вместе с тем многие из обсуждаемых в ней вопросов заметно пересекаются с посвященными архитектуре ЭВМ разделами школьного учебника «Основы информатики и вычислительной техники». Поэтому книга может быть использована для дополнительного чтения старшеклассниками, которые заинтересовались этим новым школьным предметом.

В книге довольно много иллюстраций. И хотя далеко не каждый рисунок представляет собой схему какого-либо узла машины, эти рисунки не только иллюстрируют текст, но и непосредственно продолжают его. В этом большая заслуга художника Н. И. Даниловского.

*Вычислительная машина ценна ровно настолько, насколько ценен использующий ее человек.*

**Норберт Винер. Кибернетика**

## **АРХИТЕКТУРА ЭВМ**

**С**овременная *Электронная Вычислительная Машина* — это завод в миниатюре. Завод — это не только комплекс производственных зданий, заполненных станками и разнообразным оборудованием. Завод это еще и технологический процесс — совокупность операций, реализующих обработку материалов, деталей и изделий. Именно технологический процесс приводит к чудесному превращению сырья в готовую продукцию. В нашем удивительном заводе — вычислительной машине — сырьем и полуфабрикатами, деталями и готовой продукцией является *информация*, т. е. записанные в символьной форме сведения об объектах, процессах и явлениях окружающего мира. Числа, тексты, коды, графические изображения — это основные формы представления человеческих знаний.

Электронные вычислительные машины (ЭВМ) используются людьми для того, чтобы оперировать с информацией. Машина способна *прочитать* или *ввести* информацию. Этим занимаются различные устройства ввода информации в ЭВМ. Они преобразуют информацию, записанную в той или иной форме на *информационном носителе* (перфокарте, магнитной ленте, экране электронно-лучевой трубки), в форму, непосредственно воспринимаемую машиной. Другая важная функция ЭВМ — *хранение* информации. Эту функцию выполняет в машине *Память*, или *Запоминающее Устройство* (ЗУ). Несомненно, главной задачей вычислительной машины является *обработка* информации. Эта задача поручена *Процессору*. Можно сказать, что на мини-заводе по переработке информации Процессор — это основной цех. Впрочем, возможности самих машин по обработке информации

весьма ограничены. ЭВМ умеет выполнять небольшое количество элементарных операций — машинных *команд*. Эти команды осуществляют над элементами информации отдельные простые действия, такие, как арифметические операции с числами, операции простейшего редактирования текстов, пересылки из одного места памяти в другое и т. п. Однако из последовательности этих элементарных команд можно составить *программу* — задание вычислительной машине, описывающее сложные процессы обработки информации понятным для ЭВМ способом. Перечень функций вычислительной машины будет неполным, если не сказать о ее умении преобразовывать информацию в форму, воспринимаемую людьми, т. е. *выводить* информацию из ЭВМ. Эту задачу решают разнообразные *выводные* устройства: алфавитно-цифровые печатающие устройства, графо- и фотопостроители, синтезаторы речи и т. д.

*Устройства Ввода-Вывода* (УВВ) информации обеспечивают общение человека с ЭВМ. С помощью УВВ человек вводит в ЭВМ программы и обрабатываемые этими программами данные, а также получает из машины результаты в форме, удобной для дальнейшего использования, — в виде таблиц, графиков, схем. С помощью УВВ организованы и некоторые внутренние процессы обмена информацией в самой ЭВМ («внутризаводские перевозки»).

Основные устройства машины, в частности Запоминающие Устройства и Устройства Ввода-Вывода, в известной степени автономны. Поэтому, помимо основных обязанностей по обработке информации, в функции Процессора входит забота о своих партнерах — УВВ и ЗУ: Процессор координирует работу этих устройств. Одно из замечательных качеств ЭВМ — скорость, с которой она выполняет арифметические и логические операции. Быстродействие Процессоров в современных ЭВМ измеряется десятками миллионов операций в секунду. Для того чтобы обеспечить такую высокую скорость работы машины в целом, все процессы передачи информации между устройствами — обмен данными, прием и распределение очередного задания, запись результатов — полностью автоматизированы. Процессор можно назвать дирижером ансамбля устройств, входящих в состав ЭВМ. Нотной партитурой этому «дирижеру» служит программа, предварительно записанная в ЗУ. Каждый такт пьесы, разыгрываемой под управлением Процессора,



Рис. 1

является законченным элементарным действием: сложение двух чисел, сравнение двух величин и т. д. Логическую организацию вычислительной машины, схематически описывающую взаимоотношения ее основных устройств, называют *архитектурой* ЭВМ. Представленная на рисунке 1 архитектура считается классической. Она предложена



на в 1946 году американским математиком Джоном фон Нейманом. Эта архитектура содержит в себе основные черты современных архитектурных решений вычислительных машин. Именно архитектура ЭВМ отличает современные вычислительные машины от их предшественников — счетно-перфорационных устройств, механических и электрических арифмометров. Важнейший принцип, определяющий архитектуру ЭВМ, состоит в гибком, автоматическом управлении машиной с помощью программы, хранимой в Запоминающем Устройстве машины. Из этого принципа вытекают, в частности, такие важные следствия:

— вычислительная машина является универсальным устройством, которое позволяет решать широкий класс задач и обрабатывать разнообразные данные; переход от решения одной задачи к выполнению другой осуществляется путем смены программ и соответствующих им данных;

— и программы, и данные хранятся в ЗУ; Процессор может обращаться к хранимым в памяти ЭВМ различным объектам, не делая различия между ними (если

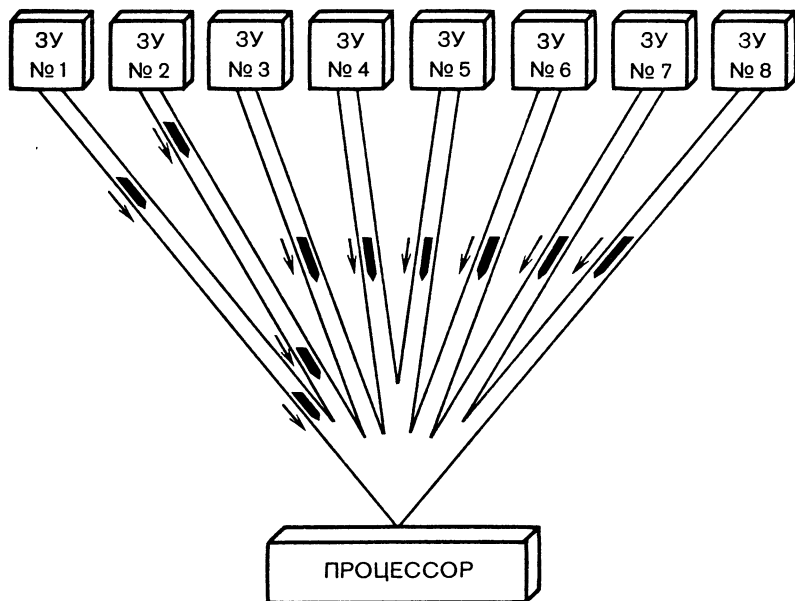


Рис. 2, а.

в программе, которая управляет машиной, запланирован такой режим работы); таким способом Процессор может перерабатывать программу, модифицировать ее. Используя последнее следствие, программисты заставляют машины улучшать, оптимизировать программы.

Архитектура — это не только сложившийся стиль, устоявшаяся логическая организация, это к тому же и

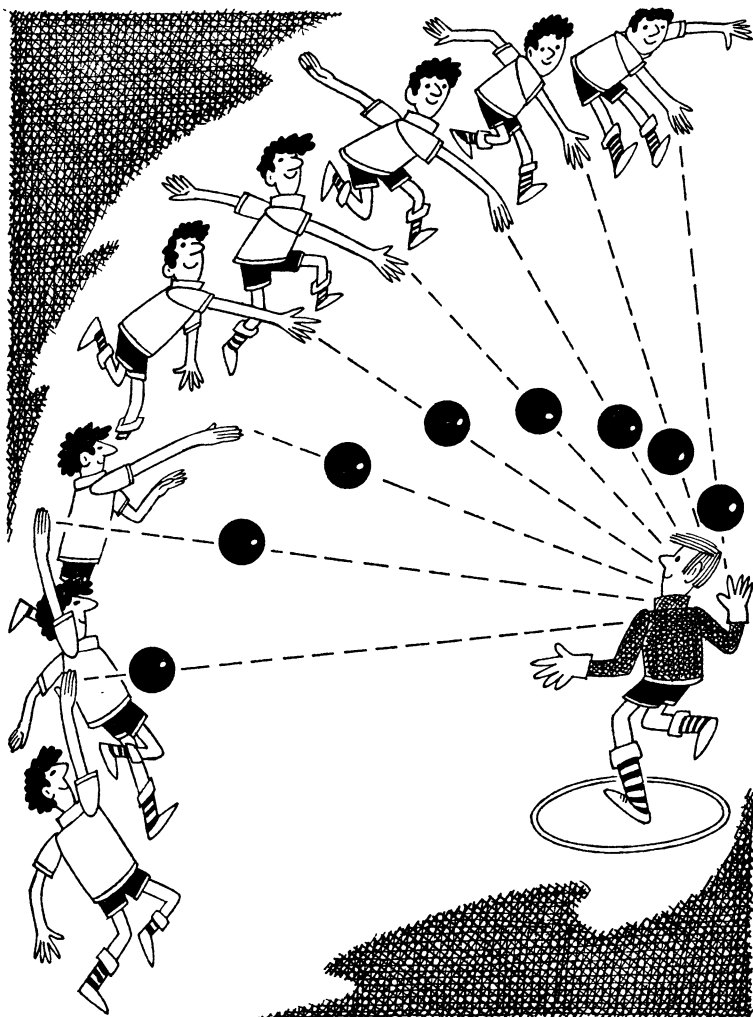


Рис. 2, б.

постоянный поиск новых решений. Основной мотив в модернизации архитектуры типа Неймана — стремление повысить производительность ЭВМ, т. е. в конечном счете сократить время решения задач. Быстродействие всякой системы определяется скоростью работы самого медленного ее элемента. Шаги в преодолении барьера скорости были предприняты в двух направлениях: технологи стремились заставить медленные устройства ЭВМ работать быстрее, в то время как архитекторы искали такие конфигурации устройств ЭВМ, которые уменьшают влияние медленных устройств на быстродействие машины в целом. Одно из архитектурных решений в этом направлении — распараллеливание процессов в ЭВМ. В уже знакомой аналогии «ЭВМ — завод» рассмотрим такую ситуацию: один из станков поточной линии работает в десять раз медленнее остальных. Производительность станка определяется его рабочим циклом — временем выполнения одной операции на этом станке. Если завод не располагает более производительным оборудованием, то вполне естественное решение состоит в том, чтобы поставить в узком месте поточной линии десять станков. Организовать работу этих станков следует так, чтобы операции на каждом из них начинались со сдвигом по времени на  $1/10$  рабочего цикла станка. Тогда временные диспропорции в работе поточной линии будут ликвидированы. В вычислительной машине самыми медленными блоками являются ЗУ и УВВ. Если принять за единицу времени рабочий цикл Процессора — время выполнения одной операции, то около десяти таких циклов занимает рабочий цикл ЗУ (время записи в ЗУ или выборки из ЗУ одной стандартной порции информации) и сотни тысяч — цикл УВВ (время ввода в ЭВМ или вывода из ЭВМ порции информации). Решения, которые могут быть предложены в этом случае, аналогичны рассмотренному примеру с медленным станком. Например, запоминающее устройство ЭВМ БЭСМ-6 состоит из восьми одинаковых блоков. Рабочий цикл каждого из них начинается со сдвигом по времени на  $1/8$  относительно соседнего блока ЗУ (рис. 2, а, б). В современных ЭВМ технологический потолок не достигнут. Тем не менее просматривается он достаточно отчетливо. В машинах, разрабатываемых уже сегодня, начинают сказываться ограничения, связанные со скоростью распространения света. Отрезок длиной 30 см свет проходит за 1 наносекунду ( $10^{-9}$  с). Это время соизмеримо с рабочим циклом процессоров ближайшего

будущего. А вот возможности архитектурных решений неисчерпаемы. Более того, с каждым новым поколением ЭВМ эти возможности постоянно расширяются.

Элементы архитектуры ЭВМ — это электронные и электромеханические устройства, составляющие вычислительную машину. Знакомству с некоторыми устройствами ЭВМ посвящены главы этой книги.

## ДВОИЧНАЯ АРИФМЕТИКА

**П**реобразование информации из одной формы в другую осуществляется с помощью *кодирования*. Наши знания об окружающем мире в известной степени определяются способностью воспринимать и анализировать информацию, т. е. умением правильно истолковывать *код*.

В большинстве реальных задач механизм кодирования может оказаться значительно сложнее того, который разгадал Шерлок Холмс в «Пляшущих человечках». Например, ученые-генетики считают, что в хромосомах клетки эмбриона закодирован полный «план-чертеж» будущего организма. Конструкторы первых ЭВМ должны были решить, как лучше всего кодировать хранимую и обрабатываемую информацию, чтобы обеспечить высокую надежность и простоту, наибольшее быстродействие (производительность), минимальные затраты (экономичность).

Наиболее полно всем этим требованиям удовлетворяет система *двоичного кодирования*, в которой информация представляется последовательностями только двух символов — цифр 0 и 1. В привычной десятичной системе счисления используются десять цифр — 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Когда необходимо изобразить число, большее девяти, используется позиционный способ записи числа; значение, определяемое той или иной цифрой, зависит от ее положения в последовательности цифр числа. Например, десятичное число 5279 представляет собой сумму, слагаемые которой являются степенями числа 10, т. е.

5279	
_____	единицы $9 \cdot 10^0$
_____	десятки $7 \cdot 10^1$
_____	сотни $2 \cdot 10^2$
_____	тысячи $5 \cdot 10^3$

Таким образом, 5279 — сокращенная запись суммы

$$5 \cdot 10^3 + 2 \cdot 10^2 + 7 \cdot 10^1 + 9 \cdot 10^0 = 5279.$$

Позиционный принцип используется и при записи двоичных чисел. В этом случае коэффициентами при степенях числа 2 будут двоичные цифры 0 и 1. Число 5279 в двоичной системе счисления изображается следующим образом:

$$1 \cdot 2^{12} + 0 \cdot 2^{11} + 1 \cdot 2^{10} + 0 \cdot 2^9 + 0 \cdot 2^8 + 1 \cdot 2^7 + 0 \cdot 2^6 + \\ + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 5279,$$

или в сокращенном виде:

$$1010010011111_2 = 5279_{10}$$

(индексы, записанные рядом с изображением числа, указывают систему счисления). Двоичное представление первых шестнадцати чисел показано следующей таблицей:

<i>Десятичное число</i>	<i>Двоичное число</i>
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Способ перевода чисел из десятичной системы счисления в двоичную, с помощью которого можно было бы

не только продолжить эту таблицу, но и преобразовать любое десятичное число  $N$  в двоичное, несложен:

- шаг 1: разделить делимое (в самом начале это число  $N$ ) пополам; зафиксировать остаток (0 или 1) и частное;
- шаг 2: сравнить частное с нулем: если частное не равно нулю, то продолжить действия — вернуться к шагу 1, предварительно отправив частное на место делимого; если частное равно нулю, то перейти к шагу 3;
- шаг 3: зафиксированные в процессе выполнения предыдущих шагов остатки записать в обратном порядке в виде двоичного числа.

Полученная таким образом последовательность нулей и единиц дает представление десятичного числа  $N$  в системе счисления с основанием 2.

$$\begin{array}{r}
 5279 \overline{) 2} \\
 \underline{5278} \phantom{0} 2639 \overline{) 2} \\
 \phantom{0} 1 \phantom{0} 2638 \phantom{0} 1319 \overline{) 2} \\
 \phantom{0} \phantom{0} 1 \phantom{0} 1318 \phantom{0} 659 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} 1 \phantom{0} 658 \phantom{0} 329 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} 1 \phantom{0} 328 \phantom{0} 164 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} 1 \phantom{0} 164 \phantom{0} 82 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} 0 \phantom{0} 82 \phantom{0} 41 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} 0 \phantom{0} 40 \phantom{0} 20 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} 1 \phantom{0} 20 \phantom{0} 10 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} 0 \phantom{0} 10 \phantom{0} 5 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} 0 \phantom{0} 4 \phantom{0} 2 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} 1 \phantom{0} 2 \phantom{0} 1 \overline{) 2} \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} 0 \phantom{0} 0 \phantom{0} 0 \\
 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \phantom{0} 1
 \end{array}$$

Итак,  $5279_{10} = 1010010011111_2$ . Обратное преобразование можно выполнить суммированием степеней двойки, соответствующих ненулевым позициям в изображении двоичного числа. Описанное выше правило перевода десятичного числа в двоичное, т. е. точное, формальное описание выполняемого шаг за шагом процесса, который завершается через конечное число шагов и приводит к решению задачи, представляет собой пример *алгоритма*. В некотором смысле алгоритмами являются кулинарные рецепты, инструкции для детского конструктора, правила дорожного движения и т. д. В зависимости от того, что необходимо сделать — сварить борщ, смастерить скворечник или проявить фотопленку, выбирается определенная последовательность правил для обработки вполне конкретных материалов. Отклонения и неточности в исполнении предписанных правил приводят к тому, что конечная цель не достигается либо искажается, хотя и не всегда таким веселым образом, как у Рассеянного с улицы Бассейной. Говоря более строго, алгоритмом называют конечную систему правил, выполнение которых приводит за конечное число шагов к решению задачи. Перевод из одной системы счисления в другую прост, но тем не менее эту работу неразумно возлагать на человека, так как переводить из системы в систему приходится огромное количество числовой информации. Этим переводам «обучили» саму вычислительную машину. Поступающие в ЭВМ десятичные числа Процессор преобразует в их двоичные эквиваленты: в наборе выполняемых Процессором операций есть и такая — перевести в двоичную систему счисления. Таким образом происходит перевод исходных данных задачи на язык, понятный машине. После того как ЭВМ выполнит обработку информации в соответствии с требованиями программы, потребуется сделать обратное преобразование: полученные в двоичной форме результаты опять же с помощью Процессора будут переведены в десятичную систему счисления. Опираясь с двоичными эквивалентами десятичных чисел, Процессор руководствуется правилами двоичной арифметики. Для сложения эти правила определяются так:

$$\begin{array}{ll} 0 + 0 = 0 & 1 + 0 = 1 \\ 0 + 1 = 1 & 1 + 1 = 0 \text{ (и перенос } + 1 \text{ в следующий старший разряд).} \end{array}$$

Например, сложение двух двоичных чисел 0101 и 0001 дает

$$\begin{array}{r} + 0101 \\ 0001 \\ \hline 0110 \text{ перенос } + 1 \end{array}$$

Выполняя описанным образом операцию сложения, Процессор не умеет непосредственно осуществить операцию вычитания. Поэтому вычитание приходится сводить к сложению путем представления вычитаемого в так называемом *дополнительном* коде.

Рассмотрим прежде всего *обратный* код числа, который называют также *инверсией*. Обратный код числа получается обращением (инвертированием) нулей двоичного числа в единицы, а единиц — в нули. Дополнительный код числа — это сумма его обратного кода и единицы младшего разряда. Например:

$$\begin{array}{r} 1001 \text{ исходное число, или прямой код числа} \\ 0110 \text{ обратный код} \\ + 1 \\ \hline 0111 \text{ дополнительный код} \end{array}$$

Вычитание в двоичной арифметике — это сложение уменьшаемого с дополнительным кодом вычитаемого. Например, для того чтобы из величины 101 вычесть величину 10, необходимо, прежде всего, сформировать обратный код числа 010, т. е. 101, затем увеличить его на единицу, т. е. получить дополнительный код — 110, и, наконец, сложить:

$$\begin{array}{r} + 101 \\ 110 \\ \hline 1011 \end{array}$$

Отметим, что перенос из старшего разряда результата означает, что полученный результат положителен:

$$5 - 2 = 3.$$

Таблица двоичного умножения выглядит более лаконично по сравнению с десятичной:



$$\begin{array}{ll} 1 \times 1 = 1 & 0 \times 1 = 0 \\ 1 \times 0 = 0 & 0 \times 0 = 0, \end{array}$$

а правила деления не отличаются от правил деления в десятичной системе:

$$\begin{array}{l} 0/1 = 0 \\ 1/1 = 1 \end{array}$$

Деление на нуль не определено.

Несложные правила двоичной арифметики обеспечивают простоту тех узлов Процессора, которые выполняют арифметические операции. Удобная для вычислительной машины двоичная система счисления не употребляется людьми из-за того, что большие числа в этой системе представляются уж очень длинными (и уныло однообразными) последовательностями цифр. Было найдено компромиссное решение — переводить в двоичную систему не все число целиком, а каждую его цифру отдельно:

5	2	7	9
0101	0010	0111	1001

Для изображения одной десятичной цифры в этом случае используются четыре двоичные (одна тетрада). Такой способ кодирования называется *двоично-десятичным*. Двоично-десятичная система не единственная из применяемых в ЭВМ вспомогательных систем счисления. Достаточно широкое распространение получила *шестнадцатичная* система счисления, которая позволяет получить более компактную запись числа (иными словами, увеличить информационную емкость одной тетрады). Десяти арабских цифр для шестнадцатичной системы недостаточно, и для изображения шести старших цифр в этой системе используют шесть начальных букв латинского алфавита:

$1010_2 = 10_{10} = A_{16}$	$1101_2 = 13_{10} = D_{16}$
$1011_2 = 11_{10} = B_{16}$	$1110_2 = 14_{10} = E_{16}$
$1100_2 = 12_{10} = C_{16}$	$1111_2 = 15_{10} = F_{16}$

Вот как десятичное число 5279 записывается в системе счисления с основанием 16:

$$5279_{10} = 1 \cdot 16^3 + 4 \cdot 16^2 + 9 \cdot 16^1 + 15 \cdot 16^0 = 149F_{16}.$$

Чтобы преобразовать, например, двоичное число  $1010010011111_2$  в шестнадцатиричную систему, двоичные разряды группируются по четыре слева направо и при необходимости старшую группу можно дополнить нулями:

$$0001010010011111_2.$$

Преобразование завершается заменой каждой группы эквивалентной шестнадцатиричной цифрой:

$$0001010010011111_2 = 149F_{16}.$$

Обратная операция переводит числа из шестнадцатиричной системы в двоичную. Например:

$$8E1C_{16} = 1000111000011100_2.$$

С помощью цепочек двоичных чисел можно закодировать не только десятичные числа, но также буквы русского и латинского алфавитов, знаки операций, специальные символы. Например, машины серии ЕС ЭВМ используют в своей работе *Двоичный Код Обмена Информацией* (ДКОИ), где каждому символу — цифре, букве, знаку — соответствует определенная комбинация из восьми двоичных цифр. Слово «ЭВМ», в частности, можно закодировать так:

Э	В	М
11111100	11000010	11010100

Двоичное представление алфавитно-цифровой информации позволяет вычислительной машине выводить в качестве результатов не только числовые, но и текстовые значения, снабжая решения удобочитаемыми текстовыми комментариями.

## АЛГЕБРА БУЛЯ

**С**писок доводов в пользу выбора двоичной системы кодирования при проектировании ЭВМ следует дополнить возможностью применения двоичной логики, которую называют иногда булевой алгеброй по имени английского математика Джоржа Буля, сформулировавшего

в XIX веке основные положения этого раздела математической логики.

Возможность математически строго описать работу машины (необязательно ЭВМ), а затем на основании такого описания создать улучшенный вариант с наперед заданными свойствами всегда привлекала конструкторов. При таких формальных описаниях обычно используют ряд математических дисциплин. Для создателей ЭВМ неоценимую помощь в анализе работы машины и в синтезе ее узлов оказала булева алгебра. Начальным понятием булевой алгебры является *высказывание*. Под высказыванием понимается любое утверждение, оцениваемое только с точки зрения его истинности. Качественные характеристики высказывания — справедливое, хорошее, содержательное, грубое — не рассматриваются. Высказывания с точки зрения булевой алгебры могут быть истинными или ложными. Например, из двух высказываний

$X$  = «Алмаз имеет кристаллическую структуру» и

$Y$  = «Волга впадает в Балтийское море»

первое истинно, а второе ложно.

В булевой алгебре высказывания могут обозначаться буквами, подобно переменным в обычной школьной алгебре. Более того, высказывания, по существу, и являются переменными булевой алгебры, принимающими значение 1 в случае истинности высказывания и 0, если высказывание ложно. Такие переменные называют *логическими* (или булевыми) переменными. Для двух высказываний приведенного примера возможна, следовательно, и такая запись:

$$X = 1; Y = 0.$$

Высказывания могут быть простыми и сложными. Высказывание называется *простым*, если его значение не зависит от значений каких-либо других высказываний. Например, в высказывание «Мюнхгаузен летал верхом на ядре» не проникли мнения соотечественников и современников знаменитого барона, и высказывание оказалось простым. *Сложным* считается высказывание, значение истинности которого определяется значениями других высказываний. Известное высказывание «Хорошо живет на свете Винни-Пух, если, конечно, он вовремя подкрепится» является сложным высказыванием: благополучное содержание первой части фразы (первого высказы-

вания) зависит от некоторого условия, составляющего вторую половину предложения (второго высказывания). Всякое сложное высказывание является логической (булевой) *функцией* некоторых двоичных аргументов — простых высказываний. Рассмотрим простейшие логические функции. С их помощью узлы и блоки ЭВМ строятся подобно тому, как из простых высказываний можно сконструировать сложное. Существуют три функции, которые составляют основной набор: доказано, что любая логическая функция может быть построена путем использования этих трех функций. «Здание» логической функции строится из «кирпичей» основного набора. Первая из этих функций — отрицание. Она обозначается словом НЕ:  $Y = \text{НЕ} (X)$ . Отрицание означает такую логическую связь между аргументом  $X$  и функцией  $Y$ , при которой  $Y$  истинно только тогда, когда ложен аргумент  $X$ , и наоборот. Так, например, отрицание высказывания «Новосибирск расположен на Оби», истинность которого равна 1, означает высказывание «Неверно, что Новосибирск расположен на Оби» (или «Новосибирск не расположен на Оби»), истинность которого равна 0. Таблица соответствия возможных значений аргумента (входных двоичных переменных) значениям функции, называемая в случае булевых функций таблицей истинности, имеет следующий вид для функции отрицания НЕ:

входная двоичная переменная (аргумент)	0	1
выходная двоичная переменная (функция)	1	0

Функция отрицания имеет еще и такое обозначение:  $Y = \bar{X}$ ; иногда эту же функцию записывают так:  $Y = \neg X$ . Вот некоторые свойства этой функции:

- двойное отрицание некоторого аргумента  $X$  равно самому аргументу, т. е.  $Y = \bar{\bar{X}} = X$ ;
- если имеется некоторое логическое равенство, то отрицание обеих его частей не нарушает этого равенства, т. е. если  $X_1 = X_2$ , то  $\bar{X}_1 = \bar{X}_2$ .

Второй функцией основного набора является *конъюнкция*  $И(X_1, X_2)$ . Ее называют иногда логическим умножением. Конъюнкцией высказываний называют такое сложное высказывание  $Y$ , которое истинно только тогда, когда истинны все входящие в него простые высказывания; во всех остальных случаях  $Y$  ложно. Примером конъюнкции может быть такое сложное высказывание, как «Летние каникулы друзья провели в пионерском

лагере, и 1 сентября они отправляются в школу». Истинность этого сложного высказывания определяется значениями истинности двух простых высказываний, независимых друг от друга (рис. 3). Таблица истинности конъюнкции двух аргументов имеет следующий вид:

первый логический аргумент $X_1$	0	0	1	1
второй логический аргумент $X_2$	0	1	0	1
логическая функция $Y$	0	0	0	1

При записи конъюнкции применяют и такие обозначения:

$X_1 \& X_2$  или  $X_1 \wedge X_2$ .

Третьим «кирпичиком» основного набора булевых функций служит *дизъюнкция* или  $(X_1 \vee X_2)$ , называемая иногда логическим сложением. Дизъюнкцией высказываний называется сложное высказывание  $Y$ , которое ложно тогда, когда ложны все входящие в него высказывания; в остальных случаях  $Y$  истинно (рис. 4). Высказывание «Физкультурник поднимает двухпудовую гирию или участник соревнований подтягивается на перекладине» определяет условие, при котором зачитывается одна из норм физкультурного комплекса ГТО: если ложно каждое из простых высказываний, составляющих эту дизъюнкцию, то участник не получает зачет.

Таблица истинности дизъюнкции двух аргументов имеет такой вид:

первый логический аргумент $X_1$	0	0	1	1
второй логический аргумент $X_2$	0	1	0	1
логическая функция	0	1	1	1

Другое обозначение дизъюнкции:  $y = X_1 \vee X_2$ .

При доказательстве логических теорем полезным инструментом служат диаграммы Эйлера-Венна (рис. 5, а, б, в). Если заштрихованный круг (рис. 5, а) представляет область истинности значений переменной  $X$ , то все, что находится за пределами этого круга, будет представлять НЕ  $X$  (или, в других обозначениях,  $\bar{X}$ ). Теперь если рассмотреть два круга, которые изображают об-

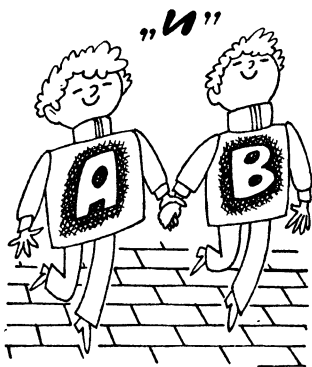


Рис. 3

ласти истинности переменных  $X_1$  и  $X_2$ , то область, в которой эти круги пересекаются, содержит те значения  $X_1$  и  $X_2$ , для которых истинна конъюнкция  $X_1$  и  $X_2$ , или  $X_1 \& X_2$  (рис. 5, б). Область, в которой истинна по крайней мере одна из переменных  $X_1$  или  $X_2$  (рис. 5, в), является областью дизъюнкции  $X_1$  или  $X_2$  ( $X_1 \vee X_2$ ).

Построение более сложных высказываний с помощью основного набора булевых функций можно продемонстрировать на таком примере: «Я буду читать, если есть хорошая книга и есть свободное время или если я ищу ответ на интересующий меня вопрос и надеюсь найти его в этой книге». Сложная функция, определяющая условие, при котором я буду читать, записывается с помощью логического выражения

$$\Phi(X_1, X_2, X_3, X_4) = (X_1 \& X_2) \vee (X_3 \& X_4),$$

где  $X_1$  — «есть хорошая книга»;  $X_2$  — «есть свободное время»;  $X_3$  — «ищу ответ на вопрос»;  $X_4$  — «надеюсь найти ответ».

Интересные соотношения между логическими функциями И, ИЛИ, НЕ описываются теоремами де Моргана:

$$\overline{X_1 \& X_2} = \overline{X_1} \vee \overline{X_2};$$

$$\overline{X_1 \vee X_2} = \overline{X_1} \& \overline{X_2}.$$

Эти формулы легко получить, рассматривая диаграммы Эйлера-Венна. Так как заштрихованная область (рис. 5, в, б) соответствует конъюнкции  $X_1 \& X_2$ , то не-

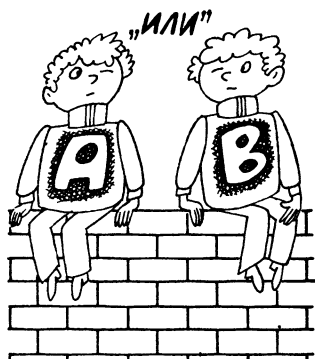
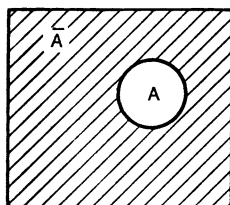
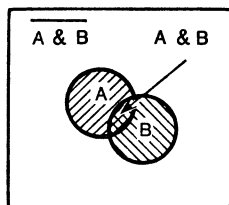


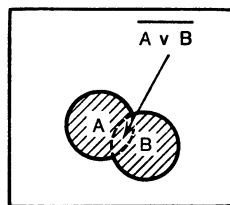
Рис. 4



а)



б)



в)

Рис. 5

заштрихованная часть плоскости —  $\overline{X1} \& \overline{X2}$ . Однако незаштрихованную область можно представить и по-другому, учитывая, что она включает одну из областей —  $\overline{X1}$  или  $\overline{X2}$ , т. е.  $\overline{X1} \vee \overline{X2}$ . Приравнивая оба выражения для незаштрихованной области, получим соотношение  $\overline{X1 \& X2} = \overline{X1} \vee \overline{X2}$ . Наиболее важные теоремы булевой алгебры перечислены в следующей таблице:

1а	$\overline{0} = 1$	16	$\overline{1} = 0$
2а	$X \vee 0 = X$	26	$X \& 1 = X$
3а	$X \vee 1 = 1$	36	$X \& 0 = 0$
4а	$X \vee X = X$	46	$X \& X = X$
5а	$X \vee \overline{X} = 1$	56	$X \& \overline{X} = 0$
6а	$(X) = \overline{\overline{X}}$		
7а	$X1 \vee X2 = X2 \vee X1$	76	$X1 \& X2 = X2 \& X1$
8а	$X1 \vee X1 \& X2 = X1$	86	$X1 \& (X1 \vee X2) = X1$
9а	$X1 \vee \overline{X1} \& X2 =$ $= X1 \vee X2$	96	$X1 \& (\overline{X1} \vee X2) =$ $= X1 \& X2$
10а	$(X1 \vee X2) \vee X3 =$ $= X1 \vee (X2 \vee X3) =$ $= X1 \vee X2 \vee X3$	106	$X1 \& (X2 \& X3) =$ $= (X1 \& X2) \& X3 =$ $= X1 \& X2 \& X3$
11а	$X1 \vee X2 \& X3 =$ $= (X1 \vee X2) \&$ $\& (X1 \vee X3)$	116	$X1 \& (X2 \vee X3) =$ $= X1 \& X2 \vee X1 \& X3.$

Приведенные соотношения дают правила преобразования булевых выражений. С их помощью можно получать эквивалентные выражения. Новые выражения могут оказаться проще. А это приводит, в частности, к экономии оборудования и повышению быстродействия функциональных узлов ЭВМ. Например, выражение

$$(X1 \vee X2) \& (X1 \vee \overline{X2}) \vee X3$$

можно упростить следующим образом:

$(X1 \vee X2) \& (X1 \vee \overline{X2}) \vee X3 = (X1 \vee X2 \& X2) \vee X3$  по теореме 11а  
 $= X1 \vee 0 \vee X3$  по теореме 56  
 $= X1 \vee X3$  по теореме 2а.

Достаточно просто построить таблицу истинности по некоторому наперед заданному булевому выражению. Для этого в выражение подставляют вместо переменных их значения и, пользуясь определениями операций И, ИЛИ, НЕ, вычисляют значение выражения. Чтобы пока-

зять, как получается таблица истинности по булевому выражению, обратимся к примеру. Допустим, необходимо построить таблицу истинности для выражения

$$\Phi(X1, X2) = (\overline{X1} \& X2) \vee X1 \& \overline{X2}.$$

Поскольку это функция от двух переменных, в таблице истинности должно быть  $2^2 = 4$  строки.

Таблица истинности (табл. 1) для булева выражения  $\Phi(X1, X2) = (\overline{X1} \& X2) \vee (X1 \& \overline{X2})$ :

Т а б л и ц а 1

X1	X2	X1	X2	X1 X2	X1 X2	Φ
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	0	0	0

Построение таблицы истинности завершают вычислением всего выражения для каждой из четырех комбинаций. Например, при  $X1 = 0, X2 = 1$  получим:

$$\Phi(0, 1) = (\overline{0} \& 1) \vee (0 \& \overline{1}) = (1 \& 1) \vee (0 \& 0) = 1 \vee 0 = 1.$$

## СИГНАЛЫ, ПОМЕХИ И ГОНЦЫ

**И**так, главным «цехом» по переработке информации в ЭВМ является Процессор, «складом» хранения информации на всех стадиях обработки — ЗУ, преобразователями информации в диалоге «человек — ЭВМ» — различные УВВ. Не менее важную роль играют незаметные и незаменимые помощники — *каналы связи*. Процесс обработки информации предполагает наличие источника информации (передатчика) и приемника. Способы, с помощью которых сообщение доставляется от источника к приемнику, могут быть самыми разнообразными. В старину, например, функции канала связи выполняли гонцы-скалоходы. Это один из примеров использования механического движения для передачи сообщений.

Передача сообщений происходит во времени. Поэтому канал связи должен обладать способностью перемещать сообщение в протяженной среде (механическое движение) либо изменять во времени какие-нибудь физические характеристики пространства (давление в жидкостях и



газах, электрические напряжения и токи, электромагнитные волны). Изменение некоторой физической величины, обеспечивающее передачу информационного сообщения, называют *сигналом* (рис. 6). Носителем информации в ЭВМ является электрический сигнал, регистрируемый приемниками информации как изменение тока или напряжения. В задачу источника информации входит создание (генерация) этого изменения. Если электрический сигнал непрерывен во времени, его называют аналоговым (рис. 7). Этот сигнал переносит информацию в виде непрерывного изменения напряжения или тока.

Напротив, сигнал, имеющий прерывистую (дискрет-

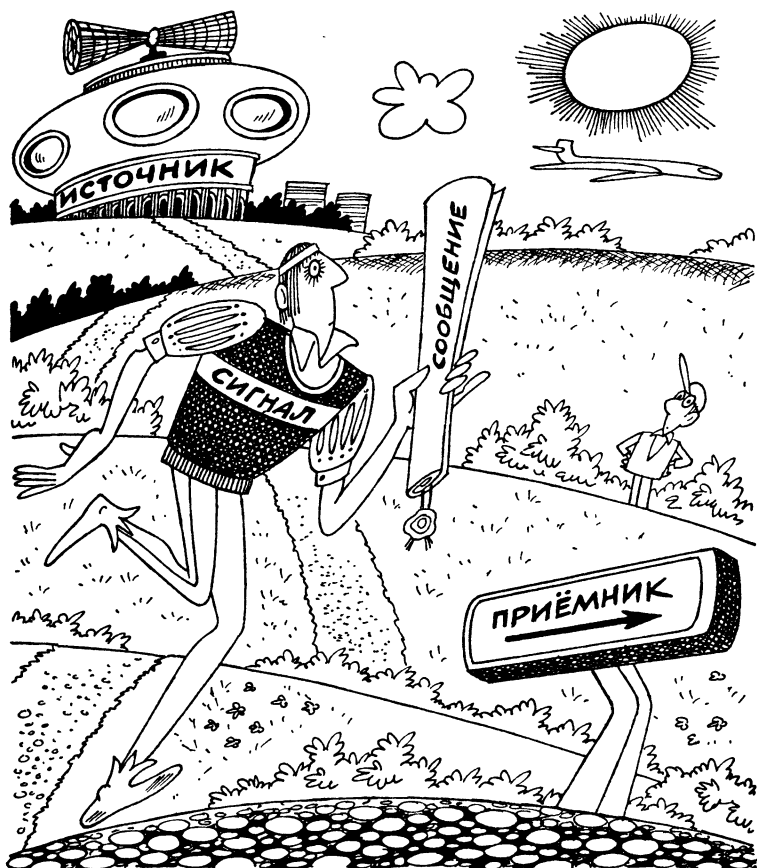


Рис. 6



Рис. 7



Рис. 8

ную) форму, называется *дискретным* или *цифровым*. Цифровые сигналы принимают значения в двух дискретных областях и хорошо приспособлены для передачи информации о процессах, в которых отчетливо выражены два состояния: «включено — выключено», «истина — ложь», «0—1». Только цифровым сигналам предоставлено монопольное право управлять информационными потоками внутри ЭВМ, так как только эти сигналы используются для передачи информации от источников к приемникам. Именно по этой причине разнообразные устройства, входящие в состав ЭВМ, построены в основном из электронных переключателей, похожих на бытовые выключатели «включено — выключено», но отличающихся миниатюрностью и высокой скоростью переключения. Такие элементы дают возможность хранить (например, с помощью магнитных материалов, различающих состояния «намагничено — размагничено») и передавать (например, изменяя напряжение в проводной линии связи) информацию простыми техническими средствами. Цифровые электрические сигналы (рис. 8) обладают высокой устойчивостью к различным помехам. Это важнейший фактор надежности такого сложного комплекса, как ЭВМ. О роковом влиянии помех, искажающих передаваемую по каналу связи информацию, прекрасно рассказано в «Сказке о царе Салтане» А. С. Пушкина: помехи (три кумушки) исказили сигнал (письмо), передаваемый по каналу связи (гонца) от источника информации (царь) к приемникам (бояре и царица). Но если говорить серьезно, то высокая помехоустойчивость цифровых электрических сигналов позволяет современным ЭВМ надежно работать вблизи таких сильных источников помех, как ускорители элементарных частиц, в условиях космического излучения. Вопросы отделения полезного сигнала от помех составляют содержание одного из разделов *кибернетики* — науки об общих законах хранения, передачи и преобразования информации в сложных системах вообще и в ЭВМ в частности. Некоторые сведения об окружающем мире наиболее естественно

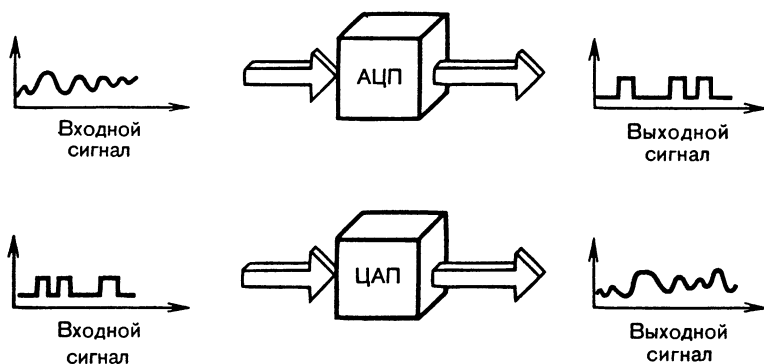


Рис. 9

представляются аналоговыми сигналами: угол поворота вращающегося вала, давление под поршнем двигателя, положение колеблющейся мембраны и т. п. Но и такая непрерывная информация не остается за бортом ЭВМ: существуют технические средства преобразования аналоговых сигналов в цифровые. Такое преобразование выполняется, как правило, непосредственно перед вводом информации в ЭВМ. Возможно и обратное преобразование: цифровые сигналы превращаются в аналоговые. Это происходит при выводе обработанной информации из машины (рис. 9). Такие преобразования сигналов дают возможность моделировать многие физические процессы в вычислительной машине с помощью аналоговых сигналов и обеспечить высокую точность обработки информации, характерную для цифрового представления. Устройства, выполняющие преобразования сигналов, так и называются — *Аналого-Цифровой Преобразователь (АЦП)* и *Цифро-Аналоговый Преобразователь*

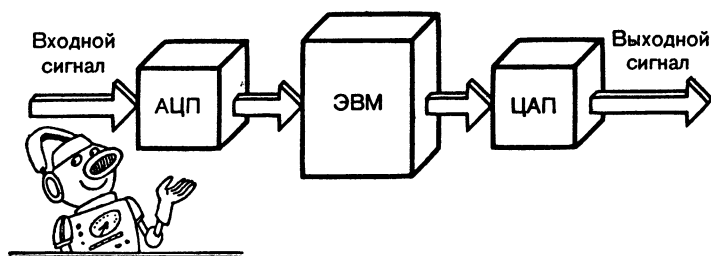


Рис. 10

(ЦАП). Преобразователи сигналов являются частью Устройств Ввода-Вывода (УВВ) — устройств, связывающих ЭВМ с внешним миром, с человеком. АЦП и ЦАП позволяют, например, применять ЭВМ для управления промышленными роботами (рис. 10). Входной аналоговый сигнал в этом случае поступает с потенциометров, датчиков или телекамер. После преобразования и обработки информации в ЭВМ выходной аналоговый сигнал управляет гидравлическими элементами, исполнительными механизмами и двигателями.

## АЗБУКА ЦИФРОВОЙ ЭЛЕКТРОНИКИ

**В**опрос «Из каких элементов строить проектируемое изделие?» является одним из самых принципиальных для каждого конструктора, в том числе и для разработчиков вычислительных машин. Перефразируя С. Лема, вычислительные машины можно назвать «подвижной суммой технологий»: за свою сорокалетнюю историю мир ЭВМ неоднократно обновлял множество физических элементов (деталей, приборов, устройств), из которых организуется логическая структура машины. Большое влияние на развитие ЭВМ оказала полупроводниковая технология — замечательный сплав достижений физики твердого тела, радиоэлектроники, лазерной техники, прецизионной (высокоточной) обработки материалов. Первыми продуктами полупроводниковой технологии, предоставленными создателям ЭВМ, были полупроводниковые диоды. Их основное свойство — проводить электрический ток только в одном направлении — было использовано для реализации некоторых логических функций. Затем появились транзисторы — приборы, дополнившие переключаемые свойства диодов возможностью создания устройств для запоминания информации. С помощью транзисторов и диодов полупроводниковая технология вытеснила из ЭВМ радиолампы и значительно повысила быстродействие машин, уменьшив в то же время размеры и вес аппаратуры.

В 60-е годы были созданы интегральные схемы (ИС). Они в миниатюре воспроизводят сложную и разнообразную структуру обычных радиоэлектронных схем, включающих резисторы, конденсаторы, диоды, транзисторы, межэлементные соединения, цепи электропитания. Интегральная схема — законченное электронное устройство, предназначенное для выполнения определенных функций.

Это прибор, в котором максимальная плотность элементов в минимально возможном объеме сочетается с большими техническими возможностями. В процессе технологической эволюции плотность размещения элементов в ИС и функциональные возможности интегральных схем постепенно увеличивались (при практически неизменных размерах корпуса ИС). В настоящее время интегральные схемы классифицируют по функциональным возможностям так: Интегральные Схемы (ИС), Средние Интегральные Схемы (СИС), Большие Интегральные Схемы (БИС). Если простые ИС предназначены для выполнения несложных логических функций, то некоторые виды БИС

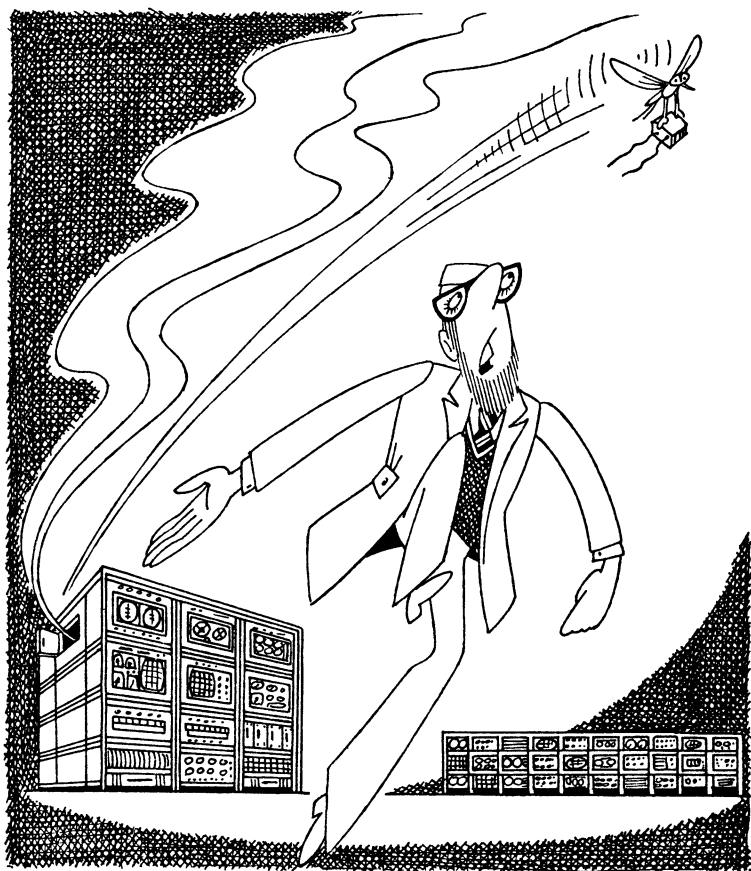


Рис. 11

способны самостоятельно обрабатывать информацию: в 1971 году появились микропроцессоры, реализованные в виде единой БИС; уже созданы БИС — микро-ЭВМ. Желая показать миниатюрность первых карманных радиоприемников, собранных на транзисторах, их фотографировали в рекламных целях рядом с авторучками или спичечными коробками. На современных фотографиях, сделанных с помощью мощных микроскопов, электронные лабиринты БИС соседствуют с вирусом гриппа (рис. 11). Интегральные схемы позволяют реализовать логические функции разнообразной сложности. К тому же корпуса ИС стандартизованы по форме. Это позволяет конструкторам создавать узлы и блоки ЭВМ из наборов ИС примерно с той же методичностью, с какой ребенок собирает разные игрушки из ограниченного набора деталей детского конструктора.

Знакомство с интегральными схемами, предназначенными для выполнения логических функций, начнем с *инвертора*, который реализует логическую функцию отрицания НЕ: работа инвертора полностью описывается таблицей истинности этой функции. На рисунке 12, а схематически изображен инвертор. Такое его условное изображение используется в схемах вычислительных машин. На рисунке 12, б представлена временная диаграмма инвертора — график зависимости сигнала, проходящего через инвертор, от времени. На вход инвертора подается цифровой сигнал, величина напряжения которого соответствует значению аргумента логической функции. Например, для  $X=1$  это напряжение составляет  $+5$  В, а для  $X=0$  —  $0$  В (рис. 12, в). На выходе инвертора получается сигнал, представляющий значение функции отрицания НЕ, т. е. значение, обратное входному (рис. 12, г):  $Y=1$  (на выходе  $+5$  В, если на входе  $0$  В) или  $Y=0$  (на выходе  $0$  В, если на входе  $+5$  В). В стандартный корпус простых ИС ( $19 \times 7.2 \times 3.2$  мм) технологии упаковывают шесть инверторов. Надо сказать, что на временной диаграмме инвертора допущена некоторая условность: судя по диаграмме, переключение цифрового сигнала происходит мгновенно; в действительности же любой физический процесс в переключательных схемах протекает за определенное время. Для сравнения можно сказать, что запаздывание электромеханического реле составляет  $50$  миллисекунд ( $50 \cdot 10^{-3}$  с), а логические элементы современной советской ЭВМ ЕС-1060 переключаются за  $3$  наносекунды ( $3 \cdot 10^{-9}$  с). Другими базовыми

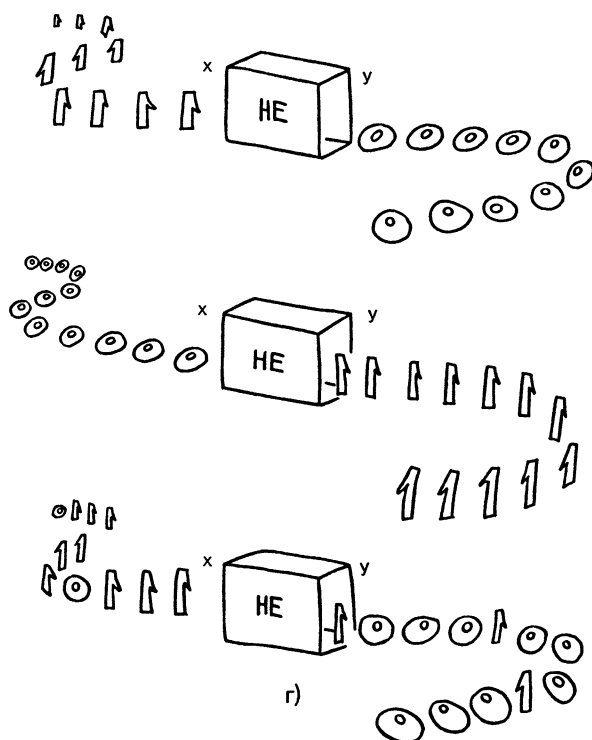
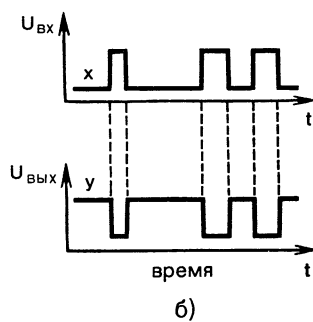
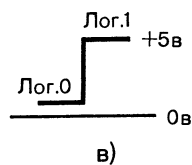
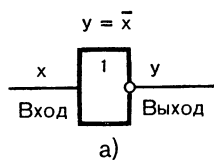


Рис. 12

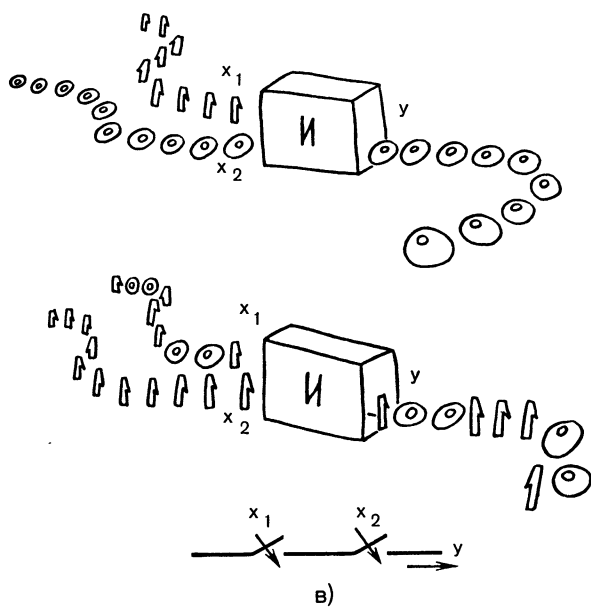
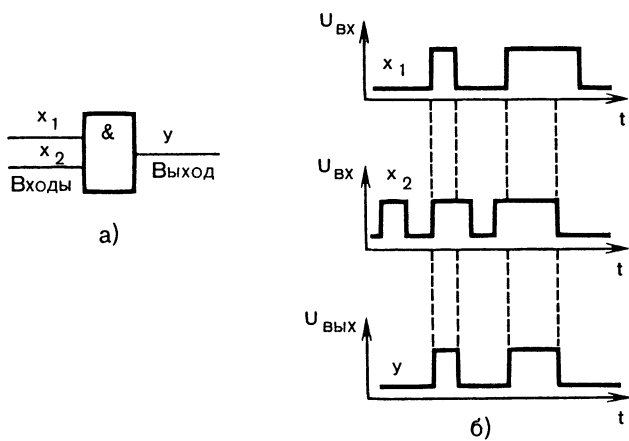
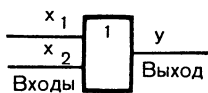
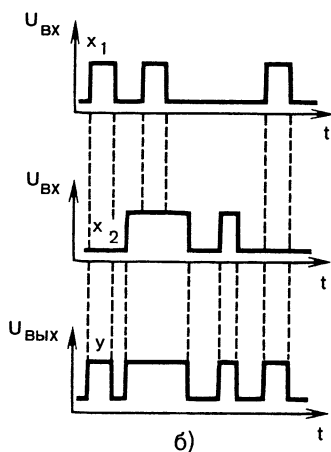


Рис. 13





а)



б)

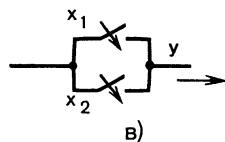
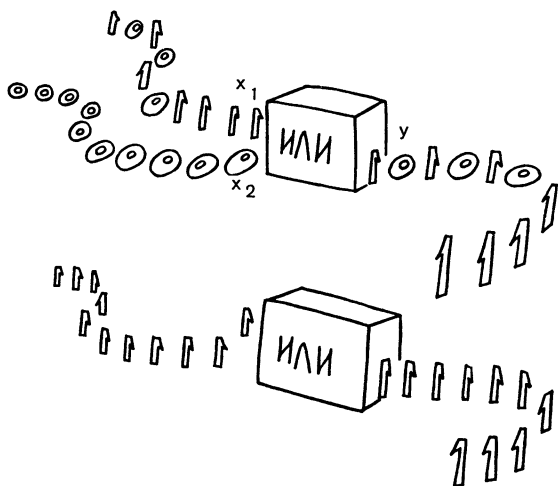


Рис. 14

элементами ИС являются *конъюнктор*, предназначенный для реализации функции логического умножения, и *дизъюнктор*, реализующий функцию логического сложения. ИС, реализующую функцию И, иногда называют *схемой совпадения*, что отражает существо работы этого элемента: цифровой сигнал, соответствующий значению логической единицы, появляется на выходе схемы совпадения тогда, когда совпадут единичные значения цифровых сигналов на входе. На рисунке 13, *а* представлено схематическое изображение конъюнктора, на рисунке 13, *б* — его временная диаграмма, а на рисунке 13, *в* — аналог схемы «И»:  $X = 1$ , если контакт замкнут,  $X = 0$ , если контакт разомкнут,  $Y = 1$ , если в цепи есть ток,  $Y = 0$ , если тока в цепи нет.

ИС, которая реализует функцию ИЛИ, называют *схемой сборки*: сигнал, соответствующий уровню логической единицы, возникает на выходе, если хотя бы на один из входов придет сигнал логической единицы. На рисунке 14, *а*, *б*, *в* приведены соответственно схематическое обозначение дизъюнктора, его временная диаграмма и аналог схемы «ИЛИ». В результате синтеза инвертора и схемы совпадения можно получить схему, реализующую функцию И — НЕ (см. рис. 15, *а*), обозначение ИС «И — НЕ» приведено на рисунке 15, *б*. Схема «И — НЕ» — один из популярных «винтиков» у конструкторов ЭВМ и у технологов, создающих новые БИС. Универсальность этой схемы заключается в том, что из нее легко получить инвертор (рис. 16, *а*), а воспользовавшись правилом

$$\overline{X1 \& X2} = \overline{X1} \vee \overline{X2},$$

можно выразить функцию ИЛИ (рис. 16, *б*).

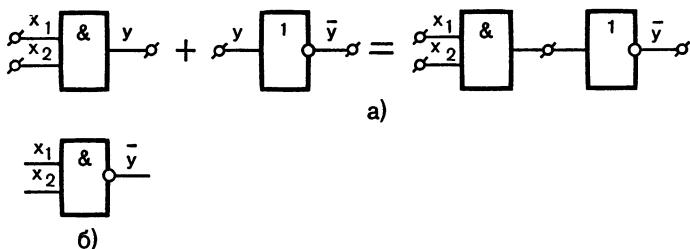


Рис. 15

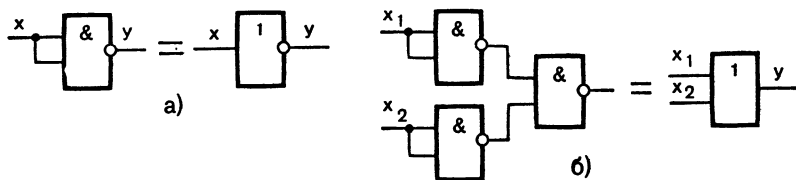


Рис. 16

Из простых схем можно собирать устройства, способные выполнять сложные логические функции. Это возможно прежде всего потому, что основной набор функций НЕ, И, ИЛИ обладает функциональной полнотой: никаких дополнительных элементов при синтезе схем, кроме схем, реализующих функции основного набора, не потребуется. Важно также, что в пределах одной группы (или, как говорят, одной серии) цифровых ИС электрические сигналы, представляющие логический ноль (равно, как и логическую единицу), одинаковы. Это дало возможность конструкторам стыковать различные по

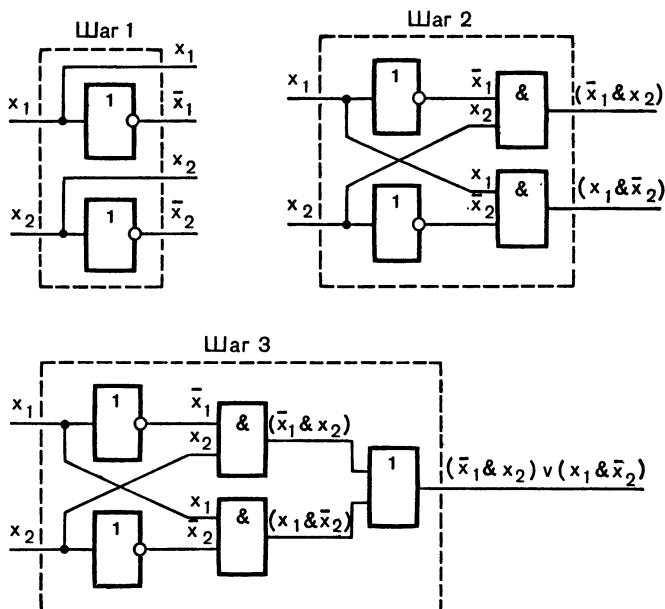


Рис. 17

функциям ИС. Таким образом можно получить разнообразные комбинации логических функций, целевое назначение которых — выполнение алгоритмов, по которым работают различные блоки ЭВМ. В качестве примера рассмотрим построение функции

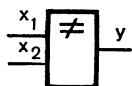


Рис. 18

$$Y = (\overline{X1} \& X2) \vee (X1 \& \overline{X2})$$

из знакомых уже И, ИЛИ, НЕ. Для этого потребуются два инвертора, две схемы совпадения и одна схема сборки (рис. 17):

- шаг 1: левая или правая половины выражений  $\overline{X1} \& X2$  и  $X1 \& \overline{X2}$  содержат прямой и обратный код значений аргументов  $X1$  и  $X2$ ; это преобразование выполняют два инвертора;
- шаг 2: левую и правую конъюнкции выполняют две схемы совпадения;
- шаг 3: полученные на предыдущем шаге конъюнкции собирает схема сборки.

Построенная схема имеет широкое практическое применение: единичное значение сигнала  $Y$  появляется на входе только тогда, когда значения сигналов на входах будут не равны друг другу. Схема имеет собственное условное обозначение (рис. 18) и собственное название — исключающее ИЛИ (или схема сравнения). ИС, выполняющую функцию исключающего ИЛИ, часто используют для поразрядного сравнения двоичных чисел. Эта операция очень важна в системе контроля информации внутри ЭВМ: эталон сравнивается с информацией, передаваемой по транспортным конвейерам машины; если происходит искажение сообщения, то схемы сравнения обнаруживают это и вырабатывают сигналы тревоги,

$X1$	$X2$	$Y$
0	0	0
1	0	1
0	1	1
1	1	0

Рис. 19

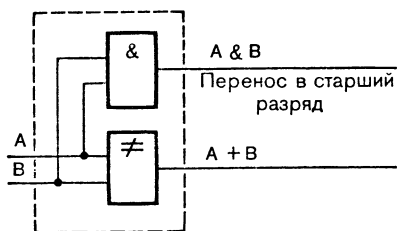


Рис. 20

призывая инженеров и техников, обслуживающих ЭВМ, заняться проверкой оборудования. ЭВМ, как всякий завод, имеет строгий отдел технического контроля. Впрочем, контроль не единственная специальность схемы сравнения. Если внимательно посмотреть на таблицу ее работы (таблица истинности, см. рис. 19), то можно заметить ее сходство с таблицей сложения двоичных чисел. Правда, перенос в старший разряд в этом случае теряется и нельзя принять перенос, поступающий со стороны младших разрядов.

Для формирования переноса в старший разряд схему сравнения (исключающего ИЛИ) соединяют со схемой совпадения (И). Последняя вырабатывает логическую единицу (в этом случае именно единицу переноса) тогда, когда значения аргументов  $A$  и  $B$  одновременно принимают значения логической единицы. Полученную схему (рис. 20) называют *полусумматором*. «Полу-» в названии этой схемы объясняется тем, что ее можно использовать для сложения только младших разрядов — нельзя учесть перенос в старший разряд. Из двух полусумматоров можно построить полный *сумматор*. Второй полусумматор складывает полученную на первом полусумматоре сумму  $A_1 + B_1$  с переносом из младших разрядов  $P_0$  (рис. 21). Единица переноса является, по существу,

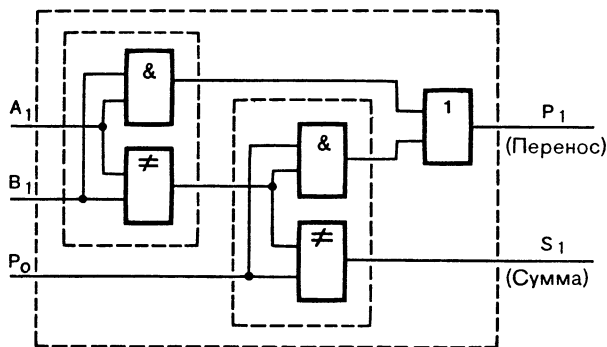


Рис. 21

третьим слагаемым в операции сложения ( $P_i$  — выходной сигнал переноса в сторону старших разрядов). Подобно рассмотренным выше схемам, реализующим самостоятельные логические функции, полный сумматор существует в виде отдельной ИС и имеет собственное обозначение (рис. 22). Таблица истинности полного сумматора определяется правилами сложения двоичной арифметики (табл. 2).

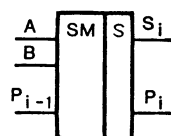


Рис. 22

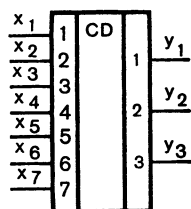


Рис. 23

Используя одноразрядные полные сумматоры, можно легко построить полный сумматор на любое количество разрядов. Сумматор — один из главных рабочих узлов Процессора, особенно той части Процессора, которая ответственна за выполнение арифметических операций. Хотя сумматор непосредственно выполняет только сложение чисел, все остальные арифметические операции — вычитание, умножение, деление — могут быть сведены к реализуемой сумматором операции сложения. Сумматор — активный преобразователь информации. Получив на входы значения слагаемых, сумматор формирует на выходе другое, отличающееся от входных значение — их сумму. В вычислительных машинах, допускающих различные способы представления одного и того же значения, часто возникает необходимость пассивных преобразова-

Т а б л и ц а 2

$A$	$B$	$P_{i-1}$	$S_i$	$P_i$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

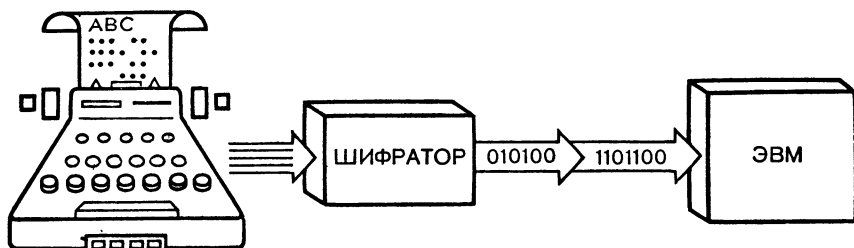


Рис. 24

ний информации — переводов из одной формы представления в другую. Для таких преобразований в блоках ЭВМ используют шифраторы и дешифраторы.

*Шифратор* — это устройство, имеющее для каждого преобразуемого сигнала персональный вход. На выходах шифратора формируется код, значение которого соответствует тому или иному входному сигналу. Код представляет собой совокупность выходных сигналов. Обычно шифратор имеет  $2^N - 1$  входов и  $N$  выходов. Когда на одном из входов появляется логическая единица, шифратор создает на выходе комбинацию двоичных сигналов, соответствующую заданному входу (рис. 23). Единица может появиться в каждый момент только на одном из входов. Активизировать несколько входов нельзя: совокупность выходных сигналов образует в этом случае ошибочный код. Шифратор часто используют, например, для организации связи между ЭВМ и электрической пишущей машинкой. Для этого каждый вход шифратора соединяется со своей клавишей (рис. 24). После нажатия клавиши на выходе шифратора вырабатывается двоичный код, однозначно соответствующий символу нажатой клавиши. Таким образом, используя клавиатуру электрической пишущей машинки и шифратор, можно ввести

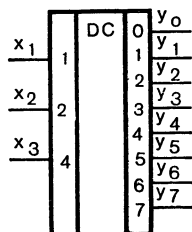


Рис. 25

информацию в ЭВМ для последующей обработки. При выходе информации из ЭВМ на электрическую пишущую машинку приходится решать обратную задачу: надо найти ту единственную клавишу (электромагнит), которой соответствует выводимая из ЭВМ комбинация нулей и единиц. Такое декодирование информации производится с помощью *дешифратора* — устройства, обратного шифратору по принципу дей-

ствия. Дешифратор имеет  $N$  входов и  $2^N$  выходов (рис. 25). На появление любого входного  $N$ -разрядного двоичного кода дешифратор реагирует однозначно соответствующей единицей на одном из выходов. Все рассмотренные здесь устройства — от простого инвертора до дешифратора — на каждый входной сигнал немедленно отвечают преобразованным выходным сигналом, значением функции. Если не считать небольшой временной задержки, о которой говорилось, когда речь шла о работе инвертора, то выходной сигнал появляется практически одновременно с входным. Вычислительная машина не только преобразует сигналы, она умеет их записывать и запоминать. Электронным системам хранения информации — Запоминающим Устройствам ЭВМ, имеющим дело с двоичными числами и кодами, необходимы устройства, способные фиксировать одно из двух устойчивых состояний. Такие устройства должны уметь находиться в одном из устойчивых состояний достаточно долго, а в случае необходимости — при наступлении определенного события — быстро переключаться в другое устойчивое состояние. Такими свойствами обладает триггер — самый массовый запоминающий элемент в вычислительных машинах. Простейший триггер имеет два входа и два выхода (рис. 26, а). Если сигнал

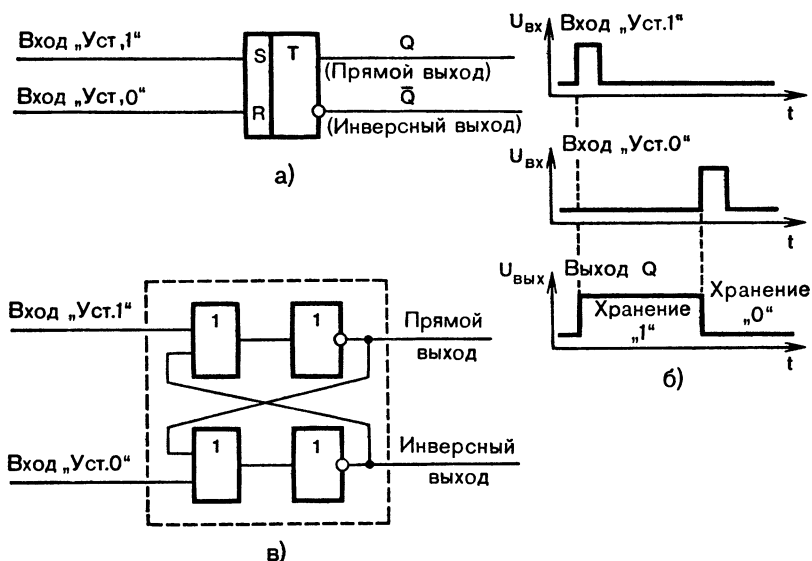


Рис. 26



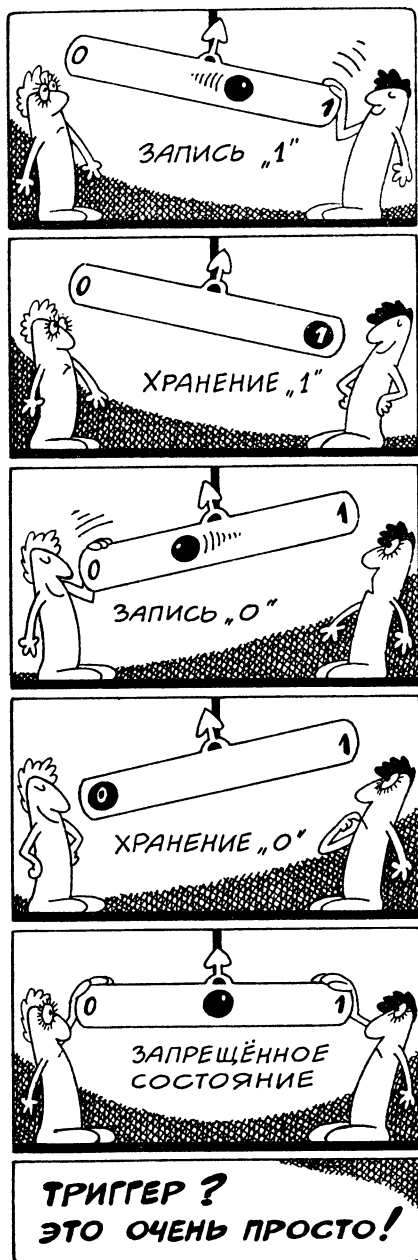


Рис. 27

логической единицы появляется на входе, обозначенном «Установка в 1», то триггер переключается в состояние хранения логической единицы независимо от того, в каком состоянии он находился до этого. Состояние триггера в текущий момент времени (его временную диаграмму) можно определить, оценив состояния его выходов (см. рис. 26, б). Если на выходе  $Q$  — логическая единица, а на выходе  $\bar{Q}$  — логический нуль, то триггер находится в состоянии хранения логической единицы. В этом состоянии триггер может находиться неограниченно долго, пока на входе «Установка в 0» не появится сигнал логической единицы. После этого триггер переключается в состояние хранения логического нуля: на выходе  $Q$  — логический нуль, на выходе  $\bar{Q}$  — логическая единица. На рисунке 26, в приведена схема построения простейшего триггера с помощью элементов ИЛИ и НЕ. Триггер хранит нуль до появления единичного сигнала на входе «Установка в 1» (рис. 27).

Главное назначение триггера — хранение

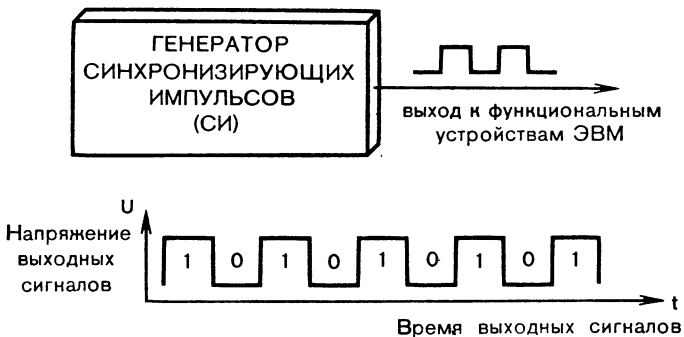


Рис. 28

информации. Смена его состояний выходов происходит почти сразу же после изменения состояний входов, так же как у других рассмотренных ранее устройств. Работа устройств с такой реакцией на сигналы называется *асинхронной*. Этим как бы подчеркивают временную непричастность таких элементов к другим событиям в ЭВМ.

Асинхронные процессы имеют право на жизнь в ЭВМ, особенно там, где речь идет об общении человека и машины. Однако основной технологический поток информационной обработки в ЭВМ состоит из маленьких стандартных операций, строго увязанных между собой во времени — не менее строго, чем операции по сборке автомобиля на конвейере автозавода. В отличие от асинхронных *синхронные* устройства срабатывают от изменения входной информации только в определенные интервалы времени. Устройством, формирующим временные метки для блоков ЭВМ, является *генератор синхронизирующих импульсов* — обычный генератор цифровых сигналов с тщательно стабилизированной частотой следования единичных сигналов (рис. 28). Можно сказать, что ЭВМ имеет свой метроном: генератор синхронизирующих импульсов является частью оборудования Процессора.

Рассмотренный ранее триггер относится к так называемому типу *R-S* (буквы *R* и *S* в названии этого типа триггеров означают сокращения английских слов: SET — установка, RESET — сброс) (рис. 29, а). Синхронизацию триггера *R-S* можно осуществить с помощью схемы совпадения (И): входной установочный сигнал будет иметь для триггера единичное значение только в момент поступления синхронизирующего импульса (рис. 29, б). Единичные сигналы не должны одновременно появляться на

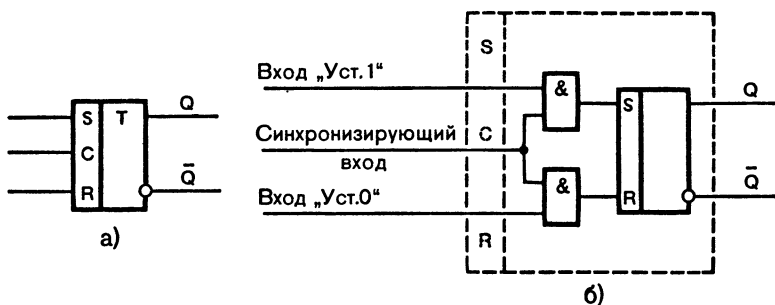


Рис. 29

входах  $R$  и  $S$ , так как в этом случае состояние триггера устанавливалось бы произвольно. Такого недостатка лишен универсальный триггер типа  $J-K$  (рис. 30). Вход  $J$  играет в нем роль входа «Установка в 1», а вход  $K$  — «Установка в 0». Триггер типа  $J-K$  имеет синхронизирующий вход и в отличие от триггера типа  $R-S$  не имеет запрещенных входных комбинаций. Более того, при появлении входной комбинации, запрещенной для обычного  $R-S$  триггера, триггер типа  $J-K$  изменяет свое состояние на противоположное. Эту последнюю особенность  $J-K$  триггера широко используют при построении различных счетчиков. Триггер типа  $J-K$  имеет асинхронные входы для предварительной установки: вход  $S$  для предварительной установки триггера в 1 и вход  $R$  для предварительной установки в 0. Его временная диаграмма приведена на рисунке 30, б. С помощью  $J-K$  триггеров можно строить различные счетчики. Например, 3-х разрядный двоичный счетчик (рис. 31, а) образован тремя  $J-K$  триггерами. Общую для всех триггеров шину сброса используют для предварительного сброса счетчика в исходное

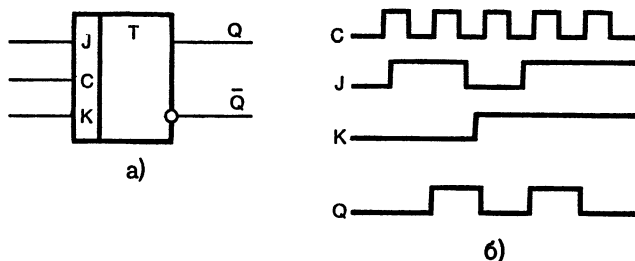


Рис. 30

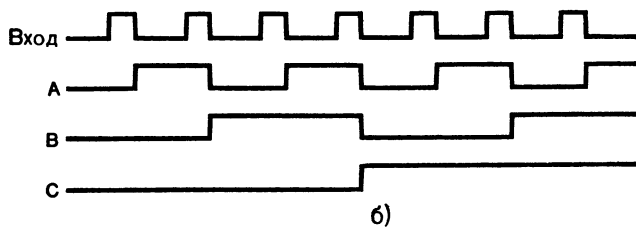
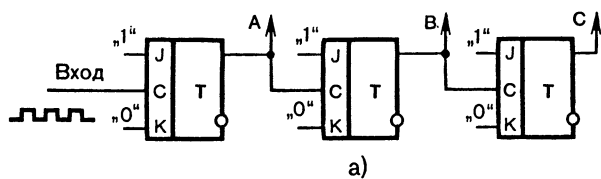


Рис. 31

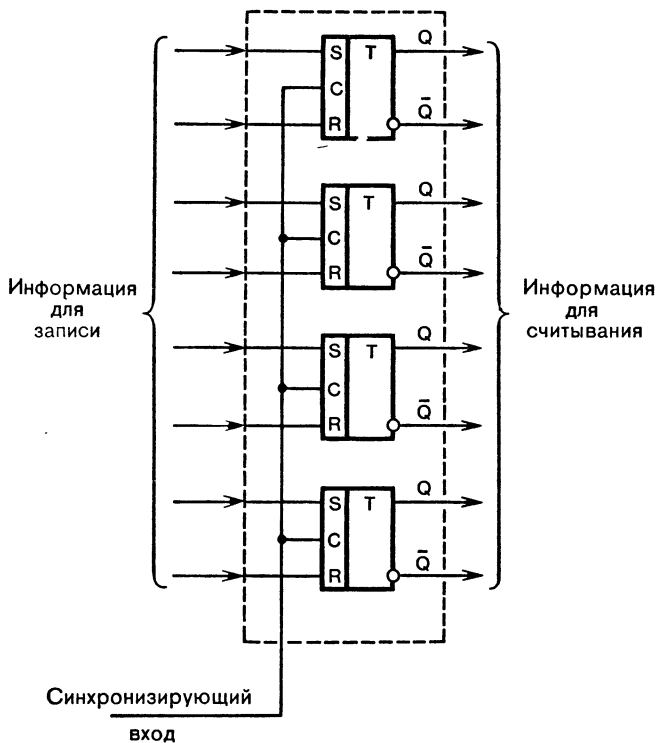


Рис. 32

(нулевое) положение. Когда на входе счетчика появляется сигнал логической единицы (импульс), счетчик увеличивает на единицу хранящееся в нем число (сначала это нуль). Таблицу работы счетчика нетрудно продолжить: далее цикл повторяется. Временная диаграмма счетчика приведена на рисунке 31, б. Триггер способен запомнить и хранить один разряд двоичного числа (кода), т. е. один *бит* — так называют наименьшую хранимую порцию информации. Следовательно, для хранения  $N$ -разрядного двоичного числа (кода) надо построить цепочку из  $N$  триггеров. Такие цепочки триггеров называют *регистрами*. Регистры используют обычно для кратковременного хранения информации. Триггеры, объединенные в регистр, имеют общую цепь синхронизации (см. рис. 32, на котором изображен 4-разрядный регистр). Это означает, что запись во все разряды регистра происходит одновременно. Путем несложного техническо-

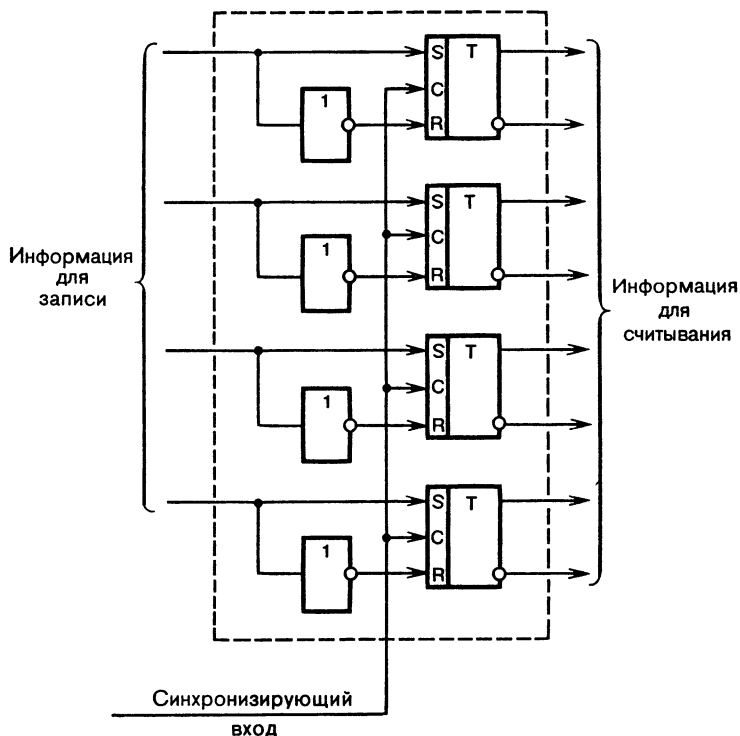


Рис. 33

го решения, показанного на рисунке 33, можно обойти возможность появления на входе триггера типа *R-S* запрещенной комбинации и использовать такие триггеры для построения регистра.

Все рассмотренные устройства — от простого инвертора до счетчиков и регистров — служат основным строительным материалом при создании электронного оборудования ЭВМ. Вместе с тем Процессор, ЗУ, УВВ представляют собой нечто большее, чем простая совокупность составляющих их элементов: главное в функционировании всякого сложного устройства — взаимодействие между различными его элементами. Вопросы взаимодействия посвящены следующие страницы книги.

## ПАМЯТЬ

Процессор является организационным центром ЭВМ, Однако около 80% всего оборудования вычислительной машины составляют различные Запоминающие Устройства (ЗУ) и средства связи их с другими блоками ЭВМ. Производительность ЭВМ во многом определяется быстродействием ЗУ. Основные функции ЗУ — запоминание и хранение информации. В памяти машины располагается самая различная информация: целые и вещественные числа, коды графических изображений и текстовые сообщения. Здесь же, в ЗУ, хранится программа, которая организует обработку данных в процессе решения задачи (рис. 34).

ЗУ состоит из *запоминающих элементов*. В качестве запоминающего элемента может быть использована либо небольшая часть физической среды, обладающей определенными свойствами, например магнитными или электрическими, либо сравнительно простое устройство, такое, как уже знакомый читателю триггер. Запоминающие элементы объединяют в *ячейки* (рис. 35) примерно так же, как из триггеров строят регистры. Ячейка служит для хранения *машинного слова* — стандартной порции информации, которую ЗУ может выдать или принять за одну операцию обмена информацией с другими устройствами. Длина машинного слова (т. е. число его двоичных разрядов) обычно фиксирована в каждой модели ЭВМ: в БЭСМ-6 — 48 бит, в СМ ЭВМ — 16 бит... Индивидуальное лицо ячейки — номер, однозначно идентифицирующий ее во множестве всех ячеек ЗУ. Этот номер

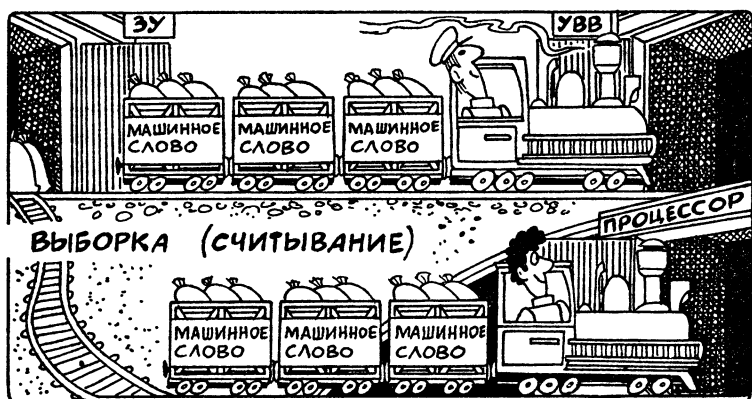
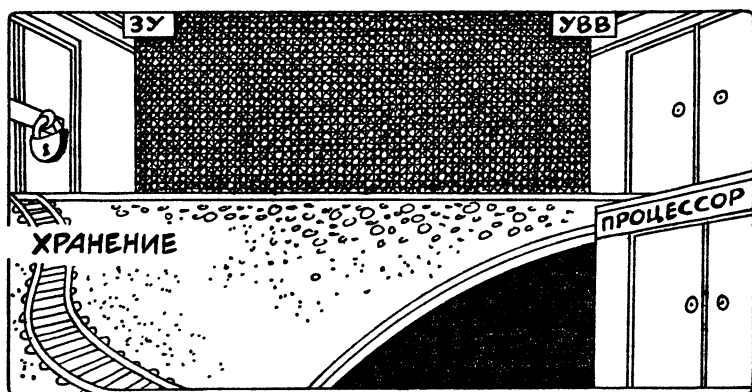
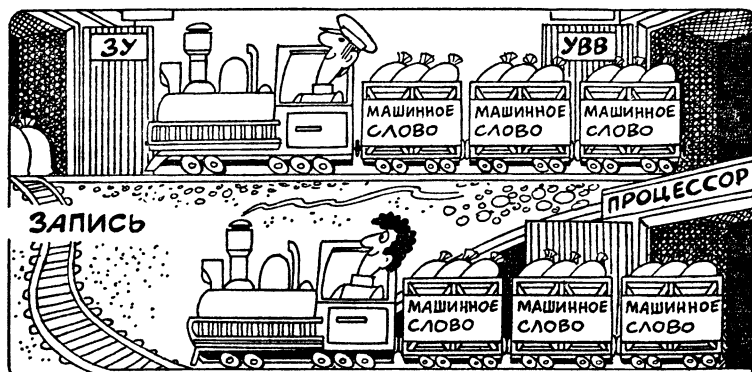


Рис. 34

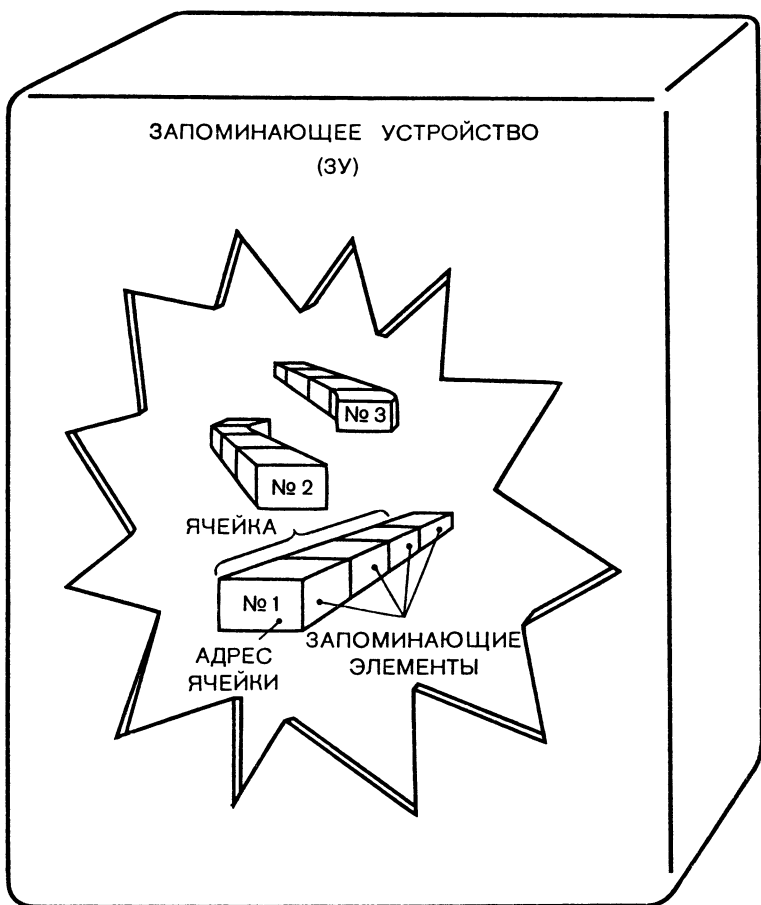


Рис. 35

называется *адресом ячейки*. По адресу в ЗУ помещается (записывается) либо извлекается (считывается) машинное слово. Адрес играет основную роль при поиске информации. Если известен адрес ячейки ЗУ, то добраться до ее содержимого столь же просто, как найти заданную клеточку в игре «Морской бой» или увидеть фигуру на указанном поле шахматной доски. Электрические соединения, связывающие ЗУ с Процессором и УВВ, называют *информационными цепями* или *шинами*. Информационные цепи образованы проводниками, соединяющими между собой регистры, счетчики, различные элементы





Рис. 36

памяти. Информационные цепи соединяют интегральные схемы, выполняющие обработку информации. Среди информационных цепей ЗУ выделяются *адресные шины*, а также *шины ввода* и *вывода* информации (рис. 36). Управляющие сигналы задают направление требуемой передачи: машинное слово может направляться в ЗУ для хранения (процесс записи) или считываться из ЗУ (процесс чтения). По адресным шинам поступают сведения о месте разыскиваемого машинного слова в памяти. По входным шинам направляется информация, которую необходимо запомнить в ячейке ЗУ с выбранным адресом, а по выводным шинам информация может быть прочитана из ЗУ. Время считывания информации из любой ячейки рассмотренного ЗУ одинаково. Такие устройства называют ЗУ с *произвольным доступом*. Наибольшее количество единиц информации, которое может одновременно храниться в ЗУ, определяет его емкость. Емкость (наряду с быстродействием) является важнейшей характеристикой ЗУ. Возможность решения той или иной задачи на ЭВМ в значительной степени зависит от емкости ЗУ. Широко применяют оценку емкости в количестве ячеек ЗУ, каждая из которых способна хранить ровно одно машинное слово. Впрочем, есть и отклонения от этого правила. Например, в машинах серии ЕС ЭВМ одна ячейка ЗУ содержит только один *байт* (восемь двоичных разрядов) и для хранения 32-разрядного машинного слова приходится использовать четыре подряд

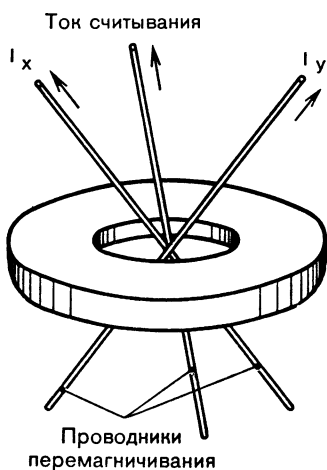


Рис. 37

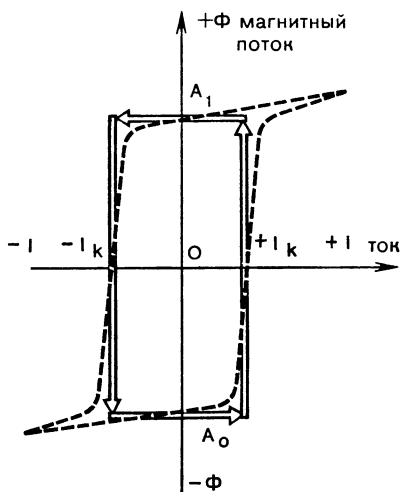


Рис. 38

расположенные ячейки. По этой причине емкость ЗУ часто выражают в байтах, килобайтах, мегабайтах ( $1 \text{ Кбайт} = 2^{10} \text{ байт}$ ,  $1 \text{ Мбайт} = 2^{10} \text{ Кбайт} = 2^{20} \text{ байт}$ ) или иногда в битах, килобитах и мегабитах. ЗУ, построенное из триггеров, имеет очень высокое быстродействие: время записи и считывания информации измеряется наносекундами. Однако стоимость такого ЗУ до недавнего времени оставалась весьма высокой, а это накладывало ограничение на конструирование ЗУ большой емкости из триггеров. В поисках дешевых запоминающих элементов конструкторы ЭВМ обратились к *ферритам* — искусственным материалам с ярко выраженными магнитными свойствами. В качестве запоминающих элементов используют кольцеобразные ферриты (рис. 37): замкнутый в кольцо магнитный поток наилучшим образом сохраняет энергию, затрачиваемую на перемagnetничивание. Проводя кольца токоведущими проводниками, можно изменять состояние намагниченности ферритового кольца, меняя направление тока в проводниках. Зависимость направления и величины магнитного потока в феррите от величины и направления тока в проводниках имеет вид петли гистерезиса (рис. 38), которая обеспечивает устойчивость двух состояний запоминающего элемента.

Процессы, которые протекают в запоминающих элементах на ферритовых кольцах во время записи или счи-

тывания информации, определяют внутреннюю организацию этого типа ЗУ и заслуживают того, чтобы о них было рассказано подробнее. Уславливаются, что кольцо хранит единицу, если значение магнитного потока внутри кольца соответствует точке  $A_1$  (рис. 38), и нуль при значении магнитного потока, соответствующем точке  $A_0$ . Размер ферритового кольца очень мал (наружный его диаметр  $0.4 \div 0.8$  мм), и для его перемагничивания требуется небольшой ток. Поэтому кольцо удается перемагнитить очень быстро — за сотни наносекунд. Резкое изменение магнитного потока в феррите происходит в тот момент, когда величина тока в проводниках перемагничивания (рис. 37) достигнет значения  $+I_k$  или  $-I_k$  (рис. 38). Из школьного курса физики известно, что изменение магнитного потока в замкнутом контуре, роль которого здесь выполняет проводник считывания, вызывает электродвижущую силу (ЭДС). По величине ЭДС можно узнать, какое значение — 0 или 1 — хранит кольцо. Если в кольце хранилась единица (рис. 38, точка  $A_1$ ), то, перемагничивая кольцо к точке  $A_0$ , можно вызвать быстрое изменение величины и направления магнитного потока. Это изменение потока отзывается появлением ЭДС в замкнутом контуре проводника считывания. Если кольцо хранило нуль, то перемагничивание к точке  $A_0$  не вызывает изменения магнитного потока в кольце, и, как следствие, величина ЭДС, наведенной в проводнике контура считывания, будет практически нулевой. Недостатком ферритового кольца как запоминающего элемента является разрушение информации при считывании. В рассмотренном примере разрушаются единицы. Для обеспечения длительного хранения необходимо регенерировать (восстанавливать) считанную информацию. Регенерацию производят с помощью вспомогательного, буферного регистра. Считанная из ферритового ЗУ информация помещается в буферный регистр. Отсюда она передается в другие блоки ЭВМ, а по линиям обратной связи — снова на запись. В сущности, за каждой операцией считывания следует операция записи. Так тратится дополнительное время на «внутрихозяйственные расходы». ЗУ на ферритовых кольцах более емки и дешевы по сравнению с триггерными. За эти достоинства приходится платить потерей быстродействия. Как триггерные, так и ферритовые ЗУ принадлежат к типу ЗУ с произвольным доступом: время считывания информации из любой ячейки ЗУ не зависит от адреса ячейки.

Другой тип ЗУ представляют Запоминающие Устройства с *последовательным доступом*. Если в ЗУ первого типа среда, используемая для хранения информации (триггеры и ферриты), неподвижна относительно средств записи и считывания, то в ЗУ второго типа среда перемещается. При этом время считывания зависит от того, какое положение в текущий момент занимает информация относительно средств записи и считывания. Примером такой среды может служить магнитная лента, которая перемещается относительно магнитных головок, выполняющих запись или считывание информации. Запоминающей средой в ЗУ с последовательным доступом обычно является тонкий слой (от 2 до 35 микрон) ферромагнитного лака, который нанесен на прочную немагнитную основу. Для магнитной ленты, например, такой основой служит лавсан (полиэтилентерефталат). Хранение информации на подвижном магнитном носителе, так же как и в ферритовом кольце, не связано с дополнительными затратами энергии. Однако способ записи и считывания здесь совершенно иной. Важным элементом любого

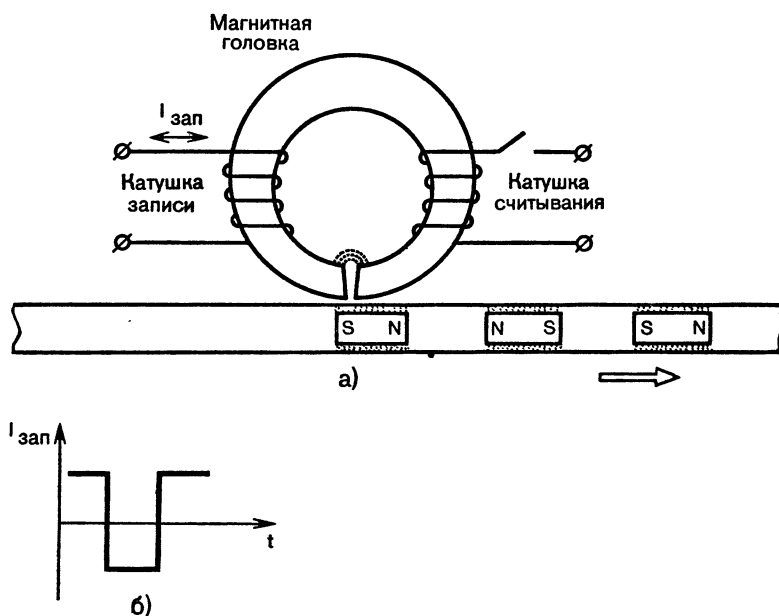


Рис. 39

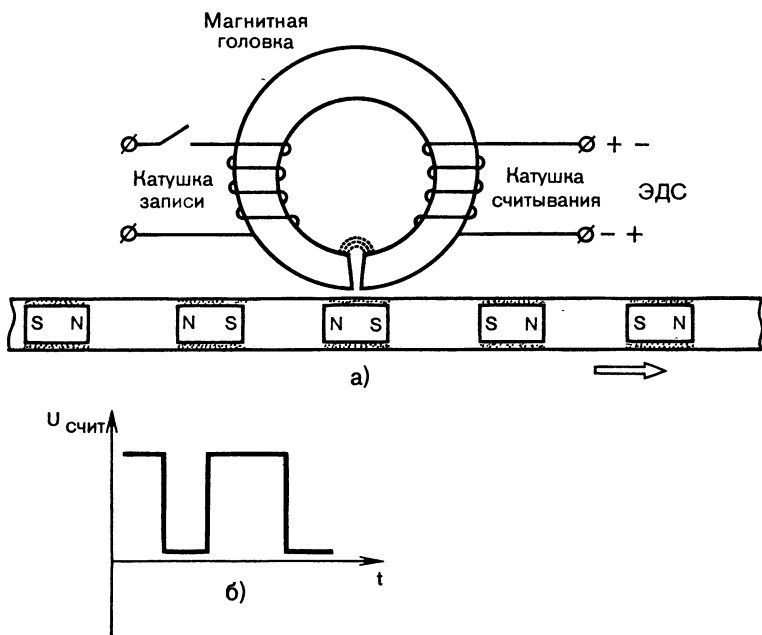


Рис. 40

ЗУ, использующего в своей работе подвижный магнитный носитель, является магнитная головка — почти замкнутый электромагнит, имеющий две катушки (одну для считывания, другую для записи). Электромагнит головки не случайно «почти» замкнут: небольшой искусственный разрыв (зазор) магнитопровода позволяет создавать на этом участке магнитной головки очень высокую напряженность магнитного поля и тем самым оставлять при записи на перемещаемом носителе магнитные отпечатки. Каждый такой отпечаток — не что иное, как миниатюрный магнит, повернутый магнитным полем головки вдоль или против движения носителя, что соответствует нулю или единице (рис. 39). Если отключить ток записи и вновь перемещать намагниченные участки носителя в непосредственной близости от зазора головки, то по мере прохождения «магнетиков» в магнитопроводе будет появляться переменное магнитное поле, точнее, переменный магнитный поток. Это приводит в свою очередь к возникновению ЭДС в замкнутом контуре катушки считывания (рис. 40, а): величина и направление ЭДС

определяются записанной на носитель двоичной информацией (характер изменения напряжения во времени отражен на рис. 40, б). Описанный принцип используется в работе таких ЗУ с последовательным доступом к информации, как накопитель на магнитной ленте, магнитные диски и магнитные барабаны.

Накопитель на магнитной ленте (НМЛ) — магнитофон — это самое емкое ЗУ. Одна катушка магнитной ленты длиной 725 м может хранить от 200 до 500 Мбит. С другой стороны, это самое медленное ЗУ: доступ к необходимой порции информации осуществляется механическим перемещением — перемоткой ленты. Несмотря на то что скорость движения ленты достигает 5 м/с, время поиска информации может измеряться десятками секунд. Однако, после того как нужная порция информации найдена, непрерывная передача информации может осуществляться с высокой скоростью — до 0,5 Мбит/с.

Информация записывается на магнитную ленту блоками, или *зонами*. Каждая зона содержит обычно большую группу машинных слов. Между соседними зонами на ленте оставлены небольшие промежутки, благодаря которым ленту можно остановить в начале выбранной зоны.

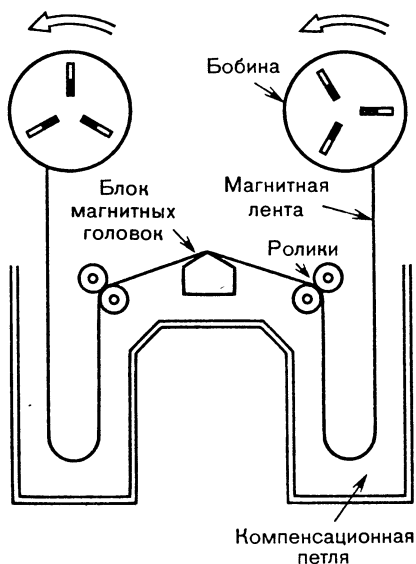


Рис. 41

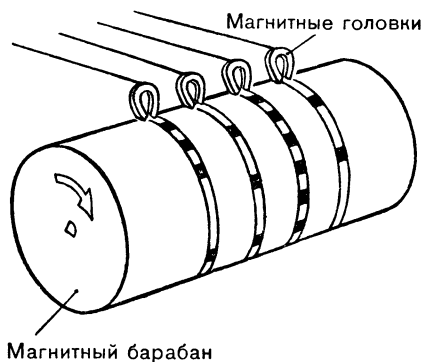


Рис. 42

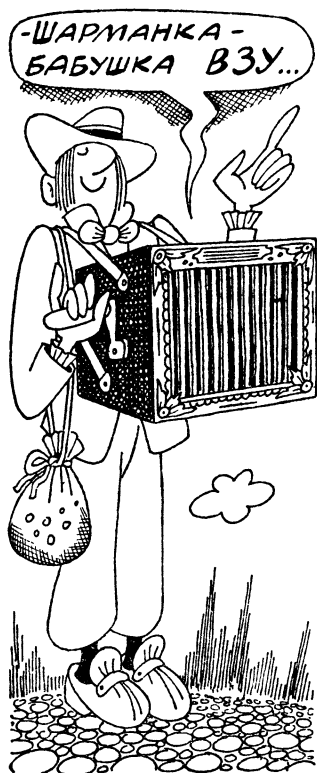


Рис. 43

Этим же целям служат компенсационные петли (рис. 41): они стремятся снизить влияние инерционных масс. Более высокое быстродействие по сравнению с магнитофоном имеют накопители на магнитных дисках и магнитных барабанах. Сравнительно небольшая поверхность магнитного барабана (рис. 42) перемещается очень быстро: скорость вращения барабана вокруг оси достигает нескольких тысяч оборотов в минуту. На поверхности барабана выделены дорожки, на которых записана информация. Линии на поверхности барабана, параллельные образующей его цилиндра, называются строками. Над каждой дорожкой установлена головка записи-считывания. При вращении барабана строки последовательно проходят под группой неподвижных головок. Процессы записи и чтения информации синхронизированы с вращением барабана. Среднее время поиска и считывания со-

ставляет  $2 \div 10$  миллисекунд. Емкость накопителя на магнитном барабане (НМБ) достигает нескольких десятков мегабитов. Это намного меньше емкости магнитофона, но превосходит емкость ЗУ, построенного из триггеров или ферритовых колец. Идея такого информационного носителя, как барабан, есть продолжение линии развития звукозаписывающей и звуковоспроизводящей аппаратуры: шарманка — граммофон — магнитофон — магнитный барабан (рис. 43).

Накопитель на магнитных дисках (НМД) сочетает в себе достоинства магнитофона (емкость) и магнитного барабана (быстродействие). Набор имеющих единую ось тонких дисков, изготовленных из алюминиевых сплавов с ферромагнитным покрытием поверхности, называют *пакетом* магнитных дисков. Пакет дисков имеет большую

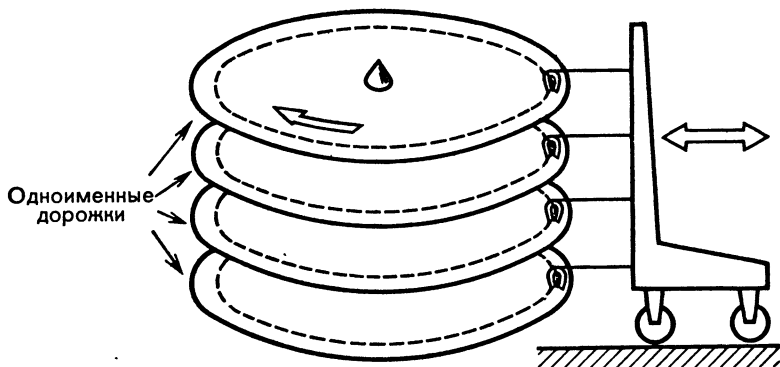


Рис. 44

по сравнению с барабаном рабочую поверхность (рис. 44). Над каждой поверхностью располагается одна подвижная головка записи-чтения, а все головки пакета объединены в блок головок. Он размещается на специальной каретке, которая может перемещать одновременно головки по радиусу дисков. Так обеспечивается доступ головок к любому месту на поверхности диска. При этом на одной поверхности магнитного диска магнитная головка поочередно работает с 202 дорожками. Время поиска информации складывается из времени установки блока головок над нужной дорожкой (время позиционирования) и запаздывания, определяемого вращением диска. В зависимости от модели накопителя время доступа колеблется от 10 до 90 миллисекунд. Диапазон емкости одного пакета магнитных дисков — от нескольких мегабитов до сотен мегабитов у разных типов устройств. Общая черта ЗУ на магнитных дисках и магнитных барабанах — постоянное периодическое движение носителей информации в обоих типах накопителей. Любое машинное слово в этих ЗУ может быть выбрано в пределах некоторого максимального временного интервала, определяемого периодом вращения диска или барабана. С каждым из таких ЗУ, как магнитофон, пакет магнитных дисков или накопитель на магнитном барабане, связан *контрôллер* — местный администратор. Полученные от ЭВМ или человека-оператора сигналы контрôллер преобразует в сигналы, управляющие работой транспортного механизма, который перемещает носитель информации, синхронизирует протекающие в накопителе процес-



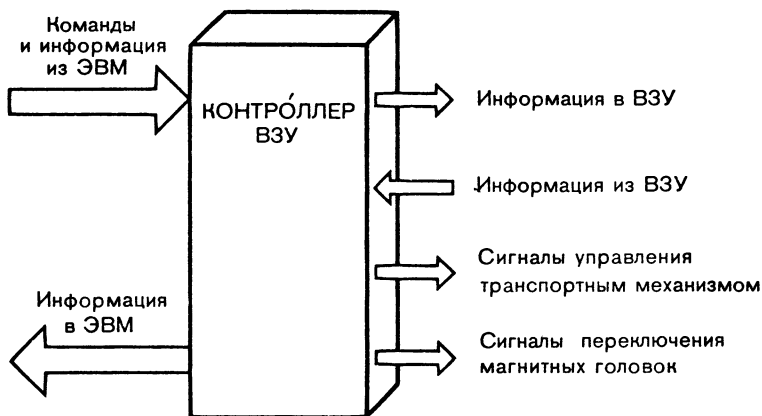


Рис. 45

сы, переключает режимы работы магнитных головок (чтение или запись), направляет информационные потоки (рис. 45). Различные подходы к реализации ЗУ определили такую их классификацию:

- Сверхоперативные Запоминающие Устройства (СОЗУ) — чаще всего триггерные ЗУ;
- Оперативные Запоминающие Устройства (ОЗУ) — обычно это ЗУ, построенные из ферритовых колец;
- Внешние Запоминающие Устройства (ВЗУ) — накопители на магнитной ленте, барабанах, дисках.

Название устройства в такой классификации отражает степень расторопности, с которой ЗУ может обмениваться информацией с другими блоками ЭВМ. Стремление создать память, наилучшим образом сочетающую в себе быстродействие СОЗУ с емкостью и экономичностью ВЗУ, привело к иерархической системе запоминающих устройств (рис. 46). Эффективная работа такой системы оказалась возможной в силу того, что в каждый конкретный момент времени выполняется ограниченный участок программы и используется только часть хранимых в памяти данных. Если требуемая Процессору информация отсутствует в СОЗУ (это главный поставщик информации для Процессора), происходит обращение к ОЗУ или к ВЗУ (рис. 47). Взаимодействие между ЗУ различных уровней иерархии напоминает переговоры короля с коровой в «Балладе о королевском бутерброде».

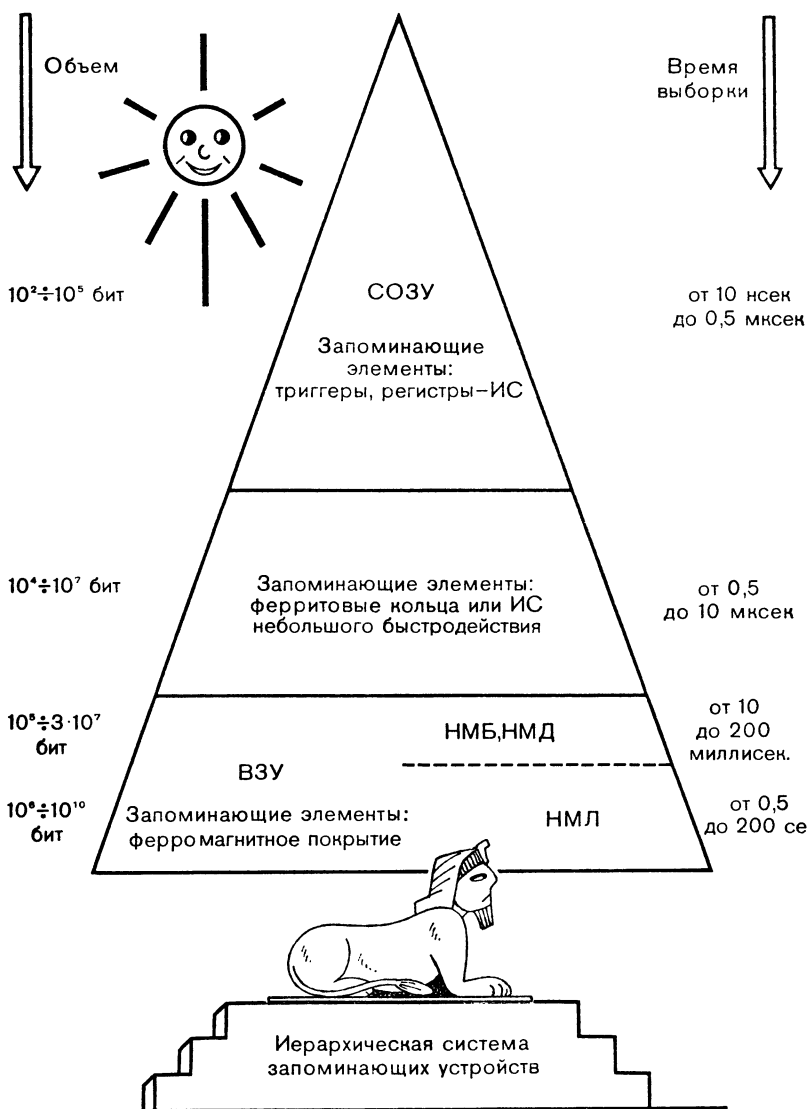


Рис. 46

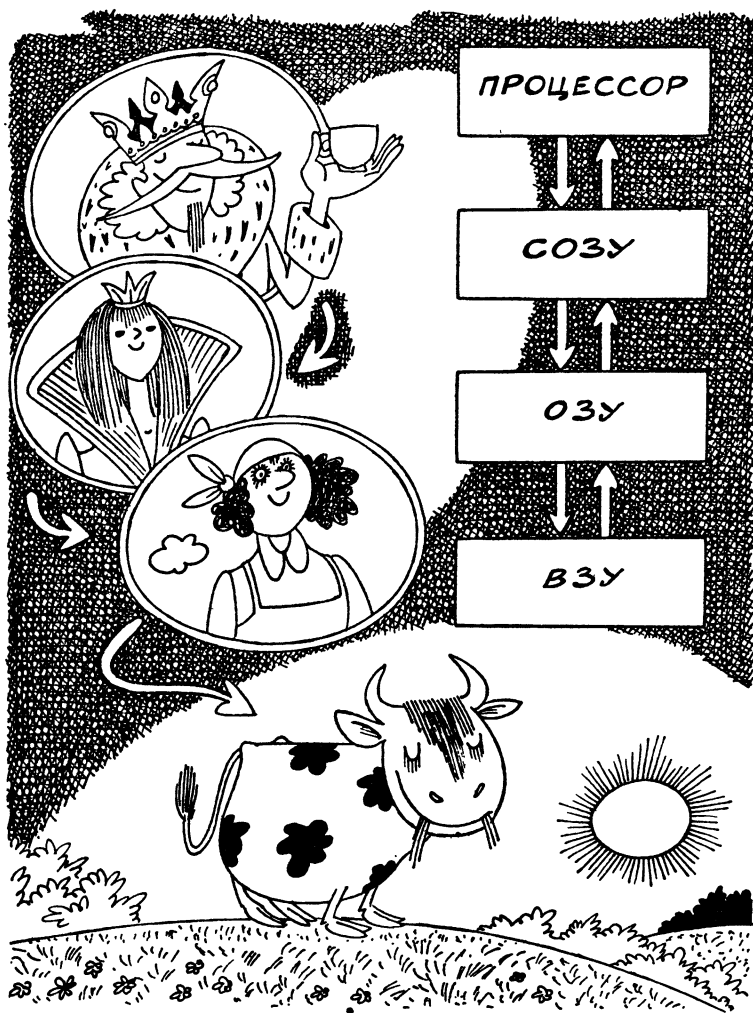


Рис. 47

## КАК РАБОТАЕТ ПРОЦЕССОР?

**Я**дро вычислительной машины — Процессор состоит из двух самостоятельных устройств (рис. 48), тесно взаимосвязанных между собой, — Арифметико-Логического Устройства (АЛУ) и Устройства Управления (УУ). АЛУ умеет выполнять элементарные арифметические и логиче-

ские операции: «сложить два числа», «сравнить два значения», «логически умножить две переменные» и т. д. АЛУ обрабатывает и изменяет содержимое ячеек ЗУ. Информацию о том, какую операцию предстоит выполнить АЛУ, в каких ячейках ЗУ необходимо взять исход-

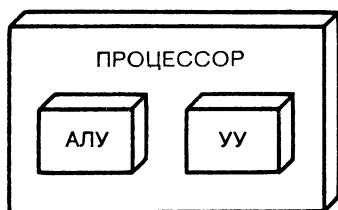


Рис. 48

ные данные и куда поместить конечный результат, в Процессор поставляет команда Программы (рис. 49). Команда — самостоятельный элемент Программы, размещаемой, как правило, в последовательно расположенных ячейках ЗУ. К моменту завершения команды АЛУ вырабатывает результат соответствующей операции — сумму, разность, конъюнкцию и т. д. Главным организатором работ, выполняемых в ЭВМ (не только в АЛУ), является Устройство Управления. Вот некоторые из его функций:

- извлечение (выборка) команды из ячейки ЗУ;
- интерпретация (декодирование) команды и соответствующая настройка АЛУ;
- выборка из ЗУ данных, указанных в команде, и запись результата в ЗУ;
- выборка из ЗУ следующей команды либо независимо от результата выполнения предыдущей команды, либо с учетом определенных условий.

Для программиста команда представляется элементарным шагом в работе ЭВМ. В действительности же команда состоит из ряда еще более простых операций, последовательно выполняемых в несколько этапов под руководством УУ. На рисунках 50—55 показана серия стоп-кадров «фильма», рассказывающего о выполнении машинной команды сложения.

Основные действующие лица нашего «фильма»:

- Регистр Команд (РК), принимающий из ЗУ очередную команду;

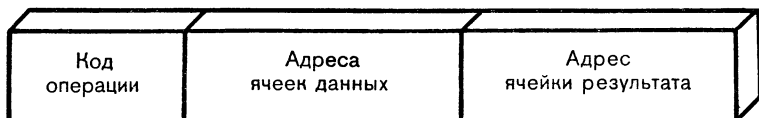


Рис. 49

- Дешифратор Команд (ДшК), который декодирует команду, находящуюся в РК;
- Счетчик Адреса Команд (СчАК), фиксирующий адрес выполняемой команды и формирующий адрес следующей команды;
- Регистр Адреса (РА), который хранит адрес обращения в ЗУ;
- Регистр 1 (Р1) и Регистр 2 (Р2), принимающие из ЗУ исходные данные, в нашем примере слагаемые;
- Сумматор (См), который непосредственно выполняет операцию сложения;
- Регистр Результата (РР), предназначенный для записи результата операции и последующей пересылки результата в ЗУ.

Шины, соединяющие между собой все эти элементы, по мере необходимости расчлениаются с помощью многовходовых схем совпадения («И» на рисунках не показаны), что дает возможность управлять передачей ин-

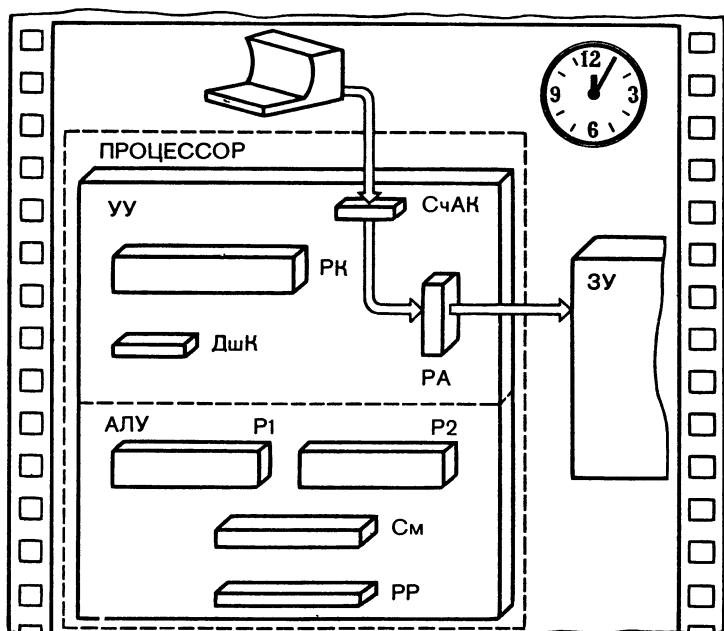


Рис. 50

формации между узлами — коммутировать и переключать информационные потоки. Кроме схем совпадения, за кадром остался генератор синхронизирующих импульсов.

### Кадр 1

Будем считать, что в начале работы адрес ячейки ЗУ, в которой хранится адрес первой выполняемой команды программы, набран на клавишных переключателях пульта управления машиной. Нажатие кнопки «Пуск» оживляет Процессор: запускается генератор синхронизирующих импульсов, УУ вырабатывает последовательность управляющих сигналов. В результате этого адрес команды будет принят в СчАК и оттуда передан в РА. Процессор обращается в ЗУ по адресу, выставленному на выходных шинах РА (рис. 50).

### Кадр 2

Команда, затребованная Процессором, считывается из ЗУ и поступает в РК.

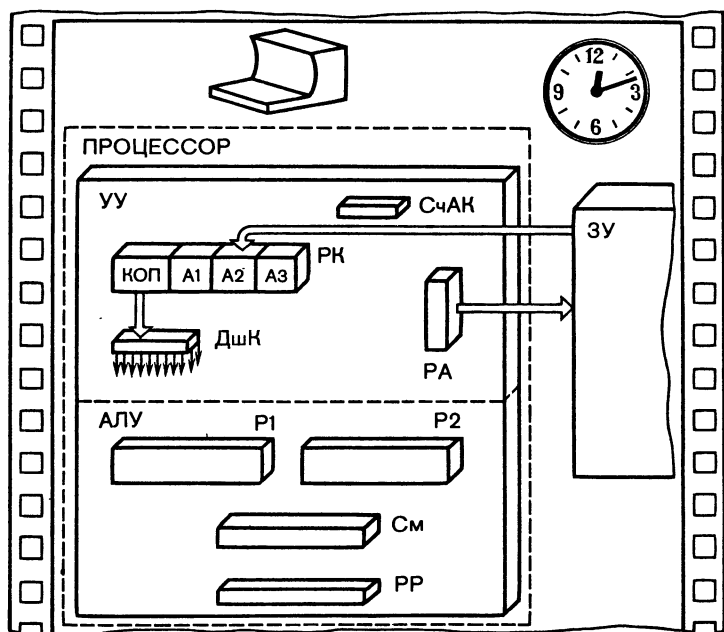


Рис. 51

От обращения к ЗУ до получения команды в РК могло пройти несколько десятков или сотен наносекунд или даже микросекунды. Это время определяется быстродействием ЗУ. Поскольку память ЭВМ не однородна, а представляет собой иерархическую систему запоминающих устройств, то Процессор непосредственно связан транспортными конвейерами только с СОЗУ (в более простых и дешевых моделях с ОЗУ) (рис. 51).

### Кадр 3

ДшК декодирует команду. Выясняется, например, что в РК находится команда «сложить число, которое расположено по адресу А1, с числом, находящимся по адресу А2, результат записать в ячейку с адресом А3». Процессор обращается по адресу А1 в ЗУ за первым слагаемым. Получив его, Процессор записывает первое слагаемое в буферный регистр Р1 — идет подготовка к выполнению операции (рис. 52).

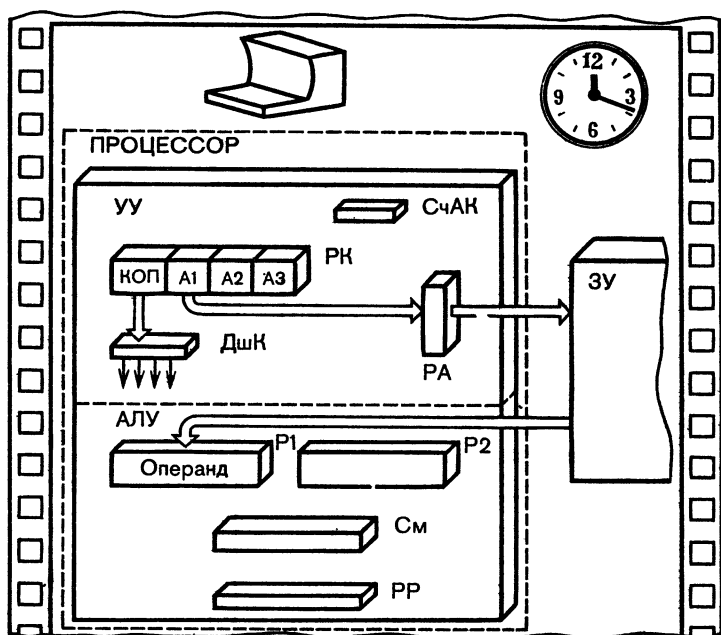


Рис. 52

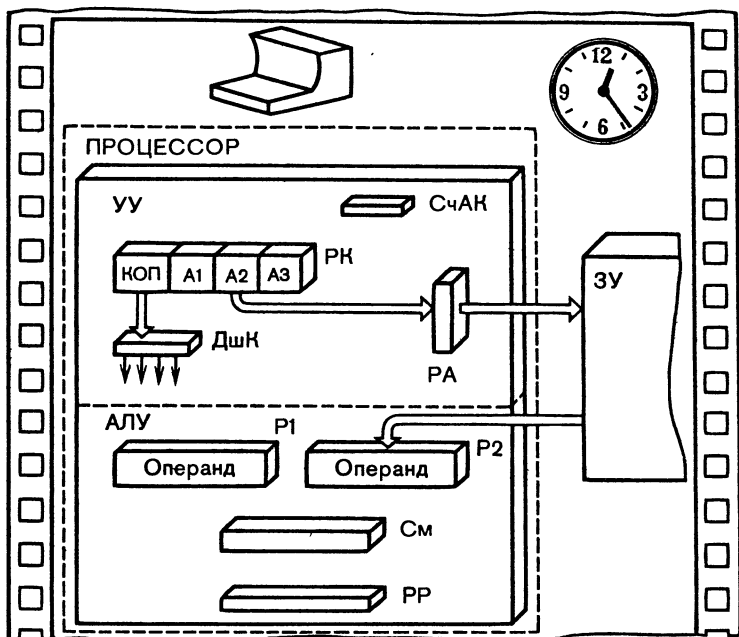


Рис. 53

#### Кадр 4

На следующем этапе из ЗУ по адресу A2 считывается второе слагаемое и направляется на временное хранение в другой буферный регистр — P2. Оба операнда, участвующие в выполнении команды, подготовлены (рис. 53).

#### Кадр 5

Сумматор принимает слагаемые, хранящиеся в P1 и P2, и выполняет операцию сложения. Сумма записывается в РР. На этом этапе Процессор не взаимодействует с ЗУ. Время выполнения операции — один цикл Процессора — несколько десятков наносекунд (рис. 54).

#### Кадр 6

Процессор формирует в РА адрес, по которому следует направить результат выполненной операции. Выполнение команды завершено. Следующая команда Программы будет выбрана автоматически в связи с тем, что



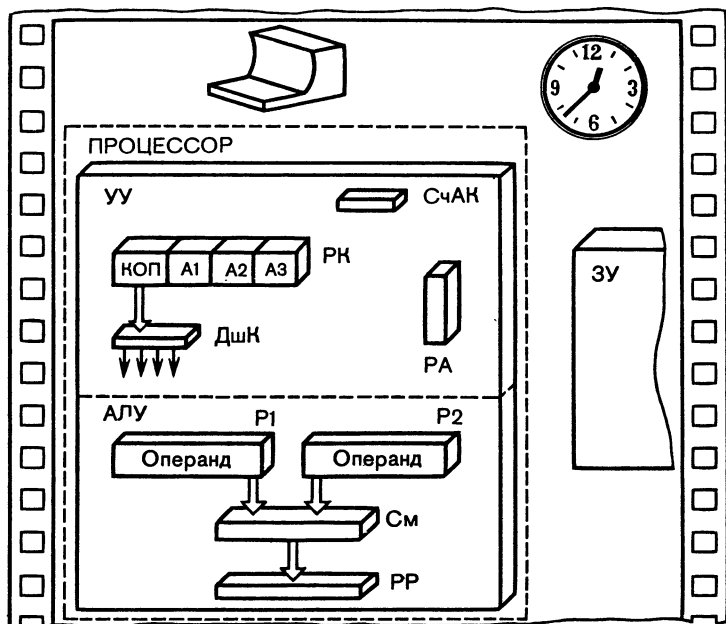


Рис. 54

в СчАК при завершении команды прибавляется единица (рис. 55).

Решение задачи на ЭВМ можно представить как выполнение последовательности команд, составляющих Программу. Каждая последующая команда извлекается из ЗУ путем увеличения СчАК на 1. Естественную последовательность выполнения команд могут изменить так называемые *команды передачи управления*, обеспечивающие проверку различных условий (например, равенство нулю результата только что рассмотренной операции) и на основании таких проверок выработку решения о выборе следующей команды из двух или нескольких альтернатив. Нормальная последовательность выполнения команд может также нарушаться прерываниями. Каждая современная ЭВМ имеет развитую систему прерываний — своеобразный сплав специальных программ и оборудования. Не вдаваясь пока в подробности, заметим, что причины прерывания работы Процессора могут быть различными: грубая ошибка в Программе, поломка оборудования, запрос Устройством Ввода-Вывода разрешения на обмен

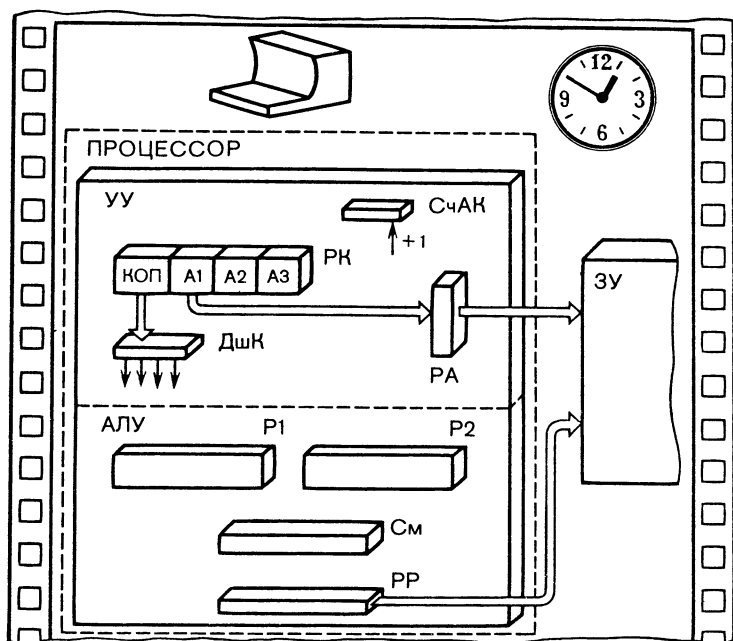


Рис. 55

информацией с ЗУ, Процессор обрабатывает прерывание и возвращается к своим основным обязанностям примерно так же, как Вы возвращаетесь к чтению книги, которое было прервано телефонным звонком.

## ВВОД-ВЫВОД ЭВМ

### Пакетный и диалоговый режимы

Для того чтобы ЭВМ могла приступить к решению задачи, в машину должна быть введена Программа со всеми необходимыми данными (рис. 56). После решения задачи ЭВМ должна передать полученные результаты пользователю. Организацию обмена информацией с вычислительной машиной называют вводом-выводом, а разнообразные функции, обеспечивающие общение ЭВМ с внешним миром, выполняют Устройства Ввода-Вывода (УВВ).

Не всегда информацию вводит в машину человек, и не любой вывод для него предназначен. Так, ЭВМ,

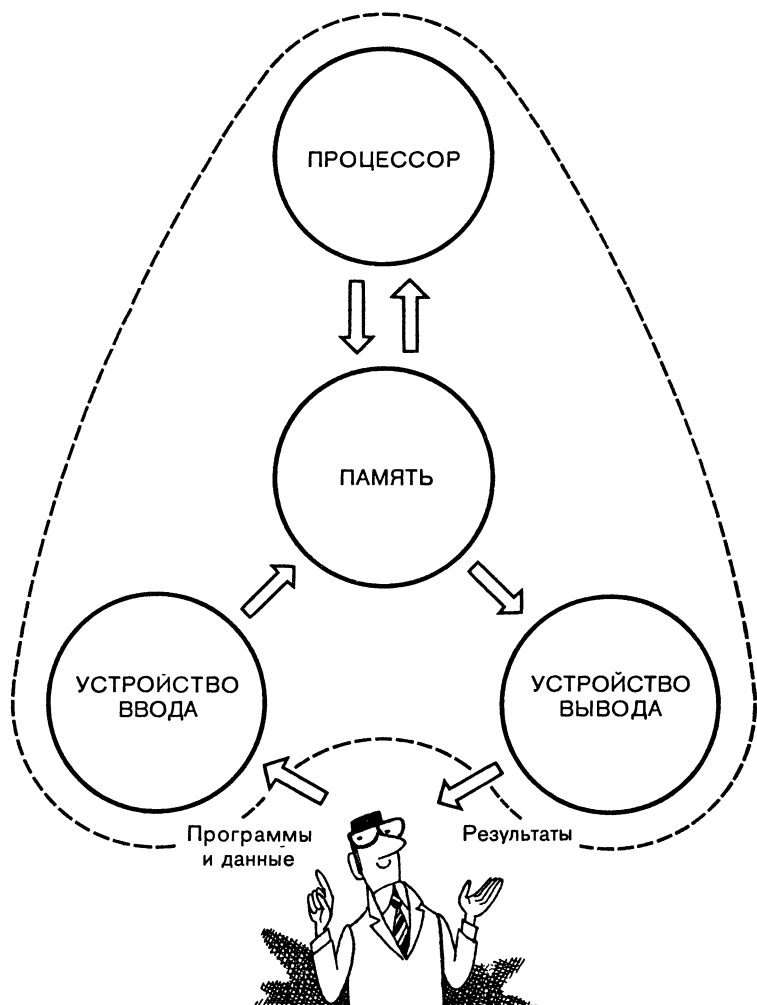


Рис. 56

управляющая автоматизированным химическим заводом, может получать информацию непосредственно от приборов-анализаторов и датчиков, а ЭВМ, обслуживающая космический корабль, может направлять управляющие сигналы двигателям, корректирующим траекторию полета. Непосредственное общение человека с ЭВМ в процессе так называемого «ручного» ввода Программы и дан-

ных с пульта, когда каждое отдельное машинное слово вводится в машину человеком-оператором, весьма трудоемко даже при небольших Программах. При таком режиме работы не может быть и речи об эффективном использовании ЭВМ. Противоречие между высоким быстродействием Процессора, с одной стороны, и человеческим несовершенством, с другой, разрешают следующими двумя основными способами. При первом способе информацию, предназначенную для ввода в ЭВМ, заранее подготавливают на каком-нибудь промежуточном информационном носителе, например на перфокартах, перфолентах или магнитных лентах. Устройства, помогающие выполнить подготовительную работу, предшествующую вводу в ЭВМ Программы и данных, называют *Устройствами Подготовки Информации*, добавляя при этом наименование информационного носителя — на перфокартах, на перфоленте или на магнитной ленте. Подготовку информации на промежуточном носителе можно выполнять без лишней спешки, поскольку устройства подготовки информации никак не связаны с ЭВМ, они работают независимо, и необходимость состязаться в скорости с машиной отпадает. В то же время подготовленную Программу и данные, отперфорированные на картах или перфоленте или записанные на магнитной ленте, можно ввести в ЭВМ с высокой скоростью благодаря специальным считывающим устройствам: для каждого типа носителя существует свое специфическое устройство ввода. По сравнению с «ручным» вводом информации с пульта ЭВМ способ предварительной подготовки позволяет существенно сократить простой дорогостоящего скоростного оборудования ЭВМ. Такой режим ввода информации в вычислительную машину называют *пакетным*. Второй способ ввода-вывода сравнительно молод: он стал возможен с появлением мощных машин последних поколений — более быстродействующих, с большим объемом оперативной памяти, чем машины 60-х годов. Внешне этот способ мало отличается от существовавшего на заре века ЭВМ «ручного» ввода: техническим средством общения человека с машиной служат подсоединенные к ЭВМ выносные *Пульты-Терминалы*. Такой Пульт-Терминал может быть удален от машины на большое расстояние. В качестве терминалов сейчас наиболее распространены *Дисплеи* — пульты с телевизионным экраном, на котором легко читать выводимые из машины сообщения и даже графические изображения, и клавиатурой, на

которой набирают запросы к ЭВМ или предписания. В роли терминалов могут выступать также подключаемые к ЭВМ телетайпы и электрические пишущие машинки. Самым важным, пожалуй, является то обстоятельство, что в современных вычислительных машинах число пультов-терминалов выросло до нескольких десятков, а в некоторых системах и сотен, тогда как у машин первых поколений пульт управления был единственным устройством ЭВМ, с помощью которого был возможен оперативный ввод-вывод. У каждого из работающих сегодня за многочисленными терминалами пользователей вычислительной машины создается иллюзия индивидуального общения с ЭВМ: та информация, которую запрашивает каждый из программистов, практически мгновенно высвечивается на его экране, а информация, которую программисты вводят в ЭВМ, сразу же размещается в нужном месте памяти. Такой режим общения с машиной называют *диалоговым*.

## К в а н т в р е м е н и д л я б е с е д ы с Э В М

Для того чтобы человек назвал реакцию машины мгновенной, достаточно, чтобы она отреагировала на его запрос через 0,1 секунды. ЭВМ с быстродействием в  $10^6$  операций в секунду выполнит за это время 100 000 (!) команд. В современных скоростных вычислительных машинах время, выделяемое поочередно каждому пользователю, сокращено до 5 миллисекунд. Но и за такое время ЭВМ может достаточно продвинуть диалог с пользователем: 5000 команд — это немалый фрагмент программы. Именно поэтому у программиста, сидящего перед терминалом, возникает уверенность в том, что машина работает только для него одного. В такой системе распределение машинного времени между пользователями строго регламентировано. За этим следят электронные часы вычислительной машины (так называемый *таймер*), которые входят в состав оборудования Процессора, а также система прерываний. Подобные системы называют системами разделения времени.

## У с т р о й с т в а В в о д а - В ы в о д а

В небольшие ЭВМ информация поступает с устройств *перфоленточного ввода* (рис. 57). В таких устройствах нет сложных электромеханических узлов, а их элек-

тронная часть проста. Вводимую через перфоленточный ввод информацию предварительно кодируют с помощью устройства подготовки информации и переносят на промежуточный информационный носитель — бумажную ленту — в виде комбинаций пробивок. Подобно магнито-

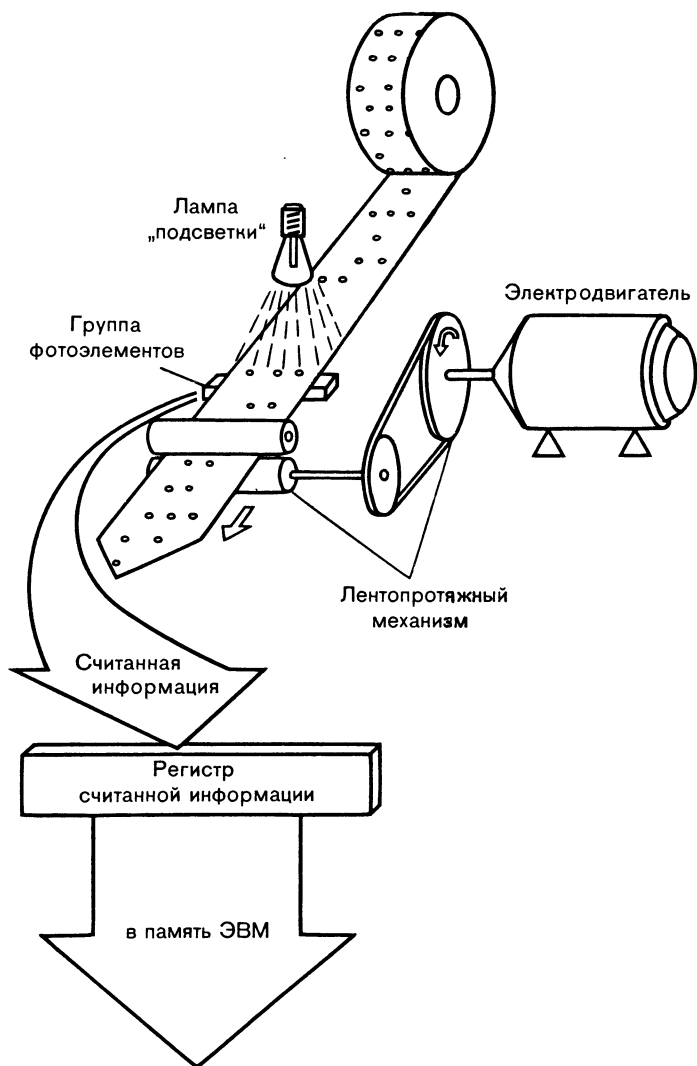


Рис. 57

фону, перфоленточный ввод является устройством последовательного доступа к информации. Движение информационного носителя обеспечивается лентопротяжным механизмом. Отличие от магнитофона состоит в том, что здесь для считывания информации используются фотоэлементы, а не магнитные головки (рис. 58). Различие в освещенности проперфорированных и неразрушенных участков бумажной перфоленты позволяет фотоэлементам легко отличать единицу от нуля.

Часть закодированной на перфоленте информации не пересылается в основную память, а лишь помогает управлять работой самого Устройства Ввода. Например, с помощью *синхронизирующей дорожки* (рис. 57) осуществляются приостановки лентопротяжного механизма в те моменты, когда перед группой фотоэлементов появляется очередная запись — поперечная строка из комбинации пробивок. При этом формируется сигнал, сообщающий Центральному Процессору о готовности

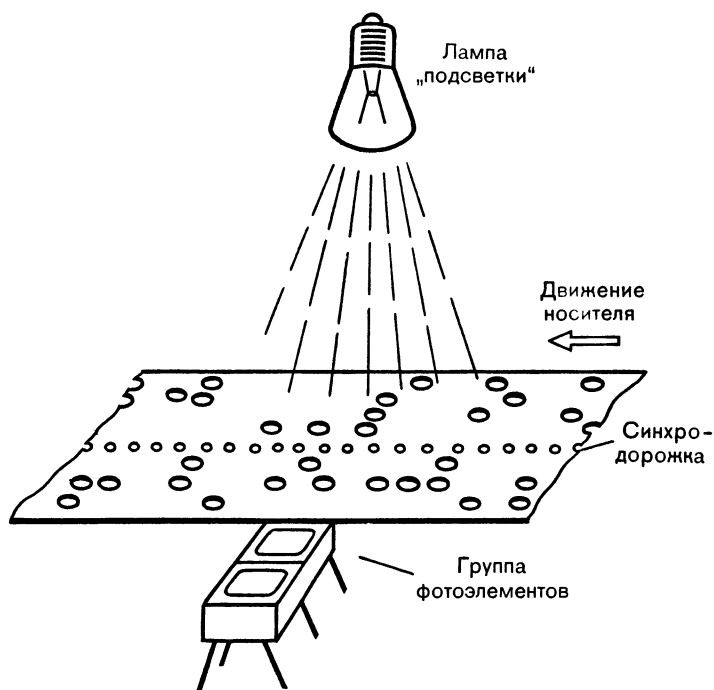


Рис. 58

[illegible]



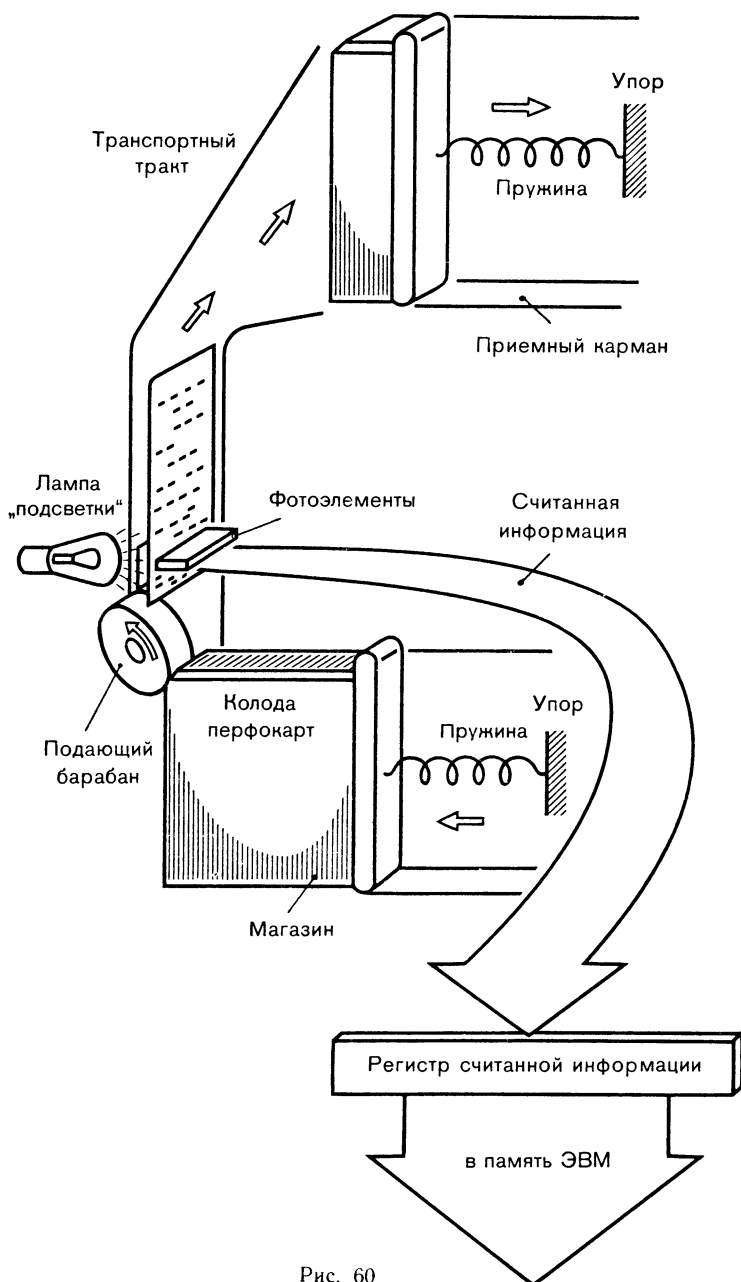


Рис. 60

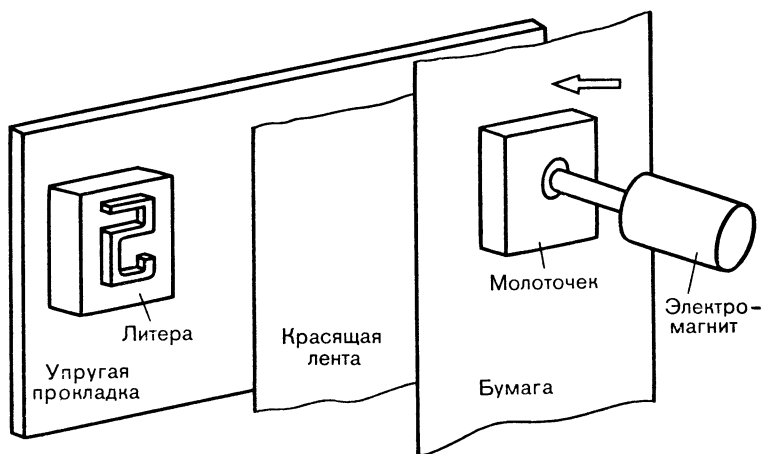


Рис. 61

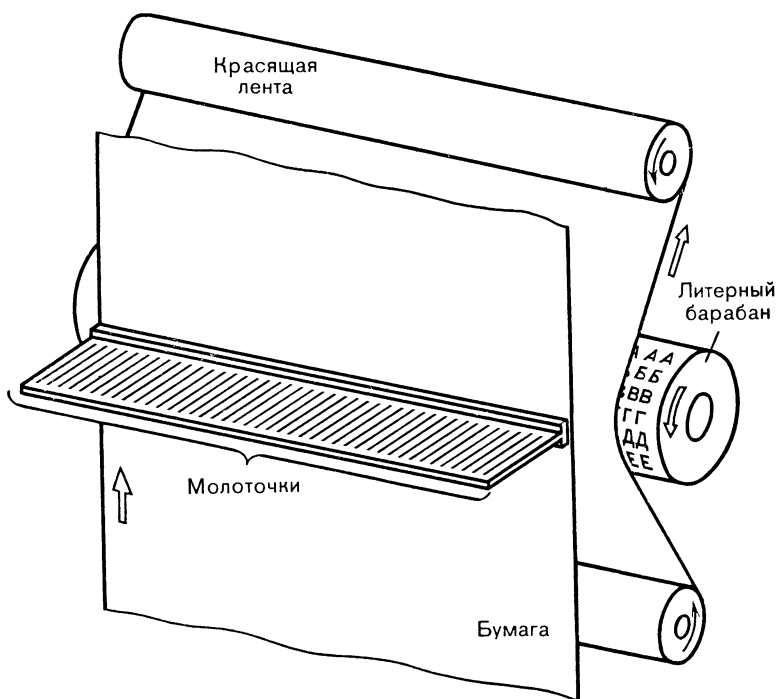


Рис. 62

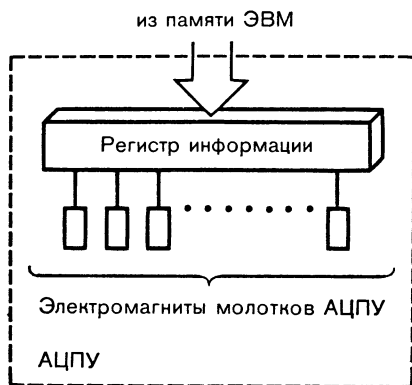


Рис. 63

зуемый Устройствами Ввода, — на перфоленту или на перфокарты. С этой целью к ЭВМ могут быть подключены перфораторы, ленточный или карточный (рис. 64). Техника пробивки в различных перфораторах, учитывающих, конечно, геометрию своего информационного носителя, примерно одинакова. Рассмотренные устройства специализируются на передаче информации только в одном направлении — либо ввод, либо вывод. Наряду с ними существуют устройства, которые способны работать в обоих направлениях. Это уже знакомые читателю Внешние Запоминающие Устройства — накопители информации на магнитной ленте, на магнитных дисках, на магнитном барабане. Хотя в разделе, посвященном памяти, они были занесены в «королевский табель о рангах» запоминающих устройств, это вовсе не мешает включить их в список Устройств Ввода-Вывода. Действительно, такие различные устройства, как магнитофон, перфоленточный ввод и АЦПУ, более объединяются единым способом управления — последовательным доступом к информации, нежели отличаются техническими особенностями.

Способность Внешних Запоминающих Устройств расширять границы основной памяти позволила им прочно закрепиться в составе любой ЭВМ независимо от пакетного или диалогового режима работы машины. Иначе обстоит дело с другими УВВ. Перфоленточный и перфокарточный ввод, перфораторы и АЦПУ составляют традиционный парк устройств машины, работающей в пакетном режиме. Удаленный (дистанционный) доступ к

но и текст самой программы, так называемую распечатку. Распечатки очень нужны программистам при отладке новых программ. Иногда выведенную из ЭВМ информацию требуется впоследствии вновь вводить в машину. В таком случае результаты лучше не печатать на АЦПУ, а выводить непосредственно на информационный носитель, исполь-

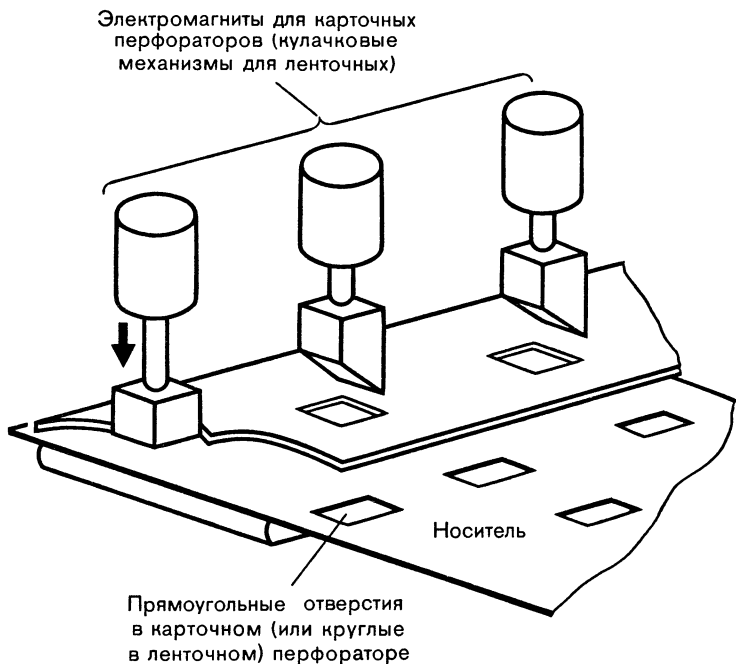


Рис. 64

ЭВМ в диалоговом режиме осуществляется с терминалов. Терминал может служить как приемником, так и источником информации. Таким образом, подобно магнитофону, терминал в полном смысле является Устройством Ввода-Вывода. Простейшим терминалом можно считать связанный с машиной обычный телеграфный аппарат. Быстродействие этого устройства невелико — всего 10 символов в секунду. Понятно, что информационные обмены в этом случае не могут быть большими. Такие устройства используются только для диалога оператора с ЭВМ. Лаконичных сообщений машины достаточно, чтобы оператор смог вовремя сменить катушку магнитной ленты, установить колоду перфокарт на читающем устройстве... — выполнить действия, подсказываемые машиной. Быстродействие УВВ сдерживается инерцией механических узлов. В графическом дисплее такие узлы практически отсутствуют, поэтому такого рода терминалы являются самыми быстродействующими устройствами диалогового режима.

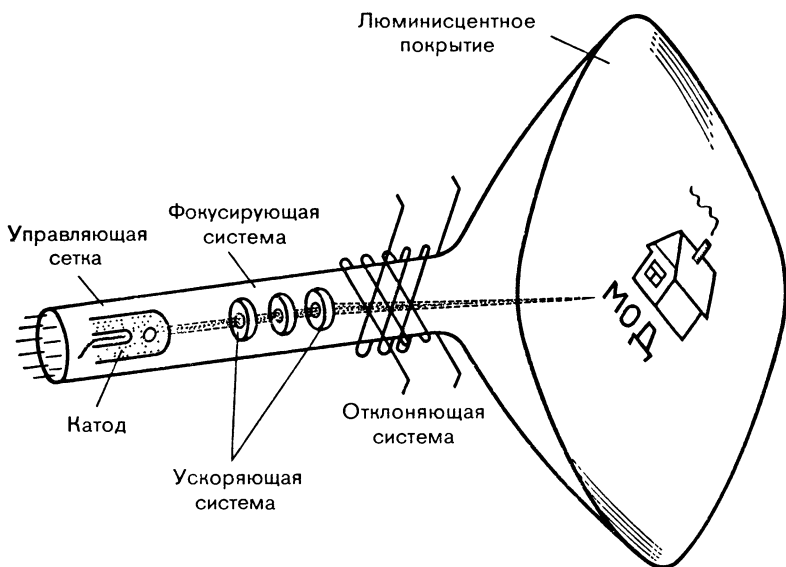


Рис. 65

Главный рабочий узел графического дисплея (рис. 65) — *Электронно-Лучевая Трубка* (ЭЛТ). Она включает в себя: 1) управляющую сетку, с помощью которой регулируют яркость изображения; 2) ускоряющую систему, которая разгоняет поток электронов до скорости, необходимой для свечения люминесцентного покрытия; 3) фокусирующую систему, которая снимает поток электронов в тонкий луч. Широко известны применения ЭЛТ в бытовых телевизорах и в радиолокационных визуальных индикаторах. Работа ЭЛТ основана на двух физических явлениях. Первое — свечение некоторых веществ (люминофоров) при их бомбардировке электронами. Свечение остается видимым некоторое время и после бомбардировки. Второе явление состоит во влиянии электрического поля на траекторию полета электронов. Это позволяет управлять пучком электронов — разгонять электроны до высокой скорости (чтобы их энергии хватило на свечение люминофора), фокусировать пучок до очень тонкого луча, а также перемещать луч для получения изображения на покрытом люминофором экране. ЭЛТ не может достаточно долго сохранять изображение на экране. Время послесвечения измеряется всего лишь десятками миллисекунд. Высвеченная только один раз точка

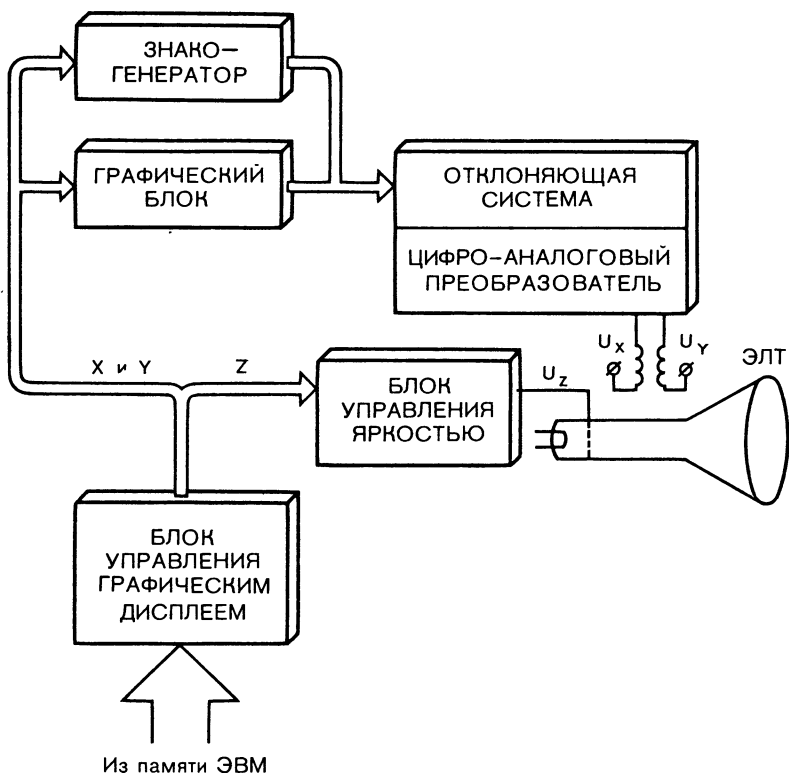


Рис. 66

быстро теряет яркость и гаснет. Этот недостаток ЭЛТ устраняют путем регенерации изображения на экране — многократного повторения высвечивания с частотой 25—30 раз в секунду. Для отклонения луча в заданную точку  $X$ ,  $Y$  на экране ЭЛТ на отклоняющую систему трубки подают в аналоговой форме соответствующие сигналы  $U_x$  и  $U_y$ , пропорциональные задаваемой величине отклонения (рис. 66, на котором изображена индикационная часть графического дисплея).

Яркость свечения тоже может управляться внешним сигналом. Это используется, в частности, для того, чтобы имитировать третье измерение по оси  $Z$  или воспроизводить полутоновые изображения. Последовательность команд ЭВМ для построения рисунка как множества точек на экране ЭЛТ сравнительно проста:

— выработать значения координат каждой точки  $X$ ,  $Y$ , а также значение яркости  $Z$ ; эта информация поступает в графический блок с УВВ или из Центрального Процессора; далее ЦАП преобразует значения  $X$  и  $Y$  в сигналы, управляющие отклонением луча, а значение  $Z$  в сигнал, управляющий яркостью свечения;

— непрерывно повторять это процесс для всех точек рисунка; в этом состоит регенерация, создающая впечатление постоянного изображения на экране ЭЛТ.

Рисунки и чертежи на экране графического дисплея сопровождаются обычно пояснительными надписями, составленными из букв, цифр и специальных знаков. По существу, каждый символ надписи — это рисунок. Коль скоро речь идет о воспроизведении на экране многократно используемых изображений символов, естественно составить программы, управляющие рисованием каждого символа, и затем вызывать эти программы для выполнения. Такие программы определяют множества высвечиваемых точек и, следовательно, изображают требуемые рисунки-символы. При каждом выполнении программы следует указать то место на экране, где должен изображаться соответствующий символ. Координаты начальной точки рисунка, например левого нижнего угла прямоугольника, в который вписывается символ, передаются графическому дисплею из памяти ЭВМ в качестве параметров рисующей программы. Рисование символов на экране ЭЛТ с помощью программы удобно, однако однообразные и многократные обмены перегружают информационный тракт «память ЭВМ — графический дисплей». Более рациональное решение состоит в том, что функции рисования отдельных символов передаются от программ аппаратуре. В графическом дисплее такую задачу решает *знакогенератор* (рис. 66). В состав знакогенератора входит запоминающее устройство, которое хранит управляющую информацию для рисования букв русского и латинского алфавитов, цифр и некоторых специальных знаков. Наличие знакогенератора позволяет передавать из памяти ЭВМ в графический дисплей только код изображаемого символа и координаты начальной точки изображения, а не стандартную программу рисования символа. Для ввода символьной информации графический дисплей располагает обычной клавиатурой, а ввод графической информации можно осуществить, например, *световым пером* (рис. 67). В состав светового пера входят, в частности, оптическая система и фотоэле-

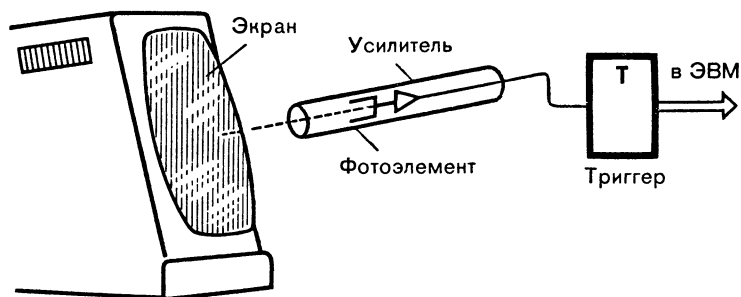


Рис. 67

мент. Оптическая система фиксирует на фотоэлементе световой поток, попадающий в поле зрения пера. Выходной сигнал от фотоэлемента усиливается и подается на вход триггера. Тот будет срабатывать всякий раз, когда световое перо фиксирует достаточно яркий источник света на экране дисплея. Подвод пера к некоторой точке на поверхности экрана возбуждает световое пятно в этой точке. Поскольку именно в эту точку направлен и фотоэлемент пера, то в момент подвода срабатывает цепочка «фотоэлемент — усилитель — триггер». Сигнал, выработанный этой цепочкой, прерывает работу машины. В число сведений о состоянии ЭВМ, регистрируемых и запоминаемых в момент такого прерывания, входят, в частности, координаты точки, к которой было подведено перо. На основании этой информации машина, обрабатывая прерывание, фиксирует указанную точку высвечиванием курсора — яркого пятна в форме квадрата или крестика. Теперь, двигая перо над экраном, можно добиться соответствующего перемещения курсора. Это перемещение оставляет след на экране — светящуюся линию, повторяющую путь светового пера. В дополнение к координатам курсора, автоматически фиксируемым машиной, ЭВМ может получать непосредственно от оператора и другую информацию, определяющую, например, нужно ли провести линию в текущую точку из ранее указанной точки или просто следует высветить отдельно текущую точку. Такая информация может быть введена в ЭВМ либо с помощью кнопок, располагаемых на корпусе светового пера, либо с помощью клавиатуры. Координаты точки, определяемые световым пером, могут быть использованы для программного построения рисунка или его трансфор-



мации. Программы, обслуживающие графический дисплей, могут переносить и поворачивать изображение, изменять масштаб, превращать плоские проекции в аксонометрические и т. д. Рисунок, построенный на экране графического дисплея, можно хранить в памяти ЭВМ. Если же нужна «твердая копия» этого изображения, например, на бумаге, то используется такое устройство вывода графической информации, как *графопостроитель*. Планшетный графопостроитель представляет собой большой стол, по краю которого установлена рельс-рейка. По рейке может перемещаться планка. В свою очередь, по планке перемещается каретка, в которую вставляется пишущий инструмент — перо графопостроителя (или несколько перьев разного цвета). Движение планки вдоль рейки воспроизводит изменение координаты  $X$  пера, а движение каретки вдоль планки — изменение координаты  $Y$ . Существуют и другие конструкции графопостроителей. Например, в барабанном графопостроителе перемещение по координате  $X$  осуществляется вращением цилиндрического барабана, на который накладывается бумага, а по координате  $Y$  — движением каретки с пером по поверхности барабана параллельно его образующей. Работа графопостроителей обеспечивается своим комплексом обслуживающих программ. Впрочем, многие из этих программ совпадают с программными средствами графического дисплея.

### Команды ввода-вывода

Работой УВВ управляют команды ввода-вывода. Они составляют небольшую группу команд из общего числа тех, которые распознает и умеет выполнять вычислительная машина. Команда ввода-вывода осуществляет обмен информацией между машиной и УВВ и содержит все сведения о выполняемом обмене:

- тип (и, быть может, номер) УВВ, участвующего в операции;

- режим работы устройства;

- количество пересылаемой информации;

- размещение информации: адрес (или адресá) в памяти, где находится предназначенная для вывода из ЭВМ информация, или адрес (адресá), по которому должна быть размещена информация, поступающая в ЭВМ с Устройства Ввода. Таким образом, команда ввода-вывода содержит в себе координаты источника и

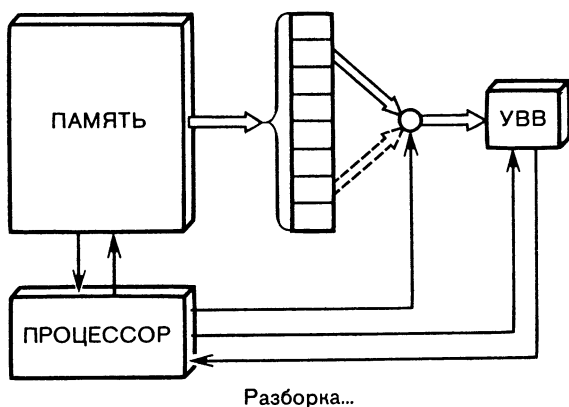
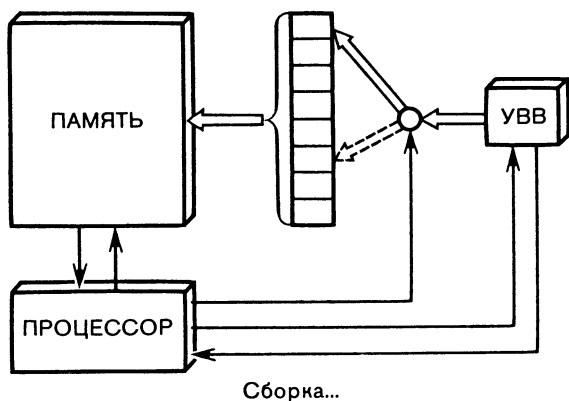


Рис. 68

приемника информации и точные сведения о способе обмена. Команды ввода-вывода в отличие от других команд Процессора не обрабатывают информацию, а осуществляют только транспортные операции. На время выполнения команды ввода-вывода УУ соединяет память информационным трактом с выбранным УВВ (рис. 68). Информационный тракт — это кратковременное оперативное хранилище передаваемой от устройства к устройству информации. Одна из его характеристик — ширина: количество одновременно передаваемой информации. Поскольку ширина информационного тракта от УВВ к памяти часто оказывается меньшей, чем машинное слово,

то подготавливаемую к передаче информацию накапливают в специальных *буферных регистрах* (рис. 68). Так, например, современные ЭВМ серии ЕС допускают обмен информацией с памятью двойными словами (64 бита), а основной обменной порцией в передачах УВВ—память является байт (8 битов). Восемь байтов собирают на буферном 64-битовом регистре, и только после того, как буферный регистр окажется заполненным, запрашивается связь с памятью.

В ранних моделях вычислительных машин в моменты связи УВВ—память — Арифметико-Логическое Устройство (АЛУ) простаивало. В лучшем случае АЛУ занималось подсчетом количества передаваемой информации в байтах или машинных словах. С появлением скоростной памяти паузы, в течение которых происходит сборка буферизации или разборка информации, стали использовать для информационных обменов между памятью и АЛУ. Это значительно повысило эффективность работы ЭВМ—сократилось время решения задач.

С помощью команд ввода-вывода можно управлять и Внешними Запоминающими Устройствами (ВЗУ) — накопителями на магнитных лентах, дисках, барабанах. С этой точки зрения ВЗУ являются Устройствами Ввода-Вывода. Специальные управляющие программы, в состав которых входят команды ввода-вывода, обеспечивают работу иерархии и разнотипных ЗУ как единой памяти ЭВМ, обладающей в целом высоким быстродействием и большей емкостью.

## Процессор ввода - вывода

Предварительное накопление информации в буферных регистрах с последующей сборкой или разборкой оказалось эффективной мерой в основном для медленных УВВ. Положение осложняется, когда Процессору приходится обслуживать такое скоростное УВВ, как накопитель на магнитных дисках. Дело в том, что на Процессор, помимо инициирования (запуска) операций обмена, возложены трудоемкие и многочисленные обязанности, связанные с организацией самого обмена: следить за состоянием буферного регистра, подсчитывать количество переданной информации в байтах или машинных словах, анализировать ситуации, возникающие в процессе обмена с разнотипными УВВ.

До сих пор о Процессоре говорилось как о некоем монархе, которому дано монопольное право интерпретировать все команды Программы. В действительности же Процессор надо сравнивать со втулкой велосипедного колеса: находится он, в самом деле, в центре, но вовсе не он один определяет основную часть функций системы. Процессор, непосредственно интерпретирующий команды Программы, называют *Центральным Процессором (ЦП)*. Для того чтобы освободить ЦП от выполнения команд ввода-вывода, которые могут занимать львиную долю машинного времени в процессе решения задачи, большинство средних и больших вычислительных машин имеют один или несколько специализированных недорогих *процессоров ввода-вывода*. В машинах серии ЕС процессор ввода-вывода называют *каналом*. Поскольку

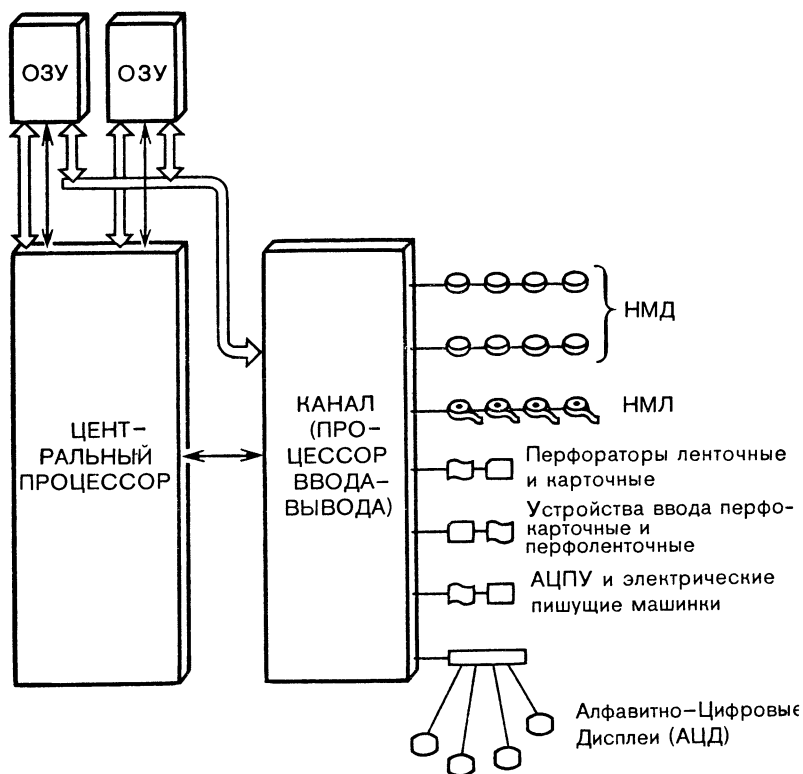


Рис. 69

ввод-вывод осуществляется специальными процессорами ввода-вывода, ЦП получает возможность большую часть времени заниматься собственно обработкой информации, следовательно, быстрее выполнять ее. Процессоры ввода-вывода работают параллельно с ЦП: в то время, когда ЦП занят вычислениями, каналы осуществляют операции ввода и вывода информации. На рисунке 69 показана структурная схема ЭВМ ЕС-1060. По современным представлениям это достаточно мощная машина. Она обладает четырьмя каналами (процессорами ввода-вывода) — один байт-мультиплексный и три блок-мультиплексных. Каналы этих двух типов отличаются друг от друга, прежде всего, своими скоростными характеристиками. У блок-мультиплексного канала пропускная способность доходит до 3 Мбайт ( $3 \cdot 2^{20}$  байт) в секунду; к блок-мультиплексным каналам присоединены скоростные УВВ — накопители на магнитных дисках, магнитофоны. Максимальный информационный поток через байт-мультиплексный канал — 1,5 Мбайт ( $1,5 \cdot 2^{20}$  байт) в секунду. К байт-мультиплексному каналу подсоединяются сравнительно медленные УВВ: дисплеи, электрические пишущие машинки, АЦПУ и устройства, работающие с перфокартами и перфолентами.

### Выполнение команды в канале ЕС ЭВМ

Подобно тому как действия Центрального Процессора определяются управляющей им Программой, работой канала управляет *канальная программа*, предварительно записанная в память. Канальная программа состоит из *канальных команд*, каждая из которых заставляет канал выполнять определенную операцию. Длина канальной команды составляет 64 двоичных разряда — двойное машинное слово в ЕС ЭВМ. Основные поля канальной команды содержат в себе такую управляющую информацию (рис. 70):



Рис. 70

- *код операции* определяет режим работы УВВ; самыми типичными операциями являются «Чтение» и «Запись»;
- *адрес данных* — поле, в котором записан начальный адрес группы машинных слов СОЗУ или ОЗУ, предназначенной для транспортировки через канал к УВВ (или от УВВ);
- *признаки*, с помощью которых уточняются подробности предстоящего информационного обмена;
- *счетчик* — поле, в котором указывается количество транспортируемых байтов.

Система команд Центрального Процессора ЕС ЭВМ также содержит в своем составе команды ввода-вывода, но в отличие от команд ввода-вывода ранних моделей ЭВМ они лишь начинают (инициируют) выполнение операции ввода-вывода. Одна из команд ввода-вывода в ЕС ЭВМ так и называется — Начать Ввод-Вывод (НВВ). О работе системы ввода-вывода в ЭВМ серии ЕС рассказывают следующие стоп-кадры.

#### Кадр 1

Центральный Процессор извлекает команду из памяти (ОЗУ или СОЗУ) и помещает ее в свой Регистр Команд. УУ Центрального Процессора узнает ее: Дешифратор Команд определяет, что речь идет именно о команде НВВ. Затем по адресному полю команды НВВ определяется номер канала и номер УВВ, с которым предстоит начать обмен (рис. 71).

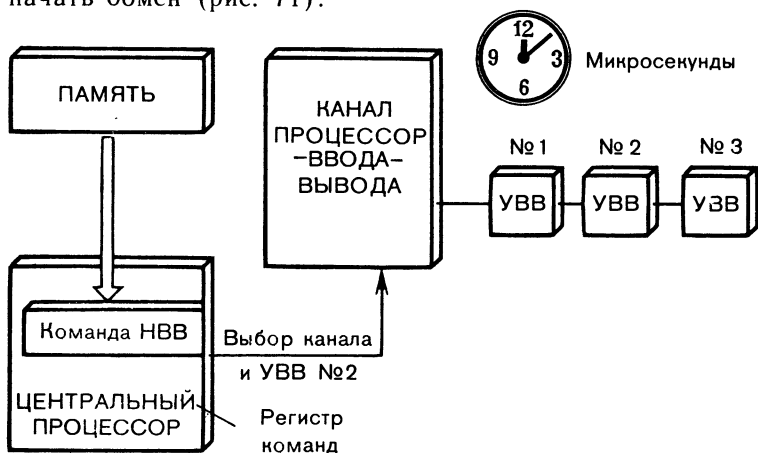


Рис. 71

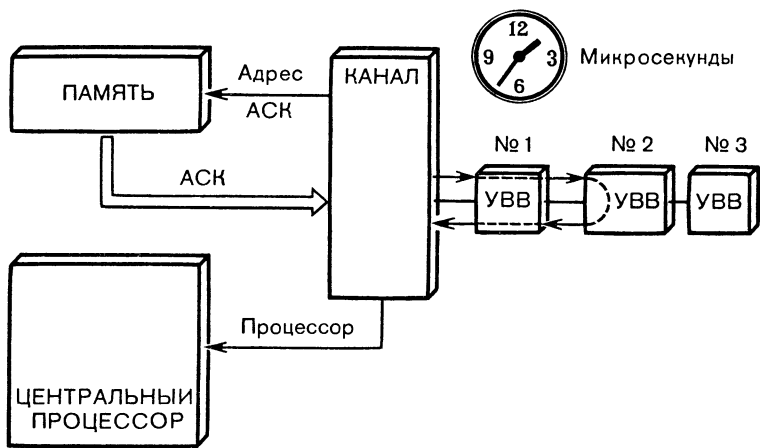


Рис. 72

#### К а д р 2

У каждого из присоединенных к каналу Устройств Ввода-Вывода есть собственный номер. Когда какое-либо из УВВ признает выставленный номер своим, оно подтверждает получение приглашения на связь. В подтверждении указывается, что УВВ с таким номером действительно готово к работе. Выполняя подсоединение или выборку УВВ, канал в то же время обращается в одну из фиксированно выделенных ячеек памяти, где хранится *Адресное Слово Канала* (АСК). АСК пересылается в канал. В АСК, в частности, содержится начальный адрес канальной программы, по которой предстоит работать каналу в процессе информационного обмена (рис. 72).

#### К а д р 3

Канал обращается к памяти по адресу, указанному в АСК, и извлекает первую канальную команду из канальной программы. Впрочем, эта программа может состоять из единственной канальной команды (рис. 73).

#### К а д р 4

Канал начинает расшифровку канальной команды: пересылает разряды 0—7 канальной команды в УВВ, узнавшее свой номер в самом начале соединения Центрального Процессора с каналом. Если УВВ способно

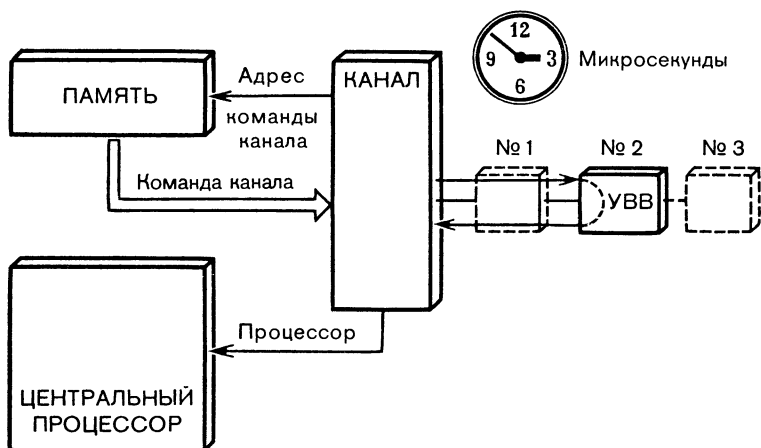


Рис. 73

выполнить предложенную команду, то канал получит от этого устройства соответствующее разрешение на пересылку информации (рис. 74).

### Кадр 5

Если обмен информацией завершен, например исчерпан счетчик последней канальной команды, то канал пре-

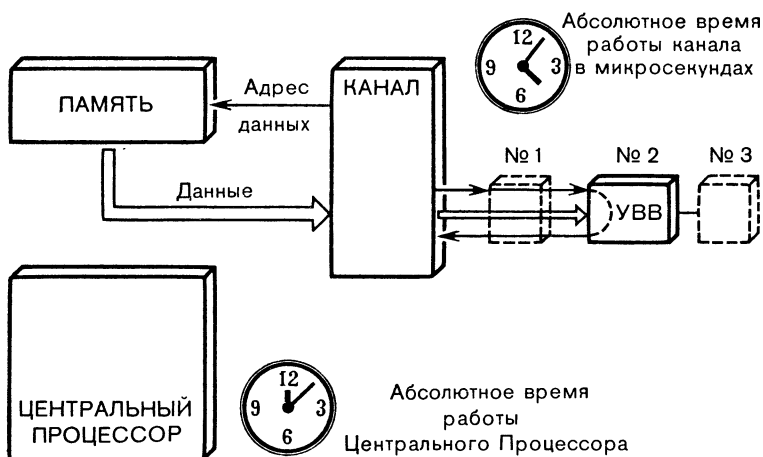


Рис. 74



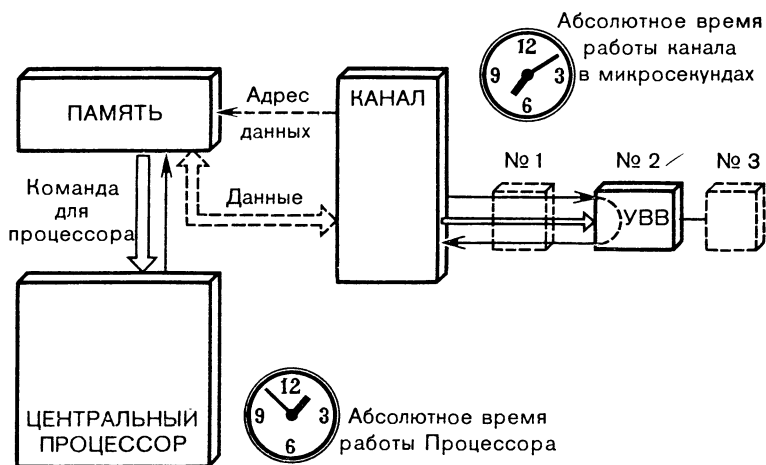


Рис. 75

рывает работу Центрального Процессора. Перед прерыванием канал подготавливает у себя в буферном регистре так называемое *Слово Состояния Канала* (ССК), в котором в закодированной форме сообщает все обстоятельства окончания сеанса связи. Запрос на прерывание поступает в Центральный Процессор, и специальные управляющие программы анализируют причины прерывания и в зависимости от них принимают решение: например, продолжить обмен или отпечатать оператору просьбу сменить катушку магнитной ленты на магнитофоне и т. п. (рис. 75).

После окончания сеанса связи Центральный Процессор «отцепляется» от канала и занимается обработкой другой информации, а канал, загруженный управляющей информацией, самостоятельно работает с памятью, предоставляя время от времени память в распоряжение Центрального Процессора. Ход дальнейшей работы канала, выполнившего каналную команду, определяется полем признаков (рис. 70). При соответствующих кодовых комбинациях признаков связь канала с памятью может быть продолжена: канал может извлечь из памяти следующую каналную команду из каналной программы. Извлечение каналной команды напоминает сюжет знакомой сказки: иголка — в яйце, яйцо — в утке, утка — в зайце... Несмотря на это, эффективность подобной организации ввода-вывода весьма высока.

## ЭКСКУРСИЯ В СИСТЕМУ КОМАНД

**У**мение вычислительной машины выполнять различные операции по обработке информации определяется *системой команд* этой машины.

На первых этапах развития вычислительной техники от программиста требовалось основательное знание системы команд той машины, на которой он работал. Правила общения с ЭВМ, основанные на использовании ее системы команд, составляют *машинный язык*. Долгое время машинный язык оставался единственным средством контакта людей с вычислительными машинами.

С тех пор арсенал человеко-машинного общения существенно пополнился: появились многочисленные языки программирования высокого уровня (о них будет рассказано в разделе «Программное обеспечение»). Однако и сегодня экскурсия в машинную систему команд может оказаться полезной. Во-первых, система команд — это зеркало архитектуры ЭВМ, в котором отражается индивидуальная неповторимость внутренней организации определенного типа машин. Во-вторых, программные средства общения с машиной на языках высокого уровня — программы-переводчики, называемые *трансляторами*, — создаются чаще всего из машинных команд. В-третьих, в некоторых отдельных случаях, когда программист стремится использовать все резервы производительности ЭВМ для наиболее эффективного решения какой-нибудь очень часто повторяющейся задачи, он обращается к машинному языку. Итак, элементом системы команд является команда, предписывающая машине выполнение той или иной операции. *Команда* — это предложение на машинном языке. Части этого предложения, порядок их следования и размеры строго определены: они задаются форматом команды. В формате команды выделяются *код операции* и *адресная часть*. О коде операции говорилось уже в разделе «Как работает Процессор?». Назначение адресной части — указать адреса *операндов*, т. е. величин, участвующих в операции. На рисунке 76 адресная часть команды разделена на три поля с адресами операндов. В первом адресном поле находится адрес первого операнда выполняемой операции, во втором — адрес второго операнда. Третье адресное поле указывает адрес ячейки памяти, куда надо записать результат операции. Все три адреса имеют одинаковую длину. Такой формат команд имела не дожив-

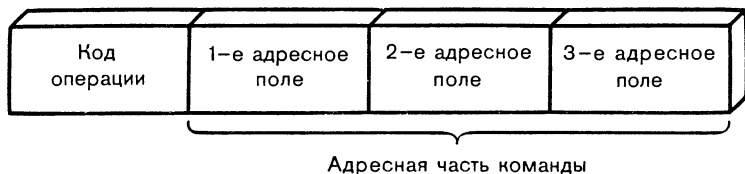


Рис. 76

шая до наших дней *трехадресная машина* БЭСМ-2. Например, команда 01 0257 0643 0016 в БЭСМ-2 означала сложение (код операции — 01) величины, хранящейся в ячейке памяти (ОЗУ) по адресу 0257, с величиной, находящейся в ячейке по адресу 0643, и размещение результата операции в ячейке с адресом 0016. Когда в адресном поле команды записывается адрес машинного слова, находящегося в памяти (как в случае БЭСМ-2), длина адресного поля определяется объемом памяти с произвольным доступом — ОЗУ. Например, для того чтобы  $4K = 2^{12} = 4096$  машинных слов могли храниться в ОЗУ в ячейках с различающимися адресами, длина адресного поля должна составлять не менее 12 двоичных разрядов. Иной вид имеет команда *одноадресной машины* (рис. 77).

В адресном поле такой команды можно указать только один адрес. Как же в таком случае задать второй операнд и куда послать результат? Отвечая на эти вопросы, отметим, что в работе Центрального Процессора одноадресной машины важная роль отведена накапливающим регистрам. Практически каждая команда использует накапливающие регистры либо в качестве источника или приемника информации при обменах с памятью, либо для хранения операндов перед выполнением операции. Например, следуя системе команд машины СМ-2, сложение двух чисел, хранящихся в памяти по адресам 0257 и 0643 с засылкой результата в ячейку с адресом 0016, выполняется такими тремя командами:

06 3 257 — в регистр-накопитель принимается число из ОЗУ с адресом 257;

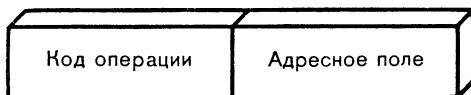


Рис. 77

04 3 643— содержимое регистра-накопителя складывается с содержимым ячейки памяти, имеющей адрес 643;

07 3 016— результат операции, оставшийся в регистре-накопителе после выполнения предыдущей команды, пересылается в ячейку памяти с адресом 16.

Не следует думать, что программа одной и той же задачи в одноадресной машине всегда вдвое длиннее, чем в трехадресной. В большинстве практических случаев (особенно, когда результат операции используется в непосредственно следующей операции) удлинение программы оказывается небольшим. Вот пример. В кодах трехадресной машины программа, вычисляющая выражение  $a^2 + a - 1$ , записывается так:

03	A	A	B	умножение	$a \cdot a$
01	A	B	B	сложение	$a^2 + a$
02	B	C	B	вычитание	$(a^2 + a) - 1$ ,

где A — это адрес числа  $a$ , по адресу C хранится единица, B — адрес ячейки памяти, используемой для хранения промежуточных результатов. В системе команд еще одного «прадедушки» современных ЭВМ — одноадресной машины «Урал-2» — та же задача решается следующей программой:

0,2<sup>1)</sup> A число  $a$  пересылается из ОЗУ (из ячейки с адресом A) в регистр-накопитель Центрального Процессора (в «Урале-2» был единственный процессор);

06 A умножение  $a \cdot a$  — перемножаются содержимое регистра-накопителя и значение, хранящееся в ячейке с адресом A; произведение остается в регистре-накопителе;

01 A сложение числа из ячейки A с содержимым регистра-накопителя; результат  $a^2 + a$  остается в регистре-накопителе;

03 C из содержимого регистра-накопителя (значение выражения  $a^2 + a$ ) вычитается содержимое ячейки C (единица); результат операции — значение выражения  $a^2 + a - 1$  — остается в регистре-накопителе.

Итак, программа увеличилась всего на одну команду (обратили ли Вы внимание на то, что в последней

---

<sup>1)</sup> Как видно, один и тот же код, например 02, в разных машинах может соответствовать разным операциям.

программе не понадобилась ячейка ОЗУ для хранения промежуточного результата?). А вот экономия оборудования в одноадресной машине значительная: ведь адресная часть каждой команды становится втрое короче; следовательно, каждая ячейка памяти будет короче — потребуется меньшее количество триггеров в СОЗУ и меньше ферритовых колец в ОЗУ. Вряд ли после этого читатель найдет удивительной *двухадресную машину* (такими были ЭВМ серии «Минск»). Зато полутораадресная машина может показаться на первый взгляд необычной. Однако довольно часто ЭВМ располагает двумя типами памяти — обычным Оперативным Запоминающим Устройством (ОЗУ) и Сверх Оперативным Запоминающим Устройством (СОЗУ). Объем ОЗУ, измеряемый в количестве ячеек памяти, весьма велик, поэтому для нумерации ячеек ОЗУ требуются большие, «длинные» числа — «длинные» адреса. СОЗУ состоит из очень небольшого числа регистров, которые можно пронумеровать «короткими» адресами. «Длинный» адрес ячейки ОЗУ и «короткий» регистра СОЗУ, используемые в одной машинной команде, представляют адресную часть команды «полутораадресной» машины. Впрочем, в более поздних машинах регистры-накопители нашли и другое применение — для определения адреса операнда. Дело в том, что в большинстве машин адрес, участвующий в операции непосредственно в момент выполнения команды (так называемый *исполнительный адрес*), не совпадает со значением адресного поля команды, а вычисляется как значение некоторой функции. Такая функция чаще всего выглядит как сумма записанного в адресном поле значения (так называемого *смещения*) и содержания одного из регистров-накопителей (*регистра базы*), указанного в команде, или двух регистров (*базы и индекса*). Например, в машине БЭСМ-6 (если отвлечься от некоторых непринципиальных подробностей) формат большинства команд имеет вид, показанный на рисунке 78.

Номер регистра базы 4 разряда	Код операции 8 разрядов	Адресное поле /смещение/ 12 разрядов
-------------------------------------	----------------------------	--

Рис. 78

Числа в полях команды указывают длину соответствующего поля, измеренную в количестве двоичных разрядов. ОЗУ машины БЭСМ-6 содержит  $32\text{ К} = 2^{15}$  ячеек, а наибольшее число, записываемое в адресное поле, во всяком случае меньше, чем  $2^{13}$ . И вот тут на помощь приходит один из шестнадцати ( $2^4 = 16$ ) регистров базы, именно тот, номер-адрес которого указан в поле регистра базы в команде. Исполнительным адресом команды является сумма 15-разрядного значения, хранящегося в указанном регистре базы, и 12-разрядного значения смещения. Например, при условии, что в третьем регистре содержится число 12 000, команда 03 004 0231 будет складывать (004 — код операции сложения) содержимое специального накапливающего регистра — *сумматора* — с числом, расположенным по адресу  $12\ 000 + 231 = 12\ 231$ , и оставит результат в сумматоре. Здесь 3 — номер регистра базы, 12 000 — база, 231 — смещение, 12 231 — исполнительный адрес. Таким образом, база обозначает начальный адрес некоторой области памяти, а адрес нужного слова в этой области получается добавлением смещения к базе. Этот способ адресации называется *относительным*. Он удобен тем, что дает возможность перемещать программы или части программ по всей адресуемой памяти (ОЗУ), а это позволяет освободить программиста от необходимости следить за размещением программы. Еще большее развитие способ относительной адресации получил в машинах серии ЕС. Регистровая память машин этой серии состоит из шестнадцати регистров общего назначения. В адресной части команды записываются чаще всего два адресных поля. Поэтому в этих машинах существует не один, а три формата команд. Первый из них «регистр-регистр» (рис. 79) описывает команды, которые оперируют со значениями, хранящимися в регистрах  $R1$  и  $R2$ . Результат операции остается в регистре  $R1$ . Второй формат — «регистр-память»: один из операндов извлекается из регистровой памяти, а второй — из ОЗУ. Адрес первого операнда, располагающегося в регистровой памяти, задан полем  $R1$ , а исполнительный адрес второго операнда, который хранится в ОЗУ, получается сложением смещения со значениями, находящимися в регистре базы (поле  $B2$ ) и регистре индекса (поле  $I2$ ). Наконец, в командах фор-



Рис. 79



Рис. 80

мата «память-память» оба операнда расположены в ОЗУ (рис. 80). Для каждого из них исполнительный адрес получается сложением смещения со значением регистра базы. Кроме того (в этом особенность машин серии ЕС), в этом формате указываются длины Д1 и Д2 полей операндов, что дает возможность работать не только с машинными словами единого размера, но и с величинами произвольной длины. Сравнивая различные системы адресации, можно отметить, что в трехадресной машине все ячейки ОЗУ совершенно равноправны с точки зрения организации и времени доступа к памяти. Машины, обладающие таким свойством, называются *линейными* или, точнее, машинами с линейной структурой памяти. Уже в обычной одноадресной машине линейность слегка нарушена: существует один особый регистр-накопитель, который может хранить информацию подобно другим ячейкам памяти, но весьма отличается от них по способу доступа. В частности, этот регистр — единственный участок памяти, куда результат может попасть без предварительных пересылок. В машинах с регистрами базы линейность нарушается неравноправием между небольшим количеством регистров (16 в БЭСМ-6) и емким ОЗУ (32 К в БЭСМ-6). Еще более заметна нелинейность памяти в машинах серии ЕС, где такое неравноправие определяется не только наличием базы и индекса в формате команды, но также и возможностью работы со словами различной длины. Вместе с тем все эти машины в принципе сохраняют линейную структуру основной части ОЗУ. Принципу линейности был нанесен серьезный удар в машинах со *стековой* структурой памяти. Понять, что такое *стек*, легче всего, представив себе стопку книг на Вашем письменном столе (английское слово *stack* как раз и означает «стопка»). Добавить книгу в стопку можно, лишь положив последнюю интересовавшую Вас книгу на самый верх стопки. Эта же книга, положенная последней, оказывается и самой первой из доступных книг. Даже если Вам нужна какая-то книга

в середине стопки, то первой книгой, которую придется брать в ходе этого поиска, будет последняя книга, положенная на верх стопки. Стек — это такая группа машинных слов, в которой в каждый момент доступно только одно слово, а именно то, которое было записано последним. Это единственное доступное слово в стеке называется *верхушкой стека*, а вся остальная часть стека — его *телом*. Добавление слова в стек означает создание новой верхушки; предыдущая верхушка при этом становится вторым элементом стека, частью тела. Выборка слова из стека делает верхушкой стека элемент, который до выполнения этой операции следовал в стеке непосредственно за верхушкой. Если память ЭВМ организована в виде стека, то для выполнения многих операций можно не указывать адреса операндов, если они предварительно помещены в верхушку стека или непосредственно следом за ней. Так, команда «Сложить», задаваемая только кодом операции (и ничем более!), складывает два числа, одно из которых находится в верхушке стека, а другое непосредственно вслед за ним, и результат помещается в верхушке стека. Как видите, в команде нет совсем адресной части. Машины со стековой структурой памяти поэтому часто называют *безадресными*. Такой термин не означает, конечно, что машинные слова в стековой памяти не имеют адресов. Адреса существуют, но после того, как операнды помещены в стек, нет необходимости указывать эти адреса в адресной части большинства команд. Вот как выглядит, например, программа, вычисляющая значение упоминавшегося уже выражения  $a^2 + a - 1$  в ЭВМ «Эльбрус» — самой быстродействующей советской машине четвертого поколения (А — адрес величины  $a$ ):

- |         |   |
|---------|---|
| ЗГЦ1    | загрузка в верхушку стека целого числа 1 (рис. 81, а);  |
| СЧВЕЛ А | считывание величины, расположенной по адресу А, в верхушку стека; прежняя верхушка погружается при этом в тело стека (рис. 81, б);  |
| РАЗДВ   | «раздвоение» верхушки; значение, хранившееся в верхушке стека, записывается теперь и в ячейку, непосредственно следующую за верхушкой; остальная часть стека опускается при этом еще глубже (рис. 81, в); |
| РАЗДВ   | еще одно «раздвоение»: дело в том, что  |



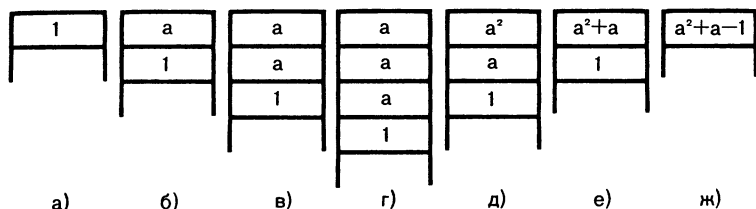


Рис. 81

- УМН      в дальнейших операциях потребуются три экземпляра величины  $a$  (рис. 81, г); перемножение чисел, расположенных в двух самых верхних ячейках стека; результат  $a^2$  остается в верхушке, а использованные операнды исчезают из стека (рис. 81, д);
- СЛ      сложение чисел, расположенных в двух верхних ячейках стека; результат  $a^2 + a$  остается в верхушке (рис. 81, е);
- ВЫЧ      вычитание: из верхушки стека — суммы  $a^2 + a$  — вычитается единица, расположенная в стеке сразу под верхушкой; результат остается в верхушке стека (рис. 81, ж).

В вычислительных машинах со стековой организацией памяти команды становятся существенно короче: ведь большая часть команд — безадресные команды. Однако главное преимущество стековой памяти состоит в создании удобств для программистов, разрабатывающих совершенные программные средства общения человека с машиной — трансляторы с языков высокого уровня. Дело в том, что стек является необходимым механизмом каждого транслятора, так что в машинах, не содержащих стековой памяти, трансляторы используют регистры для моделирования стека программными средствами. Это приводит к потерям производительности «регистровой» ЭВМ по сравнению с безадресной, стековой вычислительной машиной.

## НЕКОТОРЫЕ ПОДРОБНОСТИ О СИСТЕМЕ КОМАНД ЕС ЭВМ

**С**истема команд машин серии ЕС весьма обширна. Она содержит 174 различные по своим функциям команды, имеющие коды операций от  $00000100_2$  до  $11111101_2$ . Это многообразие команд можно распределить на несколько групп. Команды транспортировки информации, выполняющие пересылку информации из одного места памяти в другое, относят к числу наиболее простых. Однако без команд транспортировки информации не обойдется ни одна программа. Поэтому с таким же успехом команды этой группы можно назвать важнейшими. Наиболее простая команда пересылки в ЕС ЭВМ — «Загрузить регистр  $R1$  содержимым регистра  $R2$ ». Эта короткая двухбайтовая команда типа «регистр-регистр» снимает копию с оригинала — значения, хранящегося в регистре  $R2$ , и пересылает ее в регистр  $R1$ . Старое значение  $R1$ , хранившееся там до выполнения этой операции, исчезает, оно заменяется копией  $R2$ . Оригинал, конечно, остается неизменным в  $R2$ . На рисунке 82 приводится пример пересылки из четвертого регистра в десятый.

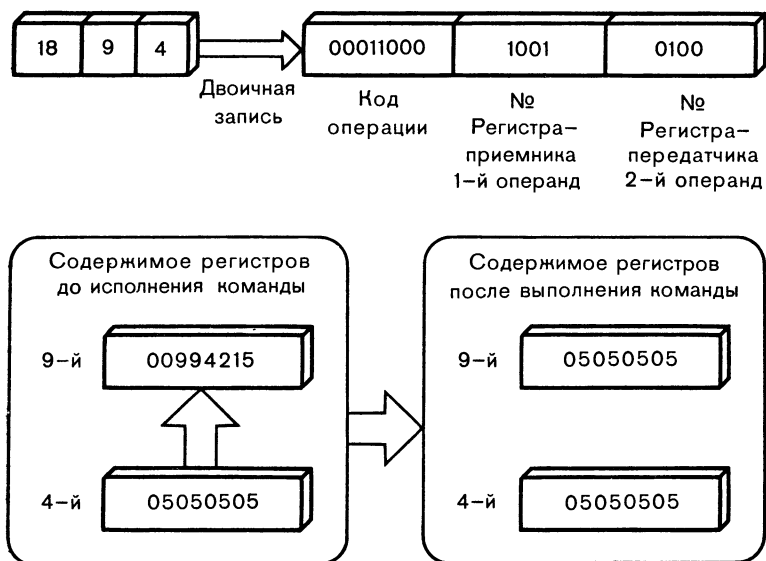


Рис. 82

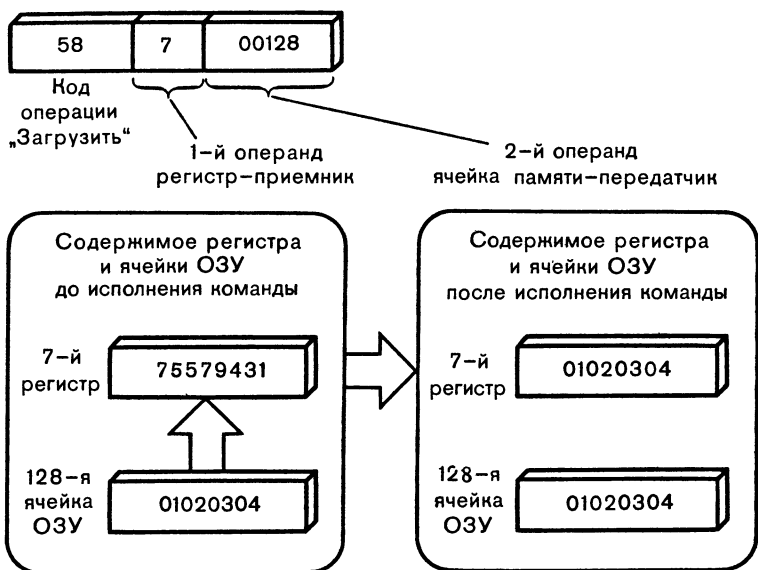


Рис. 83

Обратную пересылку — из девятого регистра в четвертый — при необходимости можно сделать командой с тем же кодом. Достаточно лишь поменять адреса регистров в команде местами, заменив передатчик информации приемником: 18 4 9.

Такой «симметричностью» не обладают четырехбайтовые команды пересылки типа «память-регистр». Команда, пересылающая информацию из ОЗУ в регистр, называется «Загрузить регистр  $R_1$  (содержимым ячейки памяти)». Она имеет код  $58_{16}$ . Пример работы команды, загружающей седьмой регистр значением, которое хранит 128-я ячейка памяти (ОЗУ), приведен на рисунке 83. Обратные пересылки из регистровой памяти в ОЗУ выполняет команда с кодом  $50_{16}$ : «Записать (в ячейку ОЗУ из регистра)». В примере, приведенном на рисунке 84, содержимое третьего регистра пересылается в 700-ю ячейку. В этих и других приводимых здесь примерах команд ЕС ЭВМ база и смещение для простоты берутся нулевыми.

Большую группу команд составляют команды машинной арифметики, непосредственно выполняющие арифметические операции. Каждая из них имеет короткую

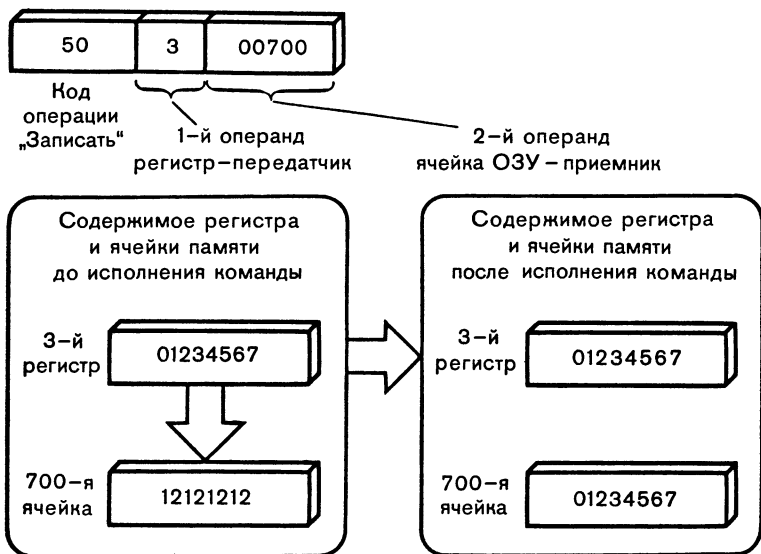


Рис. 84

(«регистр-регистр»), среднюю («регистр-память») и длинную («память-память») модификации. Короткая двухбайтовая команда «Сложить содержимое двух регистров» имеет код операции  $1A_{16} = 00011010_2$ . Ее ближайшая «родственница» — средняя, четырехбайтовая команда сложения — код, отличающийся всего на один бит:  $5A_{16} = 01011010_2$ .

Например, содержимое седьмого и четвертого регистров можно сложить, используя команду с кодом операции  $1A_{16}$ . В качестве операндов указываются регистры 7 и 4, результат операции будет записан в седьмой регистр. Таким образом, одно из слагаемых к концу выполнения команды заменяется суммой (рис. 85). Механизм четырехбайтовой команды немного отличается от предыдущего примера: второе слагаемое извлекается из ячейки ОЗУ. Например, сложение числа из регистра 9 с числом из 144-й ячейки ОЗУ выглядит так, как показано на рисунке 86. Результат выполнения, как и в случае короткой команды, будет записан по первому адресу — в девятый регистр. В некоторых командах код операции может неявно задавать расположение операндов. Так, например, короткая команда «Разделить (содержимое

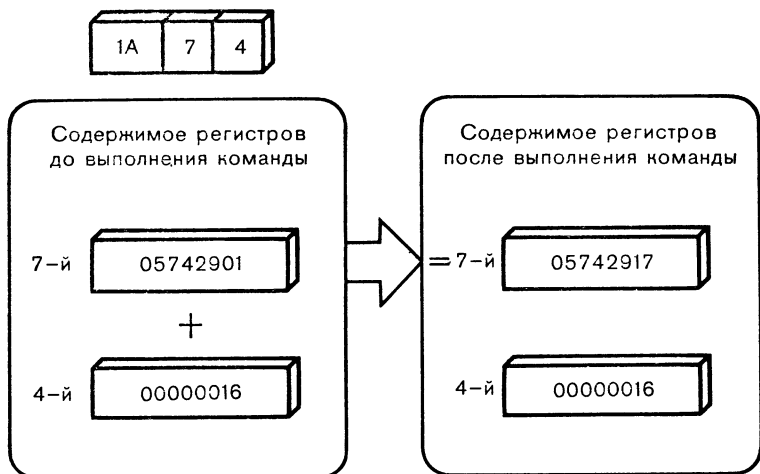


Рис. 85

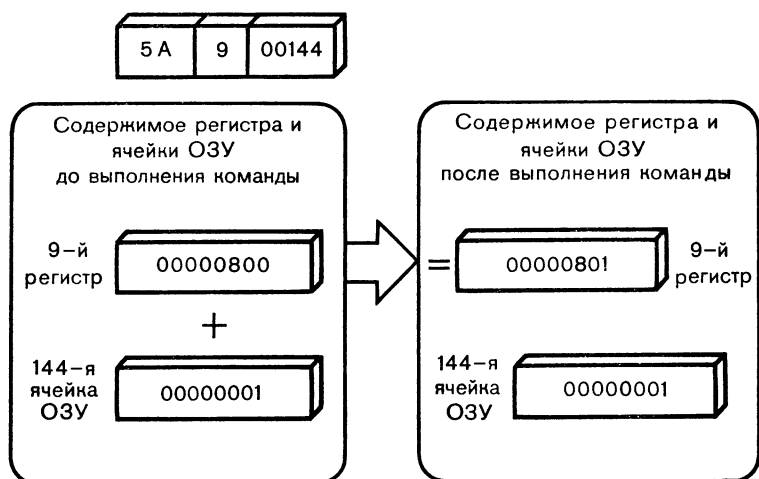


Рис. 86

регистра  $R1$  на содержимое регистра  $R2$ )» предусматривает, что делитель располагается в регистре  $R2$ . Делимое в этой операции может быть двойным машинным словом, и для него отводятся два регистра — регистр  $R1$  и расположенный за ним регистр  $R1 + 1$ . Перед нача-

лом такой операции старшие разряды делимого должны быть размещены в  $R1$ , а младшие его разряды — в  $R1 + 1$ . В этих же двух регистрах-соседях, из которых в команде, как мы видели, указывается только один, получается и результат: в регистре  $R1$  — остаток от деления, а в регистре  $R1 + 1$  — частное (рис. 87). Во всех рассмотренных примерах содержащиеся в регистрах и ячейках ОЗУ значения представлены в шестнадцатеричной системе. Тех нескольких команд, с которыми мы уже познакомились, достаточно для того, чтобы написать на машинном языке ЕС ЭВМ — в ее системе команд — простейшую программу, вычисляющую, например, среднее арифметическое двух чисел.

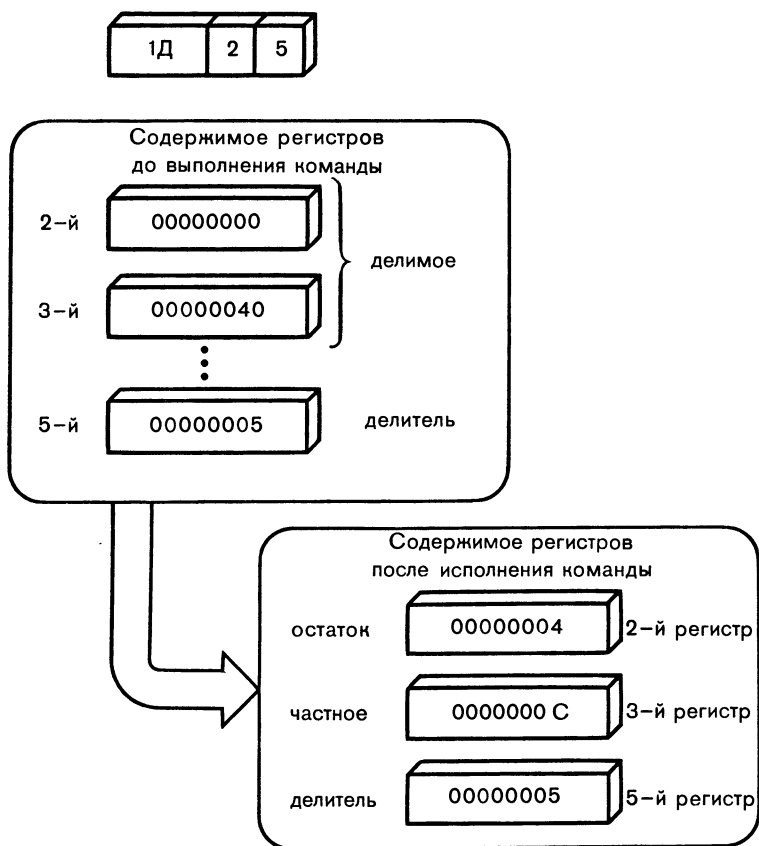


Рис. 87

Составление всякой программы начинают с *распределения памяти*. Так называют соответствие, устанавливаемое между программой и используемыми ею величинами, с одной стороны, и местом в памяти (адресами ячеек, отведенных для хранения этих величин и программы) — с другой. Пусть два интересующих нас слагаемых находятся в ячейках ОЗУ с номерами 200 и 630, а число 2 — знаменатель среднего арифметического — в ячейке 700. Если программу начать в ячейке 750, то она будет иметь такой вид:

<i>адрес команды</i>	<i>команда</i>	
750	58 4 00200	} пересылка значений из ОЗУ в регистры
754	58 9 00630	
758	58 7 00700	
75С	1А 4 9	вычисление суммы
75Е	18 3 4	подготовка к делению
760	1D 2 7	деление

В результате выполнения этой программы среднее арифметическое окажется в третьем регистре, а остаток от деления будет помещен во второй регистр.

Обратите внимание: Счетчик Адреса Команд Центрального Процессора в первых трех командах изменяется на 4 — длину четырехбайтовой команды, а в последующих командах — на длину двухбайтовой команды. Особое место в системе команд занимают команды передач управления. В только что рассмотренной программе команды выполняются последовательно, одна за другой. Такой порядок выполнения программы обеспечивается тем, что к содержимому счетчика адреса команд ЦП после выполнения каждой команды прибавляется длина этой команды, выраженная в байтах или машинных словах. Таким образом, в счетчике происходит автоматическое формирование адреса следующей команды. Теперь ясно, как поступить в случае, если возникнет необходимость выбрать для выполнения не следующую, а какую-либо другую команду, расположенную в произвольном месте памяти. Надо в нужный момент загрузить счетчик адреса команд адресом требуемой команды. Именно такую операцию выполняет команда безусловной передачи управления, или *безусловного перехода*. В адресной части этой команды указывается номер регистра, содержимое которого команда

посылает в счетчик адреса команд ЦП. Номер регистра занимает в команде безусловного перехода только половину (вторую половину) адресной части — одну тетраду младших разрядов. Первая же половина занята специальным признаком. Дело в том, что код  $07_{16}$ , который имеет команда безусловного перехода, отнесен ко всей группе команд передач управления, имеющих тип «регистр-регистр». Значение признака  $F_{16}$  означает, что речь идет именно о команде безусловного перехода, принадлежащей к этой группе. Пусть, например, команда безусловного перехода расположена в ячейке ОЗУ с адресом 1102. Следующей будет выполняться, однако, не команда из ячейки 1104, а команда, находящаяся по адресу 400: именно такое значение содержит третий регистр, указанный в адресной части команды безусловного перехода. Загрузив предварительно регистр нужным значением, можно, таким образом, передать управление на любую из команд программы. К такому приему достаточно часто прибегают при составлении программ на машинном языке. И все же по сравнению с командами безусловного перехода, схожими по своей категоричности с дорожными указателями на улицах с односторонним движением, гораздо более интересны команды условной передачи управления. Команды условной передачи управления, или *условного перехода*, позволяют осуществлять разветвления в программе: продолжать дальнейшее выполнение программы



Рис. 88



по одному из двух возможных путей. Но в отличие от древнего витязя, который, находясь на перепутье, выбирает одну из дорог случайно, команда условного перехода опирается в своем выборе на предысторию выполнения программы (рис. 88).

В Центральном Процессоре есть еще один не рассмотренный пока специальный регистр — *регистр признака результата*. Он невелик по размеру, его длина всего один двоичный разряд. Каждая из команд машины в дополнение к своим основным функциям оставляет на регистре признака результата некоторую качественную характеристику результата выполненной операции. Так, все модификации команды сложения посылают в регистр признака результата значение 0, если сумма оказывается положительной, и 1, если результат отрицателен или равен нулю. Другие операции могут трактовать значения кода в регистре признака результата по-иному. Например, после окончания команды «Сравнить два числа» регистр признака результата получает значение 0 при равенстве сравниваемых операндов и значение 1, если числа-операнды не равны. Команда условного перехода использует то значение признака результата, которое было сформировано последней выполненной командой. Если значение регистра признака результата равно нулю, то следующей выполняется команда, адрес которой указывается регистром адресной части команды условного перехода. Если же регистр признака результата равен единице, то следующей выполняется команда, расположенная непосредственно вслед за командой условного перехода (к счету команд прибавляется единица).

## СЛОВО СОСТОЯНИЯ ПРОГРАММЫ

**С**овременные вычислительные машины предназначены для многопрограммной (мультипрограммной) работы. Вспоминая аналогию с заводом, можно сказать, что изготавливаются несколько разных изделий по разным технологическим планам. Чтобы детали самолета не попали в цех, где собирают детские велосипеды, необходимо организовать контроль выдачи со склада: кому и что выдавать.

Содержимое памяти в ЭВМ охраняется ключами. Представьте себе длинный коридор склада (памяти). Каждая комната, выходящая в коридор, имеет свой замок, а следовательно, свой ключ. Для каждой ЭВМ

определено понятие такой «комнаты» — блока охраняемой памяти стандартного размера. В машинах ЕС размер такого блока составляет 2 Кбайт (2048 байт) (рис. 89). Длина ключа — четыре бита. Значит, существуют  $2^4 = 16$  вариантов ключей. Каждому блоку памяти объемом 2 Кбайт соответствует единственный четырехразрядный регистр. Когда программе выделяется



Рис. 89

память, то всем блокам памяти, в которых размещается эта программа, присваивается общий ключ. Это выполняет специальная транспортная команда: она переносит кодовую комбинацию, заранее подготовленную в одном из 16 регистров общего назначения, в регистр ключа, принадлежащий адресуемому двухкилобайтовому блоку. Для 16 Мбайт ( $2^{24}$  байт) адресуемой памяти общее число регистров ключей равно 8192. Они выделены в отдельное устройство — память ключей защиты. Это устройство входит в состав ЦП.

При обсуждении работы ЦП в разделе «Как работает Процессор?» для иллюстрации были использованы стоп-кадры событий. В каждый момент работы Центрального Процессора его состояние полностью определяется состояниями Счетчика Адреса Команд, Регистра Результата, Регистра Ключа Памяти... Состояния всех этих регистров, своеобразный «семейный портрет» элементов Центрального Процессора, хранятся в *регистре Слова Состояния Программы* (ССП). В ССП можно прочитать и содержимое регистра признака результата, и значение регистра ключа памяти: именно это значение будет сравниваться с содержимым соответствующего регистра из памяти ключей защиты. Кроме этого, ССП содержит управляющую информацию о том, может ли какая-нибудь причина прервать Центральный Процессор во время выполнения программы. Эта информация определена значением поля ССП, именуемого *маской*. И еще одно поле выделено в ССП — *поле кода прерывания*. В нем указывается код причины, вызвавшей прерывание (если оно было разрешено полем маски). Например, в поле кода прерывания может стоять тип и номер УВВ, прервавшего работу ЦП (рис. 90). Счетчик Адреса Команд, Регистр Результата и некоторые другие поля регистра ССП постоянно модифицируются: после выполнения каждой команды они принимают новые значения.



Рис. 90

С помощью команды «Загрузка ССП» можно изменить содержимое регистра ССП. Состояние ЦП после загрузки нового ССП может измениться более существенно, нежели при выполнении команд передачи управления: ЦП может перейти к выполнению другой программы или изменить отношения с системой прерываний. Команда «Загрузка ССП» тоже относится к командам управления.

## СИСТЕМА ПРЕРЫВАНИЙ

**М**ультипрограммная вычислительная машина выполняет одновременно несколько программ. В такой ЭВМ должна предусматриваться возможность временного *прерывания* одной из выполняемых программ, скажем программы А, и передачи всех ресурсов машины другой, более важной в текущий момент (или, как говорят, более приоритетной) программе, например программе Б. Программа Б, как и непосредственно ей предшествовавшая программа А, получает доступ ко всем устройствам ЭВМ. Существует опасность, что обрабатываемая программой А информация будет разрушена выполнением программы Б. Необходимо спасти всю информацию, которая относилась к программе А и содержалась в регистрах ЦП в момент прерывания, для того чтобы после завершения программы Б (или ее части) можно было бы вернуться к продолжению программы А.

В числе разнообразных причин, вызывающих прерывание программ, могут быть такие:

- от одного из Устройств Ввода-Вывода поступил сигнал о его готовности начать обмен информацией с Центральным Процессором;
- при выполнении программы возникла ошибка, и программа не может быть продолжена без вмешательства специальных программ или человека;
- закончено выполнение текущей программы;
- программист или оператор ЭВМ хочет немедленно сообщить машине сведения, которые способны повлиять на отношение ЭВМ к программе.

Возможность ЭВМ оперативно реагировать на изменение внешних условий обеспечивается системой прерываний. Такая система представляет собой переплетение аппаратных средств (блок прерываний Центрального

Процессора) и специальных управляющих программ (программы обработки прерываний).

Рассмотрим, как обработка прерываний распределяется между аппаратурой и специальными программами. В поведении системы при обработке каждого отдельного прерывания можно выделить несколько фаз.

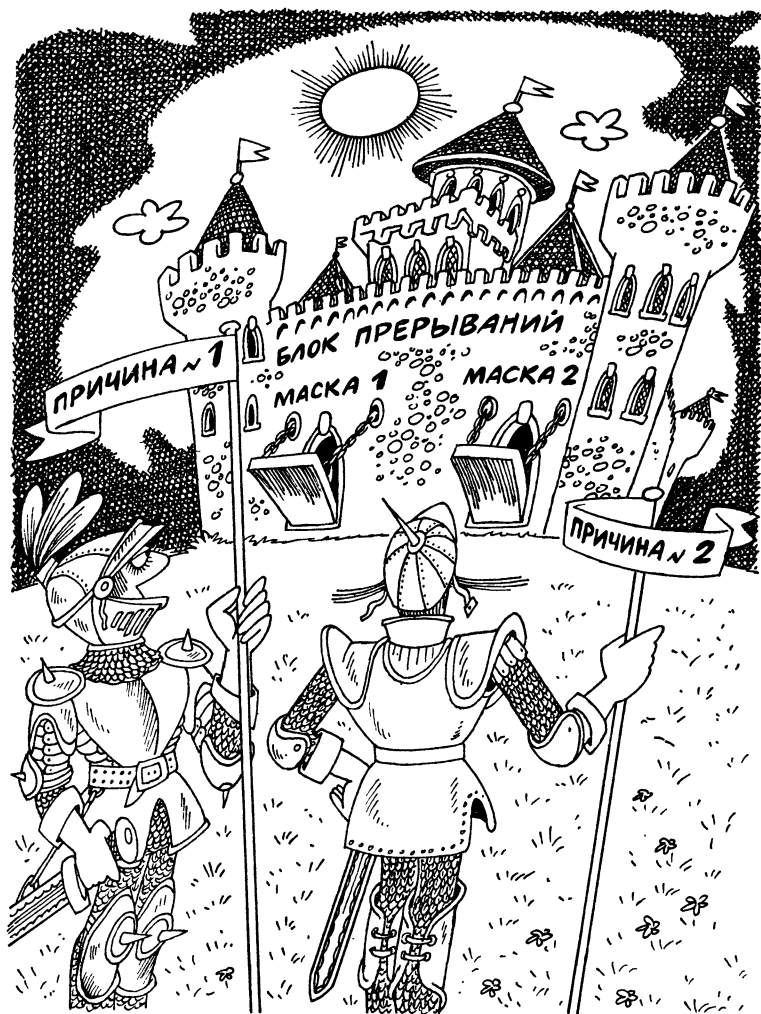


Рис. 91

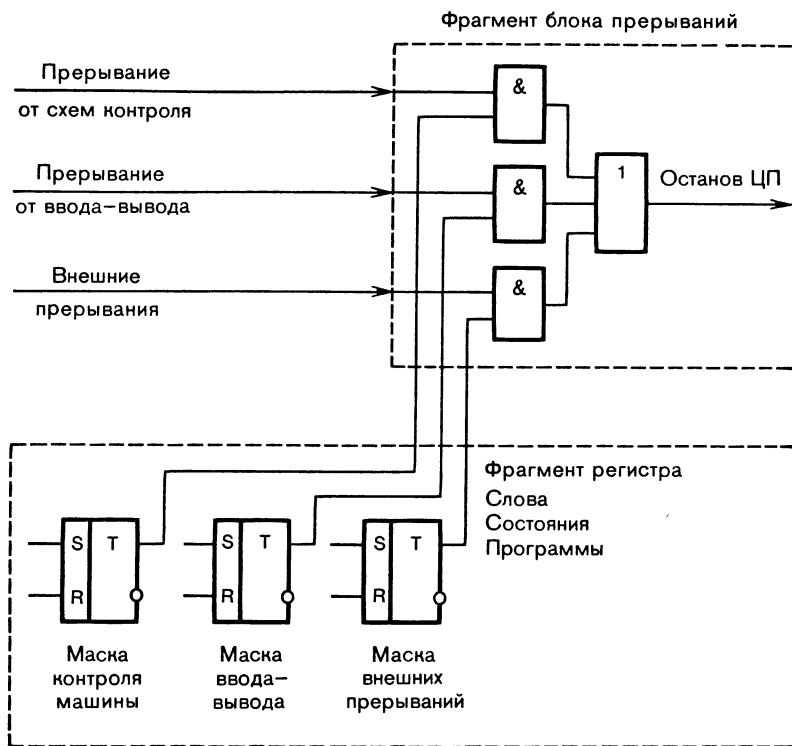


Рис. 92

### Остановка программы

Сигнал прерывания поступает в блок прерываний ЦП. Каждая группа причин прерываний имеет свой персональный вход в блок прерываний (рис. 91). Свообразными воротами, разрешающими или запрещающими прерывания, служат маски Слова Состояния Программы. Если прерывания от данной группы причин разрешены, то блок прерываний останавливает выполнение программы А (рис. 92). Начиная с этого момента, блок прерываний выполняет функции руководителя и организатора всех работ, проводимых в Центральном Процессоре.

### Запоминание старой программы

Перед тем как команды новой программы Б начнут поступать в Центральный Процессор, вся связанная со

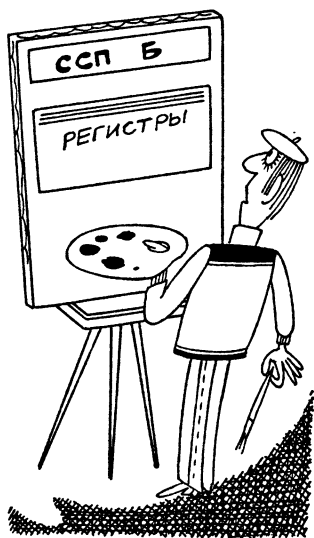
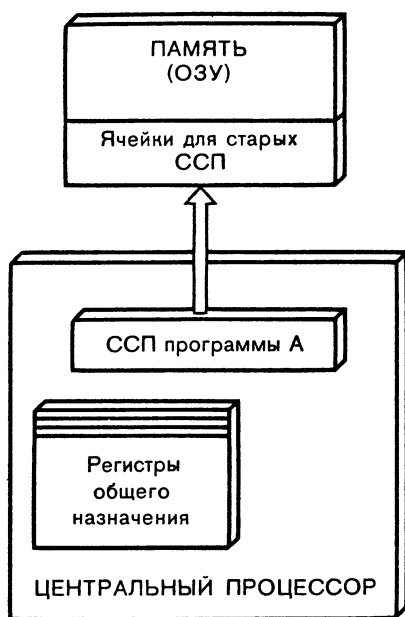


Рис. 93

старой программой А информация собирается и записывается в память. Центральный Процессор ЕС ЭВМ содержит в своем составе 16 регистров общего назначения и регистр ССП, который фиксирует состояния остальных функционально важных регистров и Счетчика Адреса Команд. Спасением содержимого регистра ССП занимается блок прерываний. С каждым типом прерываний связана определенная ячейка основной памяти. В такую ячейку памяти блок прерываний записывает старое ССП, соответствовавшее состоянию программы А в момент прерывания. Спасением содержимого регистров общего назначения займется программа обработки прерывания в следующей фазе (рис. 93).

**Запуск программы обработки прерывания**

На этом этапе иницируется (запускается) специальная программа обработки прерывания. Каждый тип прерывания имеет свою программу обработки. После того

как блок прерываний спрячет в память старое ССП, он обратится в другую фиксированную ячейку памяти, где заранее подготовлено новое ССП. В адресном поле нового ССП обычно располагается адрес первой команды программы Б. На первом этапе выполнения программы Б в память (обычно в ОЗУ) переписывается содержимое регистров общего назначения. Начиная с момента получения нового ССП Центральным Процессором и переписывания регистров общего назначения, блок прерываний слагает с себя полномочия руководителя работ до поступления сигнала следующего прерывания. Если программа Б является наиболее приоритетной, то маски ССП не позволят прервать программу Б до ее окончания (рис. 94).

### О к о н ч а н и е

Когда необходимые операции, связанные с обработкой прерывания, закончены, система «аппаратура

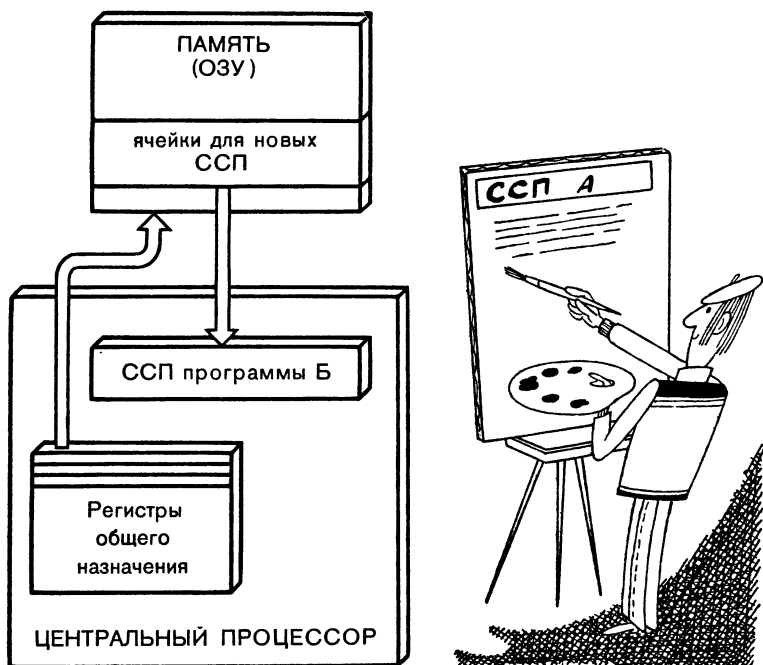


Рис. 94



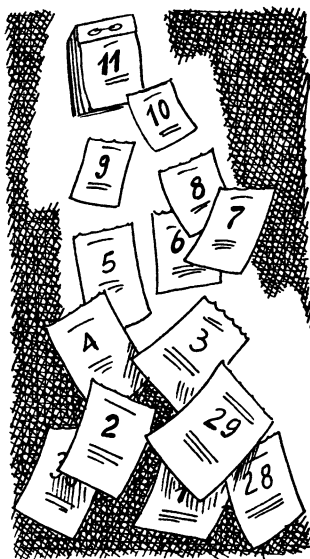
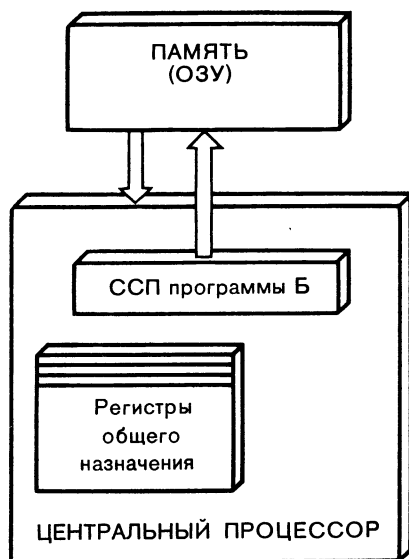


Рис. 95

ЭВМ + управляющие программы» определяет, какую программу надо выполнять далее, после завершения программы Б. Чаще всего управление передается прерванной перед этим программой А (рис. 95).

## Восстановление

При повторном инициировании прежней программы А необходимо восстановить содержимое регистров и Счетчика Адреса Команд и тем самым восстановить состояние ЦП в момент прерывания. Реставрацию проводят программы обработки прерываний. Для этого используется старое ССП, которое было записано в память в фазе запоминания. Старое ССП, содержащее в адресном поле адрес последней выполненной команды программы А, пересылается в регистр ССП Центрального Процессора. Точно так же восстанавливается содержимое регистров общего назначения. ЦП, выполнив программу Б, возвращается к прерванной программе А (рис. 96). Во время выполнения команды (из программы А) на входе блока прерываний ЦП могут появиться

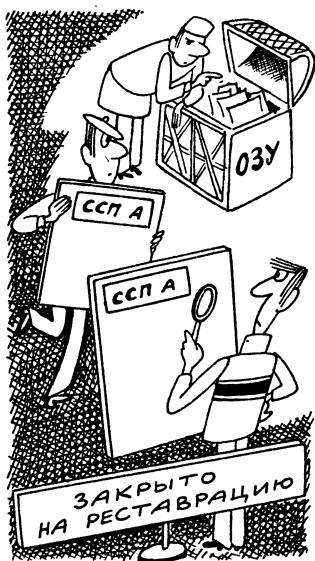
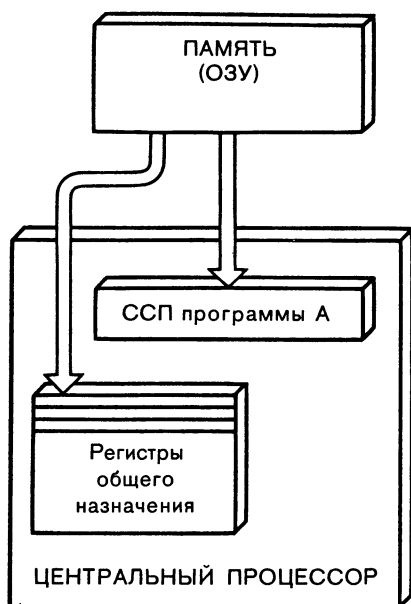


Рис. 96

сигналы нескольких прерываний любого из следующих типов:

- прерывания от *схем контроля аппаратуры*; они представляют собой сообщения о неисправности оборудования, о масштабах вызванного нарушения, о его месте в машине и, конечно, о причине неисправности;

- прерывания *ввода-вывода*; процессор ввода-вывода сообщает Центральному Процессору о завершении операции обмена или возникает ситуация, требующая участия ЦП;

- *внешние* прерывания; две ЭВМ могут быть объединены в вычислительный комплекс с общим полем оперативной памяти; для того чтобы ЦП одной из машин комплекса узнал о том, что другая машина подготовила для него определенную информацию, используются внешние прерывания;

- *программные* прерывания; такие прерывания возникают, если обнаружены ошибки в программе: неверно используются команды, выбраны недопустимые операнды или предпринята попытка обратиться в «чужую»

память (ключ защиты памяти у текущей программы не совпал с ключом защиты той области памяти, которая этой программе отведена);

— обращение к программам системы обработки прерываний; например, после окончания программы А специальная команда (последняя команда программы А) может обратиться к управляющим программам с тем, чтобы они передали Центральный Процессор и все остальные устройства ЭВМ в распоряжение другой программы, например программы Б.

Прерывания различных типов перечислены в приведенном списке в порядке значимости. Это означает, что при одновременном появлении нескольких прерываний первым обрабатывается прерывание от схем контроля аппаратуры. Остальные прерывания смогут быть обработаны лишь после того, как завершится обработка прерывания от схем контроля. Приоритетность этого типа прерываний понятна: работа на вышедшем из строя оборудовании если и возможна, то, во всяком случае, ненадежна. Следующими по значимости являются прерывания ввода-вывода. Они позволяют распараллелить работу Центрального Процессора и процессоров Ввода-Вывода, повышая тем самым эффективность работы ЭВМ в целом. В частности, не теряется информация, которую выполняемая программа может заблаговременно заказать у устройства ввода, освободившийся процессор ввода-вывода может быть загружен новой работой. Приоритет у внешних прерываний ниже, чем у прерываний ввода-вывода. По сравнению с прерываниями ввода-вывода внешние прерывания возникают существенно реже. Источником программных прерываний и обращения к управляющим программам является выполняемая программа, так что одновременно эти два типа прерываний не возникают: если выполняемая программа обращается к системе прерываний, то она не может вызвать программного прерывания, и наоборот. Иерархия прерываний, о которой шла речь, характеризует систему прерываний машин серии ЕС ЭВМ. В других современных вычислительных машинах системы прерываний могут иметь небольшие отклонения в ту или другую сторону. Отклонения касаются в основном распределения обязанностей между блоком прерываний Центрального Процессора и управляющими программами. Система прерываний позволяет согласовать последовательную и в целом синхронную обработку информации в Центральном Про-

цессоре с различными асинхронными событиями в работе ЭВМ. Это весьма важно, поскольку большинство прерываний распределено случайно во времени: невозможно запланировать заранее сбой машины, программную ошибку или момент нажатия клавиши на клавиатуре дисплея.

## ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

**З**накомясь с системой команд, мы убедились, что машинный язык представляет собой средство непосредственного общения с вычислительной машиной. Каждый тип ЭВМ имеет свою систему команд, свой машинный язык. Например, программа, вычисляющая сумму произведений

$$X \cdot Y + Z \cdot T,$$

для машины БЭСМ-6 может иметь такой вид:

```
010 0256
017 0413
000 0033
010 0071
017 1005
004 0033
```

Предполагается, что в ячейке с адресом 256 хранится число  $X$ , для числа  $Y$  отведена ячейка 413, для числа  $Z$  — ячейка 71, число  $T$  находится в ячейке 1005, результат записывается в ячейку с адресом 0033. В этой программе использованы следующие команды машинного языка БЭСМ-6:

- 010 — пересылка числа из ячейки ОЗУ в сумматор;
- 017 — умножение числа, хранящегося в ячейке ОЗУ, на число из сумматора; результат операции остается в сумматоре;
- 000 — пересылка числа из сумматора в ячейку ОЗУ;
- 004 — сложение числа из ячейки ОЗУ и числа из сумматора так, что сумма остается в сумматоре.

Написание даже столь несложной программы оказывается трудоемким делом. Видно, что для общения с машиной нужно наизусть выучить коды операций в командах машинного языка, чтобы строить из этих элементар-

ных предписаний сложные действия. Поэтому программирование на машинном языке часто называют кодированием. В командах машинного языка существенным образом отражаются конструктивные особенности конкретной машины — система адресности, формат команды, наличие регистровой памяти и количество регистров и т. п. Поэтому общение с вычислительной машиной на ее машинном языке трудоемко. Оно не затруднительно для инженера-разработчика ЭВМ, а для программистов, в обязанности которых входит подготовка задания машины и общение с нею, информация об этих конструктивных особенностях часто оказывается избыточной. Программист должен уметь описать данные, прочитать выдаваемые машиной результаты, а главное, описать решаемую задачу понятными машине средствами, т. е. составить список указаний для ЭВМ в виде программы. В такой работе программисту было бы удобнее не связывать себя с номерами регистров, размером сумматора, кодами операций. Так возникла проблема автоматизации труда программистов средствами... программирования. Одним из самых первых этапов решения этой проблемы стали языки программирования, названные *автокодами*. С помощью простого автокода можно сделать легко читаемыми колонки цифр в программах машинного языка: вместо адресов в программе, написанной на автокоде, употребляют имена величин, участвующих в вычислениях, а вместо кодов операций — их содержательные (мнемонические) обозначения. Тогда обсуждавшуюся выше программу можно переписать:

ЗП	X
УМН	Y
ЧТ	X
ЗП	Z
УМН	T
СЛ	X

Здесь вместо кода умножения используется осмысленное и выразительное обозначение УМН, вместо кода сложения — СЛ, вместо кода посылки в сумматор — ЗП (запись), вместо кода пересылки из сумматора в память — ЧТ (чтение).

Подобно системе команд автокод тоже является языком программирования. Он весьма похож на машинный язык, и все же написанную на автокоде программу

невозможно непосредственно выполнить на машине: в машинном языке БЭСМ-6 отсутствуют операции с буквенными кодами ЗП, ЧТ, УМН, СЛ. Эту программу надо сначала перевести на машинный язык. Как это нетрудно понять, сравнивая две аналогичные программы — на машинном языке и на автокоде, такую переводческую работу можно формализовать, используя таблицы-словари, которые устанавливают соответствия между кодами операций и их мнемоническими обозначениями, между адресами ОЗУ и кодирующими их буквами или словами. Формальный характер перевода с одного языка программирования на другой позволяет передать эту работу специальной программе. Программу, осуществляющую такой перевод, называют *ассемблером*, а выполняемую ею работу — *ассемблированием* (рис. 97). Ассемблер представляет собой написанную на машинном языке программу, которая, получая на входе текст некоторой программы на автокоде, выдает в качестве своего результата текст той же программы на машинном языке. Схема ассемблирования показывает, что при наличии ассемблера к ЭВМ можно обращаться с любой программой, написанной на автокоде. Следующий этап в автоматизации программирования начался с создания более совершенных автокодов. Их ассемблеры могут не только перекодировать буквенные обозначения операций, но и реализовать некоторые операции, не существующие в машинном языке. Так, система команд машины может не иметь в своем составе команды извлечения квадратного корня. Однако легко написать программу, вычисляющую корень по заданному значению подкоренного выражения, пользуясь при этом командами основного набора системы команд. После этого программист получает право использовать в своих автокодовских программах операцию извлечения корня, обращаясь к соот-

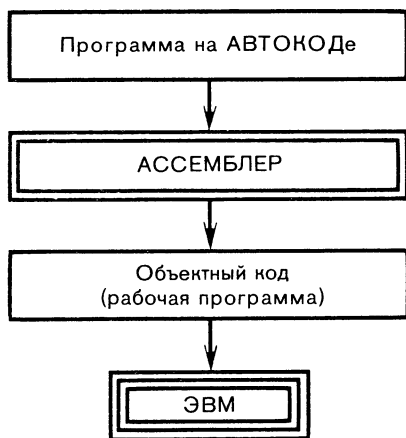


Рис. 97

ветствующей программе, хранящейся в памяти ЭВМ. Общая черта всех автокодов — их ориентация на конкретный тип ЭВМ, в машинные команды которой ассемблер переводит автокодовскую программу. Поэтому их называют машинно-ориентированными языками. Именно жесткая привязанность таких языков к конкретным типам машин определила ограничения на очередном этапе автоматизации программирования, когда возникла потребность обмена программами, создававшимися для разных типов ЭВМ. Тем более что определились тенденции достаточно быстрой смены поколений машин (в среднем один раз в пять—семь лет) и относительно продолжительной жизни программных систем — этого концентрата человеческих знаний в прикладных областях. Объективная обусловленность таких требований вызвала появление машинно-независимых языков программирования. Для этих двух больших групп языков — машинно-ориентированных и машинно-независимых — установились со временем более употребительные названия: их называют соответственно языками *низкого* и языками *высокого* уровня, подчеркивая тем самым степень удаления языка от уровня машинной системы команд. В настоящее время программисты используют десятки различных языков программирования высокого уровня, среди которых наибольшую известность приобрели Алгол, Паскаль, Фортран, ПЛ/1, Кобол и ряд других.

Программа на таком языке записывается обычно в обозначениях, легко понимаемых людьми. Каждое предписание языка высокого уровня компактно в том смысле, что заменяет собой целую группу команд машинного языка или автокода. Программа вычисления суммы двух произведений на Коболе записывается так:

$$\text{вычислить } R = X * Y + Z * T$$

В языках программирования высокого уровня практически никак не учитываются особенности отдельных конкретных ЭВМ или их систем команд. Если программа написана на одном из языков высокого уровня, например на Алголе, то эта программа может быть введена и в БЭСМ-6, и в машины серии ЕС, и вообще в машину любого типа, если для нее существует программа-переводчик с Алгола на машинный язык. Хотя общая идея решения проблемы и здесь подсказана ассемблерами, автоматический перевод с языка высокого уровня на машинный язык значительно сложнее ассемблирования:

теперь уже речь идет не о простой замене одних обозначений другими, а о реальном переводе с одного языка на другой со всеми его трудностями — лексическим анализом, синтаксическим анализом и, наконец, синтезом или генерированием результирующего текста, представляющего собой программу на машинном языке. Вместе с тем грамматика языков программирования как низкого, так и высокого уровня существенно более формальна, чем грамматика любого человеческого, естественного языка. К тому же лексика языков программирования намного беднее средств человеческого общения. Поэтому даже задачи переводов с языков высокого уровня могут быть описаны формально, а значит, могут быть запрограммированы. Несомненно, что такого рода программы являются весьма сложными. Перевод программы с языков высокого уровня называют *трансляцией*, а программу, выполняющую такой перевод, — *транслятором*.

Итак, всякий транслятор связывают с не-

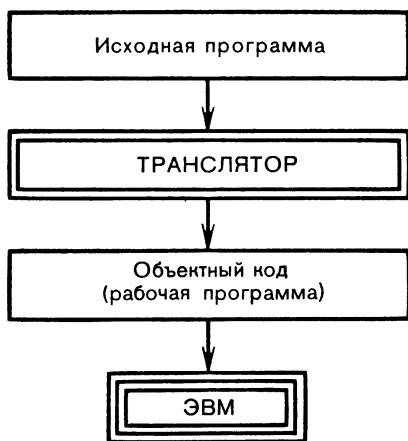


Рис. 98

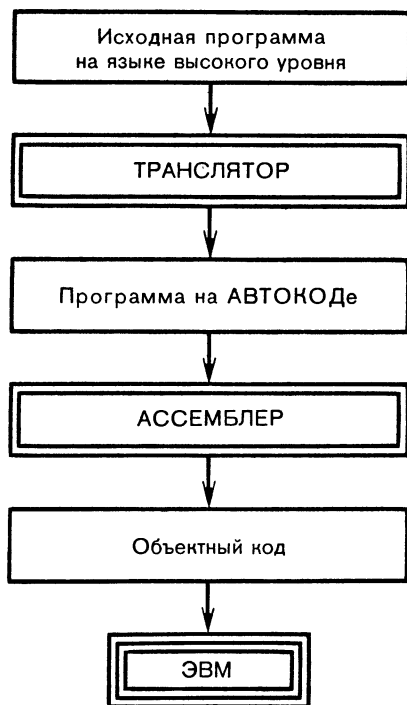


Рис. 99



которой парой «язык высокого уровня — машинный язык (или в общем случае машинно-ориентированный язык)». Будучи переводчиком, транслятор представляет собой программу, на вход которой поступает текст написанной на исходном языке программы. Эту последнюю программу называют *исходной* (или транслируемой) программой. Результат работы транслятора называется *рабочей* (или исполняемой) программой (или, иногда объектным кодом) (рис. 98). Довольно часто показанная схема на практике выглядит более развернуто: сначала выполняется трансляция с языка высокого уровня на автокод, а затем уже ассемблирование (рис. 99). Благодаря наличию трансляторов программу, написанную на языке высокого уровня, таком, как, скажем, Алгол, можно выполнять на разнотипных машинах. Так, если для ЕС ЭВМ существует транслятор с Алгола на машинный язык ЕС ЭВМ или их автокод, то программист может работать со своей программой, написанной по правилам Алгола, на любой машине этой серии. Та же программа сможет выполняться и на БЭСМ-6, и на СМ-4, которые имеют свои трансляторы с Алгола (рис. 100). Благодаря набору трансляторов ЭВМ становится полиглотом.

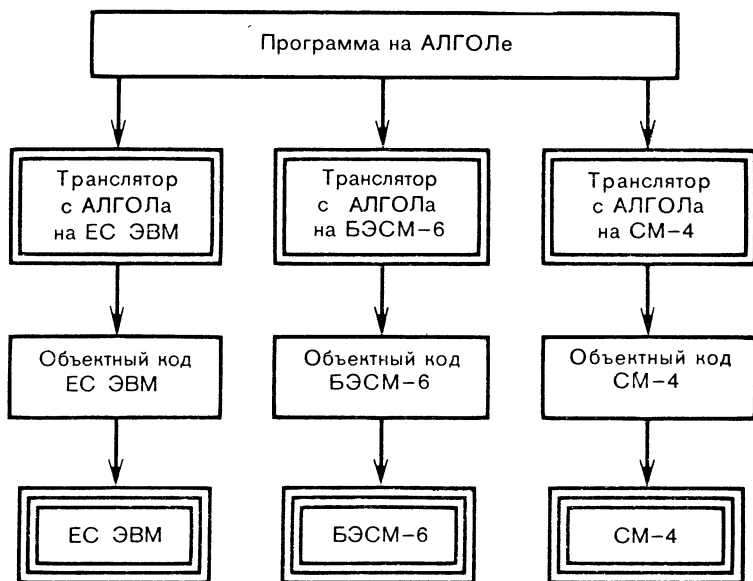


Рис. 100

Преимущества программирования на языках высокого уровня пользователи вычислительных машин оценили очень быстро. И сразу же языки программирования стали появляться один за другим. Уже в 60-е годы на страницах нескольких программистских журналов появился рисунок на сюжет библейской притчи о вавилонском столпотворении. Когда-то в незапамятные времена, когда все человечество говорило на едином языке и каждый понимал другого, задумали люди построить в Вавилоне огромную башню (столп) до самого неба. Увидев это незаконченное еще сооружение, бог рассердился на смелых и гордых людей и наказал их, сделав человеческое племя многоязычным. Люди перестали понимать друг друга, и строительство вавилонского столпа остановилось.

Рисунок новой вавилонской башни, каждый камень которой — один из языков программирования, воспроизведен здесь в слегка измененном виде: добавлены некоторые появившиеся с тех пор языки да названия языков англо-американского происхождения даны в русской транскрипции (рис. 101). Можно согласиться с тем, что многоязычие — бедствие человеческого общества: людям на Земле жилось бы, вероятно, лучше, если бы все мы говорили на одном языке. А вот многообразие языков программирования объективно оправдано. Оно объясняется тем, что каждый язык программирования предназначен для решения некоторого класса задач, некоторой предметной области. Такой класс задач соответствует определенной сфере человеческой деятельности, характеризующейся обычно специфической совокупностью методов решения задач. Различные предметные области приложений программирования представляют, например, инженерно-технические расчеты, учетно-статистические задачи, моделирование дискретных и непрерывных процессов и т. п. Каждая предметная область формулирует свои требования к языку, которым описываются задачи соответствующего класса. Языки программирования высокого уровня, создаваемые по требованиям тех или иных предметных областей, называют проблемно-ориентированными (в отличие от машинно-ориентированных языков программирования низкого уровня). Например, в языке Фортран, спроектированном специально для описания научно-технических и инженерных расчетов, существуют удобные способы представления векторов и матриц — объектов, применяемых при решении разнообраз-



ных физико-технических задач. Зато по возможностям обработки текстов Фортран значительно уступает таким языкам, например, как Лисп или Сетл. Это вполне естественно: методы работы с текстовыми строками лежат вне предметной области Фортрана. Совершенно иная предметная область — обработка и оформление отчетно-статистической документации. Ее потребности обусловили создание языка Кобол. Все более и более заметную роль в многочисленных приложениях ЭВМ играет машинная графика. Разработана группа языков, в которых геометрические объекты — точка, прямая, окружность, плоскость — являются элементарными понятиями языка.

Каждый из языков программирования в известном смысле универсален: средствами любого языка можно описать решение задачи, коль скоро существует ее алгоритм. Однако ни один программист не станет программировать на Коболе решение дифференциальных уравнений метеопрогноза: такая программа неудобна для чтения и потеряет в компактности и, что более важно, в быстрой работе. Каждый язык эффективен в своей предметной области. Хороший программист должен не только уметь написать программу на том или ином языке, но, главное, уметь выбрать язык, наиболее подходящий для решения задачи. Разные режимы общения с вычислительной машиной определяют еще одну классификацию языков программирования. Для многих задач оказывается вполне удовлетворительным режим решения, при котором подготовка программы и ее данных полностью отделена от выполнения программы на ЭВМ. В таком режиме программа и ее данные предварительно наносятся на некоторый информационный носитель, чаще всего на перфокарты. Одиночной записи, например, приведенного выше предписания на Коболе соответствует одна перфокарта (рис. 59). Программе соответствует, таким образом, набор перфокарт, колода. Подготовленную к решению колоду перфокарт программист передает оператору ЭВМ, и тот, объединив несколько колод в один пакет, через Устройство Ввода с перфокарт вводит их в память машины. Именно отсюда и родилось название пакетного режима общения с ЭВМ. Ясно, что выполнение программ в пакетном режиме характеризуется не только особенностями машины, но и возможностями языков программирования. Языки, которые обеспечивают общение с ЭВМ в пакетном режиме, называют пакетными. На таких языках создаются программы расчетов зара-

ботной платы и школьных расписаний, долгосрочных прогнозов и остойчивости корабля, экономических планов завода и оптимального электроснабжения региона. Есть, однако, много задач, которые принципиально нельзя решать в пакетном режиме. Если вычислительная машина используется врачом при оперативной постановке диагноза болезни, то врачу-пользователю ЭВМ необходимо получить диагноз за минимально короткое время после ввода симптомов болезни в машину. Когда ЭВМ помогает учителю на школьном уроке, то ответ машины ученику или сообщение о неправильных действиях учащегося ЭВМ выдает с оперативностью самого быстрого отличника. К числу таких же задач относятся, в частности, разнообразные проблемы управления транспортом: диспетчеризация на железнодорожной станции, резервирование билетов на авиалиниях и т. п. В таких задачах существенно, что пользователь (железнодорожный диспетчер или кассир Аэрофлота) не имеет возможности ждать формирования пакета заданий, на которое в пакетном режиме уходит достаточно много времени. Реакция машины на переданную ей команду должна быть быстрой, ответ машины на запрос пользователя важно получить немедленно, практически мгновенно, словно машина разговаривает со своим собеседником в живом диалоге. Такой режим общения с ЭВМ называется *диалоговым*, а языки, с помощью которых составляются работающие в диалоговом режиме программы, — диалоговыми. О том, как машина создает иллюзию мгновенных ответов, уже говорилось в разделе «Ввод-вывод ЭВМ».

Прежде чем начать общение с собеседником на каком-либо языке, надо предварительно изучить этот язык. Не являются исключением из общего правила и языки программирования. Вместе с тем такое требование, как изучение нового, пусть даже несложного, языка, становится серьезным препятствием для очень широкого круга пользователей, которые не умеют программировать и могут даже не знать алгоритма решаемой на ЭВМ задачи, но тем не менее заинтересованы в общении с машиной. Можно считать, что такие непрофессиональные пользователи, которые, впрочем, могут быть квалифицированными специалистами в любой из областей человеческой деятельности, умеют лишь набрать запрос машине на клавиатуре терминала и прочитать выданное машиной сообщение. Современное программирование предлагает такого рода пользователям удобные средства, называе-

мые пакетами прикладных программ. Пакет прикладных программ (ППП) представляет собой программную систему, которая в рамках узкой предметной области обеспечивает общение с машиной (формулирование заданий и получение результатов). Такое общение происходит на естественном языке или в терминах этой предметной области — на простом, не требующем профессиональной программистской подготовки языке, называемом входным языком пакета (можно отметить некоторую неудачу в выборе терминологии, свойственную, впрочем, почти всем молодым научным дисциплинам: среди пакетов прикладных программ немало диалоговых систем).

ППП используются в самых разнообразных приложениях. Известны, например, различные ППП машинной графики, позволяющие проектировать с помощью ЭВМ схемы и рисунки, используя при этом только информацию о геометрии рисунка — координаты точек, типы линий, масштаб изображения и т. п. Характерный пример ППП — генераторы задач по отдельным школьным предметам, точнее, по отдельным темам в том или ином предмете. За счет достаточного сужения предметной области, например, к генератору задач по школьному курсу механики учитель может обращаться, набрав на терминале запрос на естественном, русском языке:

ДАЙТЕ, ПОЖАЛУЙСТА, 20 КИНЕМАТИЧЕСКИХ  
ЗАДАЧ С НАЧАЛЬНОЙ СКОРОСТЬЮ

или

МНЕ НУЖНО 15 ЗАДАЧ С НЕЕДИНСТВЕННЫМ  
РЕШЕНИЕМ.

В ответ программы этого пакета печатают требуемую серию задач — тексты сформулированных машиной учебных задач, а также (по желанию пользователя) их ответы и развернутый комментарий решения. Трансляторы и пакеты прикладных программ — это большие и сложные программы, без которых невозможно организовать общение с машиной в разнообразных приложениях. Совокупность всех трансляторов и пакетов, которыми располагает вычислительная машина, составляет важнейшую компоненту программного обеспечения ЭВМ (рис. 102). При наличии программного обеспечения вычислительная машина представляется обращающемуся к ней человеку-пользователю не простым набором технических уст-

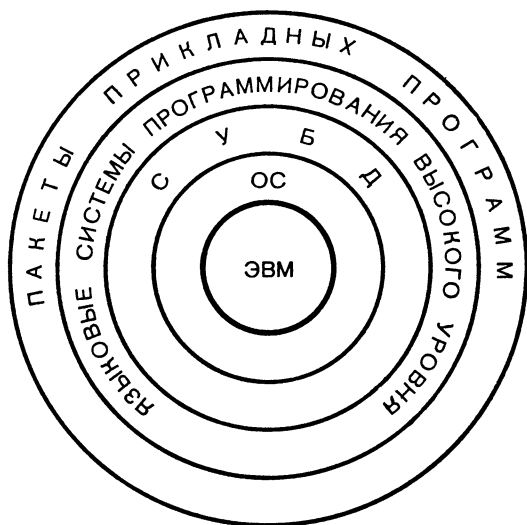


Рис. 102

ройств — ЗУ, АЛУ, УУ и УВВ, а своеобразной вычислительной системой, в которой, собственно, аппаратура ЭВМ составляет ядро, одетое в несколько слоев программного обеспечения. Пакеты прикладных программ и трансляторы языков высокого уровня составляют в этом многослойном обеспечении один из наиболее близких к пользователю слоев. Программы самого глубинного слоя относятся к *операционной системе* (ОС) ЭВМ. Функции ОС — управление ресурсами машины, организация потока проходящих через машину программ, облегчение написания и подготовки программ для ЭВМ. Современные ОС — это сложные комплексы программ, насчитывающие миллионы машинных команд. Именно операционные системы берут на себя хлопоты по обработке прерываний, по организации иерархии памяти и ее функционированию, управляют довольно сложными системами ввода-вывода. Операционные системы не разрешают пользователю непосредственно обращаться к аппаратной части ЭВМ, зато они представляют ему многообразие всевозможных удобств. Мы видели, например, неоспоримое преимущество диалогового режима общения с ЭВМ. Надо сказать, что вычислительная машина может, вообще говоря, работать в диалоговом режиме, даже имея

един-единственный терминал — пульт машины. Такие конфигурации машин часто используются в конструкторских бюро для индивидуальных инженерных расчетов. Однако, когда машина обслуживает одновременно многих кассиров, продающих авиабилеты, или отвечает на разнообразные запросы группы школьников, выполняющих лабораторную работу, все преимущества диалогового режима общения с такой машиной будут потеряны: в этом случае пользователю придется стоять в очереди и ждать, когда закончат диалог с ЭВМ все его коллеги. Перспективными с точки зрения использования в подобных ситуациях являются только многотерминальные машины, способные обслуживать в диалоговом режиме одновременно десятки и сотни пользователей так, что ни один из них не ощущает замедления реакции машины из-за занятости соседних терминалов. Такую разновидность диалогового режима называют режимом *коллективного использования ЭВМ*. Возможность коллективного использования средств обработки информации — важнейшее требование к современным вычислительным системам. К удовлетворению этого требования стремятся не только конструкторы аппаратуры ЭВМ, но, не в меньшей мере, и программисты — разработчики программного обеспечения режима коллективного использования, которое представляет наиболее ответственную компоненту общесистемного программного обеспечения. Непосредственно к слою операционных систем примыкают программы, организующие работу с используемыми в программе данными. Описание представления данных является одной из важных составных частей деятельности программиста. Эффективное выполнение программ зависит от того, как представлены и организованы обрабатываемые ею данные. В программах, составляющих на ранних этапах развития ЭВМ, данные составляли неотъемлемую часть этих программ. Такое положение казалось естественным: если программисту нужно обрабатывать какие-либо данные, то он их организует и записывает, а затем пишет программу в соответствии с задуманным им представлением данных. Другой программист, даже если ему и приходится частично использовать ту же информацию, изобретает другую, кажущуюся ему наиболее подходящей для его задачи форму представления данных, обосновывая это тем, что он пишет другую программу. В наши дни, когда на ЭВМ решаются большие и сложные задачи и очень часто системы задач,



положение изменилось. В запоминающих устройствах современных вычислительных машин накапливается все более и более обширная информация, которая определенным образом отражает результат деятельности больших коллективов предприятий и целых отраслей. В таких условиях создаются новые отношения между данными и программами. Действительно, сейчас немыслимо с каждой из многочисленных задач автоматизированной системы управления (АСУ) большого завода связывать свой набор данных, не считаясь с представлением данных в едином заводском информационном фонде. Более рационально, создавая программу, ориентироваться на принятые в этой АСУ описания данных. Таким образом, данные становятся общими для большой совокупности программ и, что особенно важно, для большого коллектива пользователей ЭВМ. Так проявляются два основных взаимозависимых принципа, определяющие развитие современного программирования, — обобществление данных и отделение данных от программ. Информационные хранилища, создаваемые на основе этих принципов, представляют собой базы данных. *База данных* определяется как совокупность взаимосвязанных хранящихся вместе данных, используемых оптимальным образом во многих программах разных пользователей. Данные запоминаются так, чтобы они были независимы от программ, использующих эти данные. Для добавления новых или изменения существующих данных, а также для поиска данных в базе применяются единые для всей базы программные средства. Размещение данных и их перераспределение в запоминающих устройствах, контроль за частотой использования — вся эта работа с элементами баз данных невидима пользователю. Она выполняется специальными программами. На долю программ, предназначенных для того, чтобы предельно упростить общение широкого круга пользователей с базой данных, выпадают и такие функции, как организация ввода данных, обновление базы данных, обеспечение защиты информации от возможного воздействия пользовательских программ и технических сбоев, перераспределение информации между различными Внешними Запоминающими Устройствами, обслуживание вывода. Совокупность служебных программ, которые обеспечивают жизнедеятельность баз данных, называют *системой управления базами данных* (СУБД). Целый ряд дополнительных проблем возникает тогда, когда Процессор ЭВМ имеет доступ не

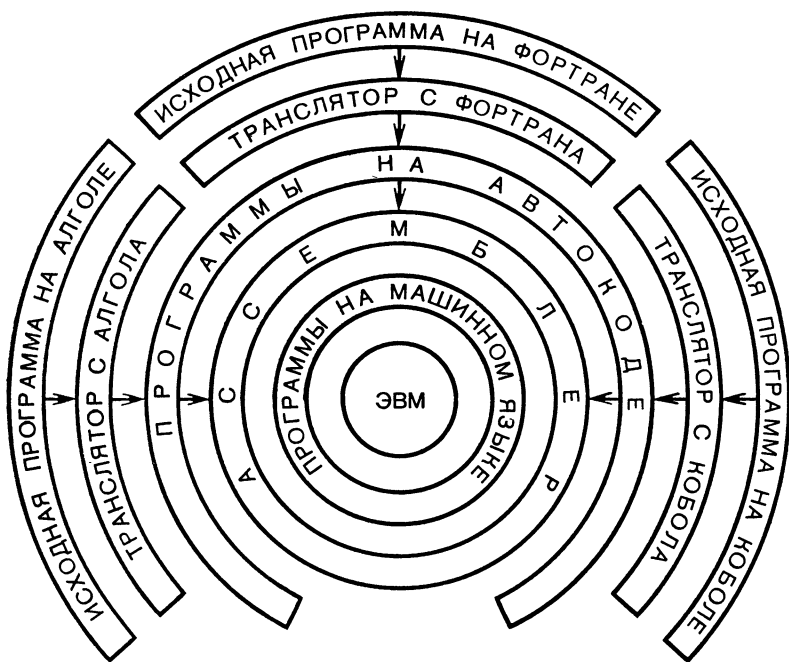


Рис. 103

только к «своим» ЗУ, а является одним из узлов информационной сети, связывающей большое число вычислительных машин. Тогда для Процессора должно стать безразличным, какое из ЗУ, какая из баз данных предоставила в его распоряжение обрабатываемую информацию. Система хранения информации во многих различных, в том числе территориально удаленных, базах данных называется *распределенной базой данных*. Управление распределенными базами данных осуществляется специальными средствами программного обеспечения информационных систем.

Вычислительная система «ЭВМ + программное обеспечение» представляет собой многоуровневую систему (рис. 103). Границу между отдельными уровнями не всегда удастся провести так отчетливо, как на этом рисунке. Более того, даже граница, разделяющая программное обеспечение и ядро вычислительной системы — технические устройства, составляющие аппаратную часть



Рис. 104

ЭВМ, — в настоящее время представляется достаточно расплывчатой. Ранее уже шла речь об эквивалентности аппаратных и программных реализаций тех или иных функций. Решение о способе реализации принимается на основании таких факторов, как стоимость, скорость вычислений, емкость памяти, надежность и частота ожидаемых изменений (рис. 104).

## МИКРОПРОГРАММИРОВАНИЕ

Пользователь ЭВМ — программист и ее разработчик — инженер-электроник смотрят на вычислительную машину с двух разных точек зрения. Для программиста элементарной порцией действия представляется машинная команда. С точки зрения инженера, выполнение машинной команды складывается из ряда более элементарных действий — *микроопераций*. Каждый из функциональных узлов машины — сумматор, регистр, схема сравнения, счетчик и т. п. — предназначен для выполнения конкретной микрооперации. Пересылка информации, например, между двумя регистрами тоже является микрооперацией. Выполнение каждой микрооперации занимает некоторый фиксированный интервал времени, называемый *тактом*. Величина такта определяется, прежде всего, быстродействием интегральных схем, использованных в конкретной модели ЭВМ.

Процесс выполнения команды «Сложить», на примере которой рассказывалось о работе Центрального Процессора, можно представить в виде последовательно выполняемых микроопераций:

- 1-я микрооперация — передача из СчАК в РА (рис. 50);
- 2-я микрооперация — чтение из ОЗУ в РК (рис. 51);
- 3-я микрооперация — чтение из ОЗУ в Р1 (рис. 52);
- 4-я микрооперация — чтение из ОЗУ в Р2 (рис. 53);
- 5-я микрооперация — сложение в сумматоре, передача из См в РР (рис. 54);
- 6-я микрооперация — запись из РР в ОЗУ, увеличение СчАК на единицу (рис. 55).

В ходе каждой микрооперации УУ Центрального Процессора вырабатывает управляющие сигналы, которые либо разрешают транспортировку информации от одного узла к другому, либо начинают операцию в одном из узлов машины, например в сумматоре или счетчике.

Существуют два способа построения Устройств Управления ЭВМ. Один из них состоит в том, что для каждой машинной команды УУ содержит свой набор схем. Такой набор схем в нужном такте вырабатывает соответствующие управляющие сигналы. Наборы схем, соответствующие отдельным командам, связаны между собой. Например, набор схем команды «Сложить» составляет часть набора, определяющего команду «Умножить». Такой принцип организации УУ называют жест-

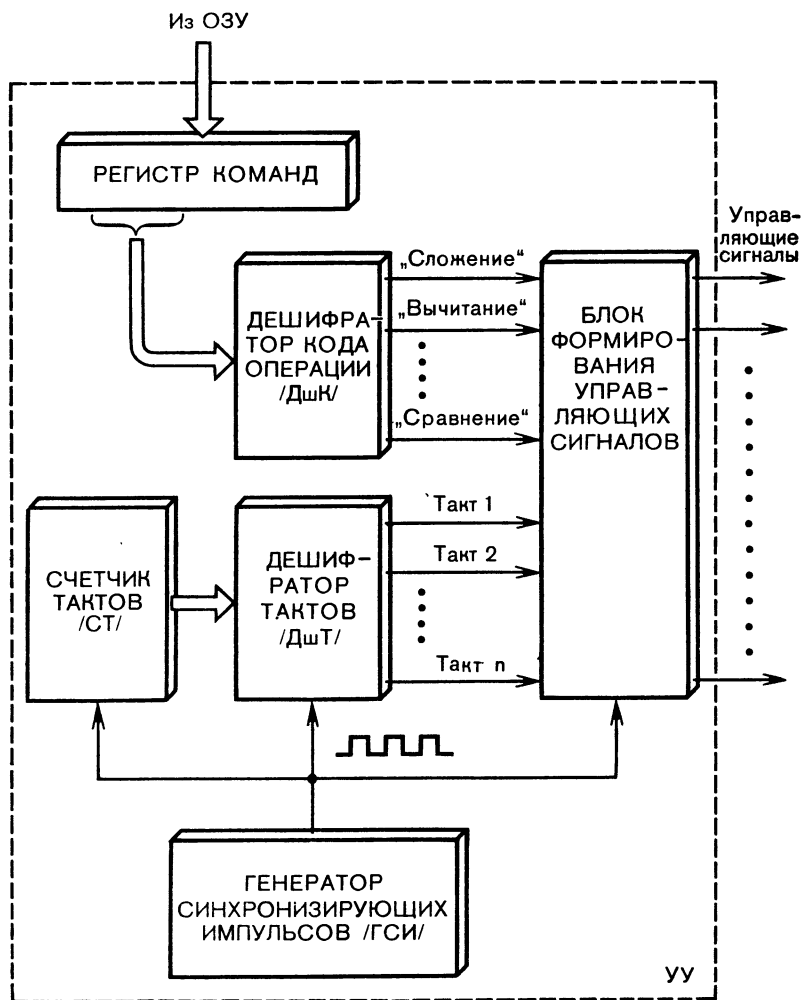


Рис. 105

кой логикой. В состав УУ Центрального Процессора с жесткой логикой обычно входят регистр команд, дешифратор кода операции, счетчик тактов, дешифратор тактов, генератор синхронизирующих импульсов (ГСИ) и блок формирования управляющих сигналов (рис. 105). Выполняемая машинная команда хранится в регистре команд. Дешифратор кода операции преобразует код

машинной команды в сигнал, появляющийся на персональном для каждой команды выходе дешифратора. После завершения очередного такта ГСИ вырабатывает импульс, и содержимое счетчика тактов увеличивается на единицу. В соответствии с сигналами, поступающими от дешифратора кода операции и дешифратора тактов, блок формирования управляющих сигналов в каждом такте работы УУ вырабатывает определенные сигналы. При формировании управляющих сигналов могут анализироваться и учитываться состояния некоторых управляющих регистров. Так, при выполнении команды «Условный переход» существенно содержимое регистра признака результата. Другой подход основан на принципе микропрограммного управления. Сущность этого принципа состоит в том, что каждая машинная команда выполняется специальной программой. Такую программу называют *микропрограммой*. Она хранится в так называемом *постоянном запоминающем устройстве* (ПЗУ). Информация записывается в ПЗУ только один раз — в процессе изготовления ЭВМ на заводе. В ходе эксплуатации вычислительной машины ее ПЗУ работает только в режиме считывания. Получаемая при этом информация предназначена исключительно для вспомогательного, внутримашинного использования и не доступна программисту. Именно с помощью информации, записанной в ПЗУ, микропрограммное Устройство Управления организует требуемые последовательности управляющих сигналов. При микропрограммной организации УУ каждой микрооперации ставится в соответствие слово (или часть слова) ПЗУ, называемое *микрокомандой*. Микропрограммы, соответствующие отдельным командам из системы команд ЭВМ, как раз и составляются из таких микрокоманд. ПЗУ хранит микропрограммы точно так же, как основная память ЭВМ помнит «большие» программы, написанные программистами. Последовательно считываемые из ПЗУ микрокоманды, принадлежащие, например, микропрограмме машинной команды «Сложить», служат источником информации о поведении Центрального Процессора. Структурная схема микропрограммного Устройства Управления приведена на рисунке 106. В каждом такте работы Устройства Управления из ПЗУ в регистр микрокоманды поступает очередная микрокоманда. С помощью группы дешифраторов микрокоманда расшифровывается, и на выходах этих дешифраторов появляются управляющие сигналы, необходимые в дан-

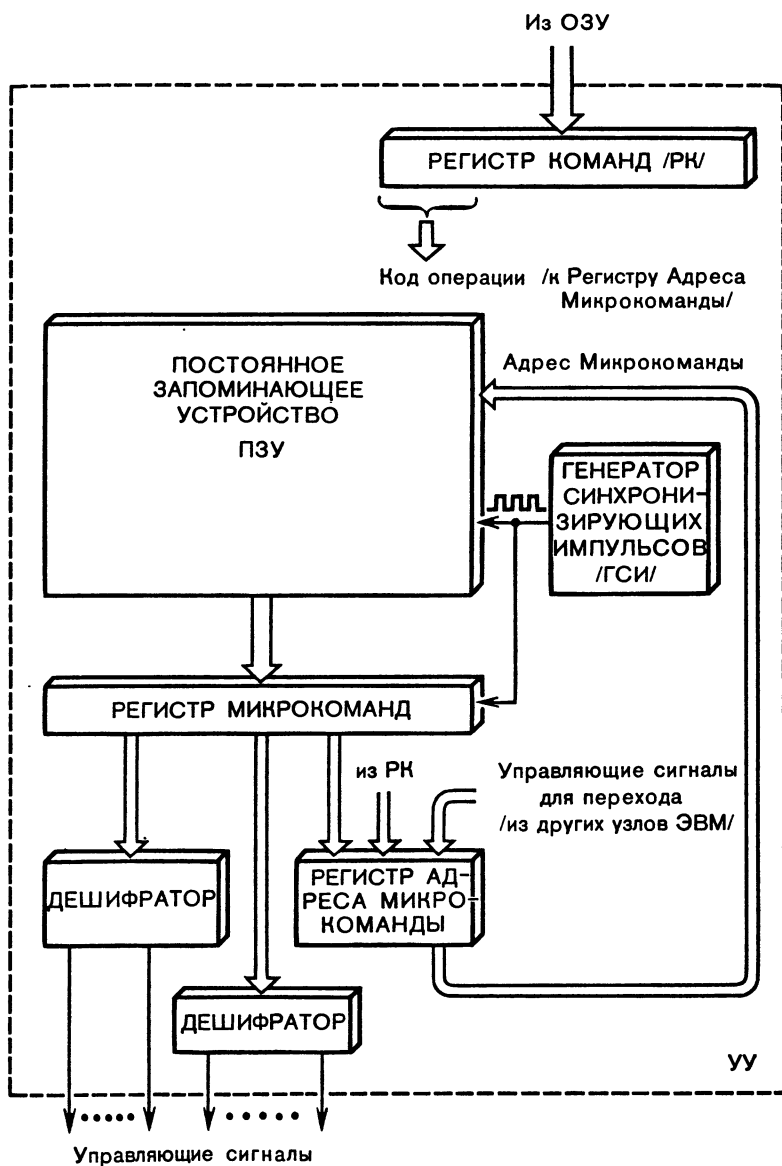


Рис. 106

ном такте. Микрокоманда содержит несколько полей. Код микрооперации указывает вид работы, которая должна быть выполнена в такте этой микрооперации. Например, код микрооперации 00001<sub>2</sub> может обозначать «Переслать содержимое регистра, номер которого указан в поле P1, в регистр, указанный полем P2». Поля P1 и P2 содержат номера регистров Центрального Процессора, принимающих участие в микрооперации. Микрокоманда содержит в своем формате адрес следующей выполняемой микрокоманды. Ее следует выбрать из ПЗУ после выполнения текущей микрокоманды. В том случае, если выбор следующей микрокоманды зависит от каких-либо дополнительных условий (примерно так же, как это происходит при выполнении машинной команды «Условный переход»), эти условия задаются полем кода перехода.

Важно отметить, что любой из способов построения Устройства Управления — либо на основе жесткой логики, либо с использованием принципа микропрограммного управления — не более чем инженерный метод проектирования. Программист-пользователь может и не догадываться о принципах организации оборудования. Жесткая логика и микропрограммное управление имеют свои достоинства и свои недостатки. Главным преимуществом жесткой логики является высокое быстродействие, а важным качеством микропрограммных устройств — возможность гибкой замены микропрограмм, что позволяет, в частности, реализовать в ЭВМ некоторой модели системы команд других моделей. Для этого достаточно либо заменить ПЗУ (вместе с содержимым), либо (что делают чаще) применить для хранения микропрограмм не ПЗУ, а обычное скоростное ЗУ. В этом последнем случае заменять хранимые в памяти микропрограммы можно в «домашних» условиях: подобную работу может выполнить персонал, эксплуатирующий ЭВМ. Загрузку микропрограмм в такое ЗУ часто осуществляют с портативного кассетного магнитофона — кассеты подготавливаются на заводе-изготовителе ЭВМ. Советская ЭВМ ЕС 1035 имеет два комплекта микропрограмм: с помощью одного в машине реализуется стандартная для этого типа машин система команд ЕС ЭВМ, а с помощью другого — система команд выпускавшейся ранее вычислительной машины «Минск-32». Любопытно, что эти машины — ЕС 1035 и «Минск-32» — разительно отличаются друг от друга и архитектурой, и системами команд, и способами представления информации. Различны даже длины машинных



слов: в ЕС ЭВМ — 32 разряда, а в «Минске-32» — 37 разрядов. Микропрограммная реализация на одной машине (ЕС 1035) системы команд другой машины («Минск-32») называется эмуляцией. Наличие средств эмуляции в ЕС 1035 позволяет использовать большой фонд прикладных программ, разработанных ранее для «Минска-32». Это весьма важно: стоимость разработки современных больших программных систем зачастую значительно превышает стоимость оборудования ЭВМ.

## «ГНОМЫ» И «ВЕЛИКАНЫ» В МИРЕ ЭВМ

**З**аводы вычислительной техники ежегодно выпускают тысячи машин, различных по назначению, возможностям, архитектуре и быстродействию. Разнообразие типов ЭВМ объясняется, прежде всего, разнообразием сфер применения вычислительных машин в народном хозяйстве. Средства обработки информации принято делить на следующие классы: микропроцессоры, микро-ЭВМ, мини-ЭВМ, универсальные ЭВМ средней производительности, универсальные ЭВМ большой производительности, сверхбыстродействующие вычислительные системы. Несмотря на то что четких границ между классами нет, тем не менее каждый класс характеризуется своим функциональным назначением и архитектурными особенностями.

### М и к р о п р о ц е с с о р ы

Хорошо освоенная технология производства больших интегральных схем позволила в начале 70-х годов создать качественно новые автономные устройства — *микропроцессоры*. Миниатюрный Центральный Процессор (размером, например,  $36 \times 15$  мм), подобно «взрослому» процессору, имеет собственную систему команд, содержит в своем составе АЛУ, УУ и различные внутренние регистры. Разрядность машинного слова микропроцессора обычно невелика — от 1 до 8 бит для разных типов микропроцессоров. С целью удлинения машинного слова (обычно до 8—16 бит) конструкторы стыкуют однотипные микропроцессоры (рис. 107). Выпуск микропроцессоров сопровождается производством некоторых вспомогательных БИС, выполняющих функции блоков микропрограммного управления, блоков прерывания, синхронизирующих устройств. Они позволяют легко достроить

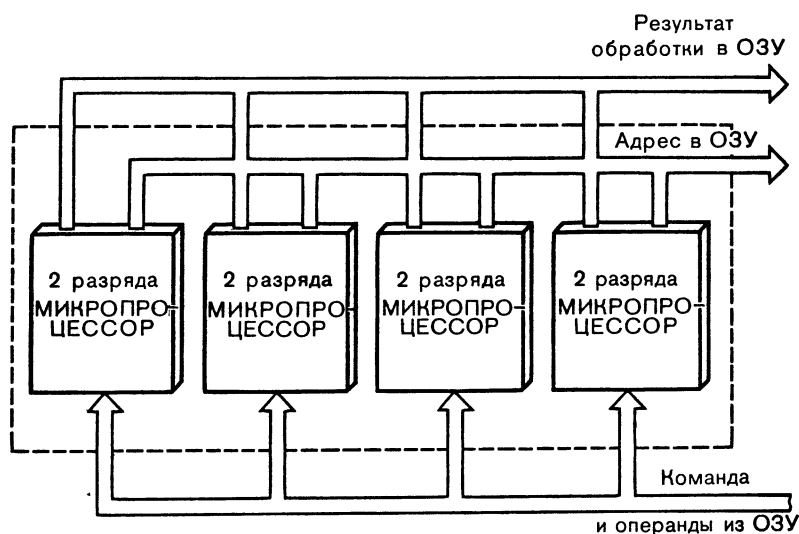
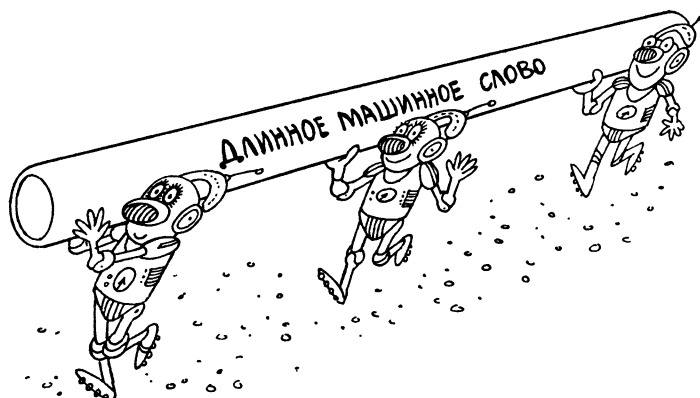


Рис. 107

группу микропроцессоров в сочетании с интегральным полупроводниковым ЗУ до завершенной машины — микро-ЭВМ. Микропроцессоры и вспомогательные БИС, образующие одно семейство, называют микропроцессорным комплектом. Отмечавшаяся уже эквивалентность программных и аппаратных средств отражает некоторую двойственность программ и машинных блоков. В связи

с этим можно вспомнить знаменитую формулу А. Эйнштейна  $E = Mc^2$ , описывающую двойственность энергии и массы физического тела. С появлением микропроцессоров конструкторы ЭВМ заговорили о «новой двойственности» программного обеспечения и аппаратуры. Способность микропроцессоров автономно выполнять обработку информации позволила перенести многие сложные функции, ранее осуществлявшиеся программным путем, в схемы больших вычислительных машин. Благодаря этому удалось значительно повысить «интеллектуальный уровень» отдельных блоков ЭВМ и машины в целом, открылась возможность реализовать параллельные вычисления.

## М и к р о - Э В М

Подлинным чудом интегральной технологии стало создание однокристалльной микро-ЭВМ: на пластинке из полупроводникового материала площадью приблизительно в  $1 \text{ см}^2$  удалось разместить Центральный Процессор, ОЗУ и другие устройства.

Размер машинного слова в однокристалльной микро-ЭВМ — от 8 до 16 бит, объем оперативной памяти — от 1 до 16 Кбайт, быстродействие — сотни тысяч операций в секунду.

К классу микро-ЭВМ относятся не только однокристалльные ЭВМ, но и упоминавшиеся уже микромашины, собираемые из микропроцессорных комплектов.

С момента появления первого микропроцессора — 1971 год — сменилось четыре (!) поколения микро-ЭВМ.

## М и н и - Э В М

Многие микро-ЭВМ представляют собой крохотные копии соответствующих типов минимашин. «Родственники» имеют одинаковые системы команд — это обеспечивает их программную совместимость. Правда, система команд минимашинны обычно богаче системы команд «младшего родственника»: все программы, написанные для микро-, могут быть выполнены на мини-ЭВМ; обратное утверждение справедливо не для всех программ. Производительность мини-ЭВМ достигает миллионов операций в секунду, обычная длина машинного слова —

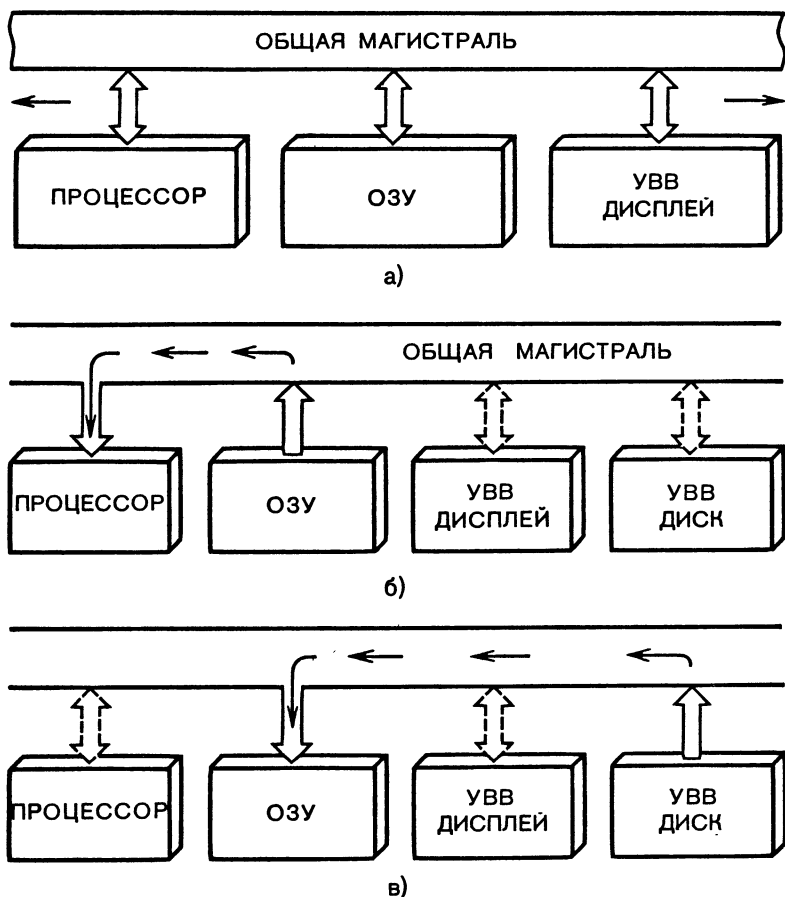


Рис. 108

16 бит, объем оперативной памяти — 32—64 Кбайт. Индивидуальные черты классов машин нигде не проявляются так ярко, как в способах организации ввода-вывода. Сравнительно простой способ ввода-вывода информации реализован в микро- и мини-ЭВМ. Основные блоки машины — Центральный Процессор, оперативную память, разнообразные УВВ — соединяют обычно между собой единой шиной, называемой общей магистралью. Ее используют для передачи информации от одного блока ЭВМ к другому (рис. 108, а). Одновремен-

но в общей магистрали может работать только один дуэт устройств. При этом одно из них служит источником информации, а другое — приемником. На рисунке 108, б изображено схематически считывание информации из ОЗУ в Процессор, а на рисунке 108, в — запись информации из УВВ (диск) в ОЗУ.

Регистрам УВВ и регистрам Центрального Процессора, как и ячейкам ОЗУ, присвоены адреса. Это позволяет выполнять операции с регистрами УВВ и Центрального Процессора точно так же, как с операндами из ОЗУ. В составе каждого из устройств мини- или микро-ЭВМ, подключаемых к общей магистрали, имеется несколько адресуемых по общей магистрали регистров. Это регистры команд, регистры состояния и буферные регистры данных (рис. 109). Буферные регистры предназначены для кратковременного хранения информации, которую необходимо выдать в общую магистраль или принять ее оттуда. Число и разрядность регистров данных в значительной мере определяются типом подключаемого к общей магистрали устройства. Регистр команд получает из общей магистрали от Центрального Процессора управляющую информацию о том, какую операцию (прочитать, записать, высветить символ на экране дисплея, перемотать магнитную ленту и т. п.) и как (откуда, куда, сколько) должно выполнить устройство. В регистре состояния концентрируется информация о состоянии устройства. Аппаратура управления (контроллер) постоянно следит за работой всех элементов устройства и фиксирует в регистре состояния важнейшие события деятельности устройства: «Включено» (электропитание, двигатель), «Занято» (предыдущей операцией), «Готово» (к выполнению предлагаемой операции). В этот же регистр поступают сведения о сбоях оборудования во время информационного обмена. Поскольку регистр состояния доступен из программы, его содержимое можно пересылать в Центральный Процессор для последующего анализа. К общей магистрали могут быть подсоединены и другие процессоры, близкие по архитектуре и системе команд. В этом случае центральным является тот процессор, который контролирует смену работающих на общей магистрали дуэтов. Описанную систему организации ввода-вывода имеют мини-ЭВМ таких типов, как СМ-3 и СМ-4. Эти модели программно совместимы: система команд СМ-3 является подмножеством системы команд СМ-4. Функциональные возмож-

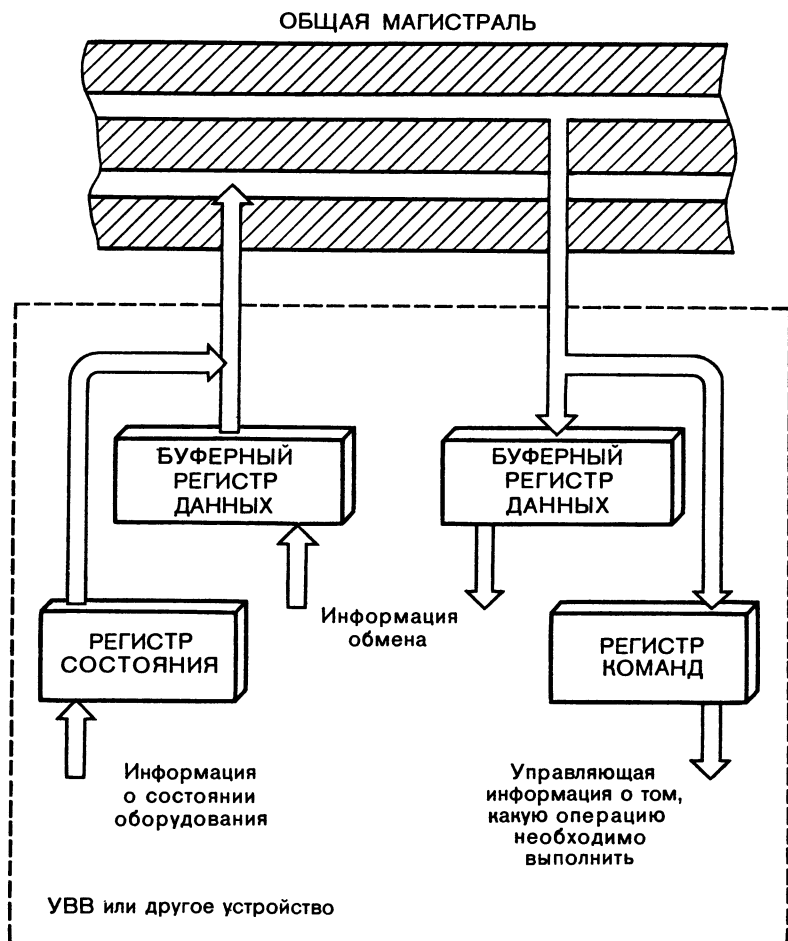


Рис. 109

ности мини-ЭВМ СМ-3 в несколько раз превосходят соответствующие технические характеристики микро-ЭВМ «Электроника-60», имеющей сходную с СМ-3 и СМ-4 архитектуру. Широко распространены такие представители класса мини-ЭВМ, как выпускавшиеся ранее М-6000, М-7000 и заменившие их в настоящее время СМ-1 и СМ-2. По системе команд и архитектуре эти машины резко отличаются от машин СМ-3, СМ-4: у них отсутствует общая магистраль. Структурная схема мини-

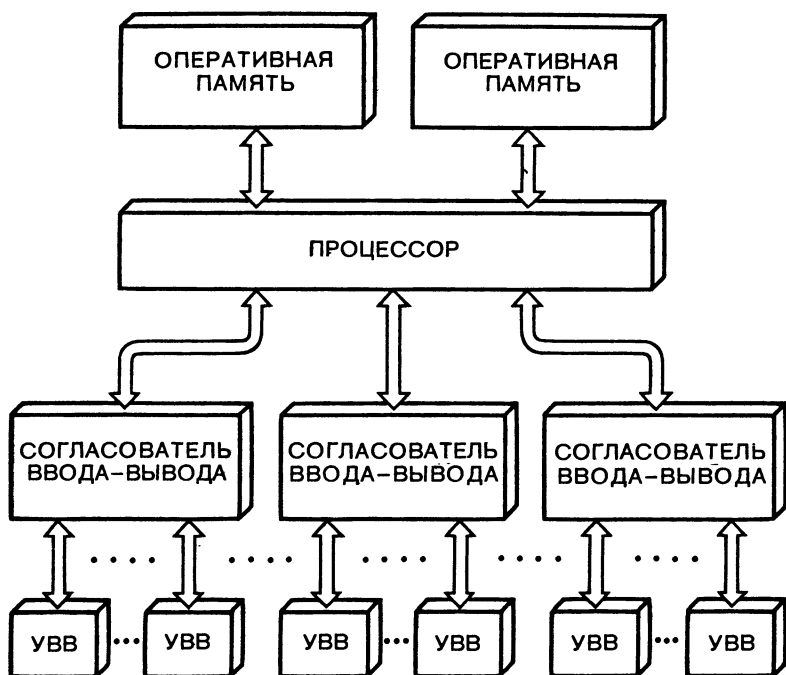


Рис. 110

ЭВМ СМ-2 приведена на рисунке 110. Общая черта всех рассмотренных мини-ЭВМ — модульный принцип их организации. Это дает возможность пользователю сравнительно просто видоизменять состав и конфигурацию вычислительной системы и, в частности, создавать специализированные комплексы, ориентированные на решение определенного класса задач. В качестве основных модулей используются процессоры, ферритовые и полупроводниковые ЗУ, ВЗУ — магнитофоны и накопители на магнитных дисках. Специфика применения мини-ЭВМ определяет вид модулей УВВ: перфоленточные устройства ввода-вывода, печатающие устройства, дисплеи и, что очень важно, различные Устройства Связи с Объектом (УСО), т. е. устройства, с помощью которых мини-ЭВМ может контролировать различные технологические процессы и управлять ими. Вычислительные комплексы, в состав которых наряду с ЭВМ входят устройства связи с системами контроля и управления в энергетике, метал-

лургии, на транспорте и т. п., называют управляющими. Применение ЭВМ в таких комплексах связано с тем, что контроль за сложным поведением управляемого объекта может находиться за пределами физических возможностей человека. Высокая надежность мини-ЭВМ, низкая стоимость, а также их небольшие габариты и вес способствуют повышению экономической эффективности управляющих вычислительных комплексов. Сегодня мини-ЭВМ обучены сотням специальностям: они управляют конверторной плавкой на металлургическом заводе, продают билеты и резервируют места на самолеты Аэрофлота, регулируют уличное движение и управляют энергосистемой страны.

### Универсальные ЭВМ

Микро- и мини-ЭВМ специализируются на обработке логической и символьной информации. Сознательно упрощая архитектуру микро- и минимашин, их разработчики исключили все элементы, предназначенные для выполнения вычислительных операций. Короткое машинное слово и уменьшенные возможности АЛУ не позволяют эффективно выполнять на микро- и мини-ЭВМ арифметические вычисления. А вот машины, принадлежащие к классу универсальных ЭВМ, одинаково успешно обрабатывают как логическую, так и арифметическую информацию. Большинство универсальных вычислительных машин объединены в семейства ЭВМ разной производительности. Машины группируются в семейства единым принципом построения системы команд, единым способом связи с УВВ и, следовательно, общим для всего семейства набором УВВ. Типичным семейством универсальных машин является ЕС ЭВМ (единая система ЭВМ), насчитывающая в своем составе 18 типов ЭВМ разной производительности — малой, средней и большой. Машины имеют единую архитектуру и программно совместимы «снизу вверх»: программы, написанные для младших моделей, успешно выполняются на старших. *Малые* ЕС ЭВМ по своим техническим характеристикам пересекаются с классом минимашин. Производительность их колеблется от 20 до 100 тысяч операций в секунду, объем ОЗУ — от 64 до 152 Кбайт. Парк УВВ — печатающие устройства (широкоформатная печать и электрическая пишущая машинка), перфоленточный и перфокарточный ввод-вывод. Среди ВЗУ — магнитофоны и (в не-



большом количестве) накопители на магнитных дисках: объем памяти на магнитных дисках — от 15 до 100 Мбайт. В семействе ЕС ЭВМ к группе малых машин относятся ЕС 1010, ЕС 1015, ЕС 1020, ЕС 1022, ЕС 1025. Производительность *средних* ЭВМ находится в диапазоне от 80 до 400 тысяч операций в секунду, объем ОЗУ — от 128 до 1000 Кбайт, объем внешней памяти на магнитных дисках — от 50 до 200 Мбайт. В состав внешних устройств, помимо обычных УВВ, входят дисплеи и графопостроители. К машинам этой группы ЕС ЭВМ относятся ЕС 1030, ЕС 1035 и ЕС 1040. Машины *большой* производительности — ЕС 1045, ЕС 1050, ЕС 1052, ЕС 1055, ЕС 1060, ЕС 1065 — имеют быстродействие от 500 тысяч до 6 миллионов операций в секунду, объем ОЗУ — от 500 тысяч до 16 миллионов байт, объем внешней памяти на магнитных барабанах и дисках — несколько сотен мегабайт. К классу универсальных ЭВМ большой производительности следует отнести еще недавно выпускавшуюся машину БЭСМ-6 — бывшего флагмана советской вычислительной техники. Принципы, заложенные в основу ее архитектуры, не потеряли своего значения до сих пор. Первая особенность этой машины состояла в том, что для достижения необходимого баланса между высокоскоростным Центральным Процессором и сравнительно медленным ОЗУ в ней был принят принцип «распараллеливания»: ОЗУ состоит из восьми блоков, допускающих одновременную выборку информации (см. раздел «Архитектура ЭВМ», рис. 2). Последовательно выполняемые команды программы и их операнды размещаются в разных блоках памяти. Это позволяет вести обработку информации практически в темпе работы Центрального Процессора. Второй особенностью БЭСМ-6 является способность аппаратуры Центрального Процессора просматривать команды вперед и заблаговременно подготавливать предъявляемые ОЗУ запросы операндов или команд. Буферные регистры адресов накапливают очередь заказов. Это позволяет сгладить неравномерность поступления запросов к памяти и тем самым повысить эффективность ее использования. Третья особенность — использование СОЗУ. Основное назначение этого устройства — экономия обращений в ОЗУ. СОЗУ представляет собой совокупность регистров. Управление СОЗУ осуществляется таким образом, что часто используемые группы команд или операнды оказываются в СОЗУ и готовы к немедленному использованию в УУ или АЛУ

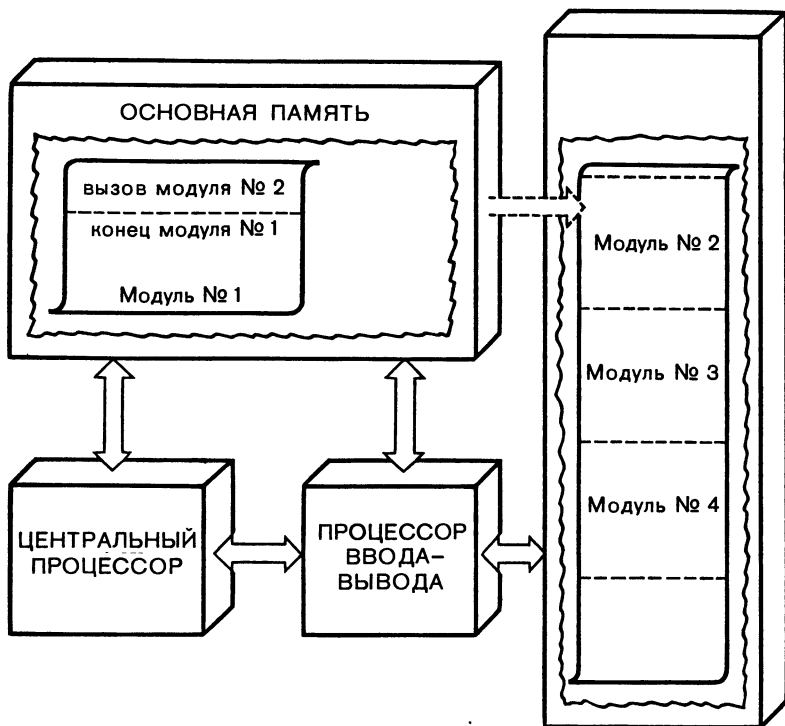


Рис. 111

Центрального Процессора. Регистровое СОЗУ позволяет в ряде случаев экономить до 60% всех обращений к памяти. Эти структурные особенности БЭСМ-6 получили название принципа «водопровода». Если проследить время, за которое частица воды проходит по участку водопровода, то оно может оказаться сравнительно большим, хотя скорость потока на выходе может быть высокой. Аналогично если измерить время от начала выполнения машинной команды до ее завершения, то для каждой команды это время будет достаточно велико, однако наличие СОЗУ и работа буфера адресов, просмотр команд вперед и параллелизм выполнения отдельных операций приводят к тому, что скорость обработки информации очень высока: быстроедействие БЭСМ-6 — 1 миллион операций в секунду. Четвертой архитектурной особенностью БЭСМ-6 (этой особенностью обладают

и последние, старшие модели ЕС ЭВМ) является наличие *виртуальной* памяти. В машинах, имеющих обычную иерархическую структуру памяти (СОЗУ — ОЗУ — ВЗУ), программы и данные перед выполнением размещаются в ВЗУ. Программы или данные пересылаются из ВЗУ в ОЗУ по мере того, как в них нуждается Центральный Процессор. Для этого программист разделяет свою программу на программные модули, каждый из которых может разместиться в ОЗУ. В начале работы программы вызывается программный модуль № 1. Закончив работу, он вызывает модуль № 2 и т. д. (рис. 111). Разделение программы на модули и размещение модулей в ВЗУ относится к обязанностям программиста. Кроме этого, программист должен организовать в своей программе пересылку программных модулей из ВЗУ в ОЗУ. Виртуальная память снимает все эти заботы с программиста, дает возможность строить программы в предположении, что машина имеет ОЗУ, совпадающее по объему с полным ВЗУ. Каждый адрес, к которому обращается Центральный Процессор, преобразуется с помощью специальной аппаратуры из так называемого виртуального адреса в реальный адрес ОЗУ. Виртуальная память создает у пользователя-программиста иллюзию того, что в его распоряжении имеется емкое ОЗУ, даже если в действительности основная память ЭВМ невелика: например, ЕС 1060, имеющаяся ОЗУ объемом 2 Мбайт, благодаря механизму виртуальной памяти предоставляет пользователям 16 Мбайт основной памяти. Для материализации подобных миражей Центральный Процессор содержит небольшое скоростное ЗУ. В нем хранится так называемая таблица страниц, похожая на книжное оглавление, помогающее быстро определить место главы, в которой находится искомая информация. Если нужная информация отсутствует в ОЗУ (об этом Центральный Процессор может узнать, анализируя таблицу страниц), то вырабатывается запрос к Операционной Системе для извлечения необходимой информации из ВЗУ и пересылки ее в ОЗУ перед тем, как будут продолжены вычисления. Рисунок 112 схематически изображает виртуальную память так, как она представляется программисту-пользователю; скрытая от него «небольшая» основная память содержит только непосредственно используемую в данный момент (активную) информацию.

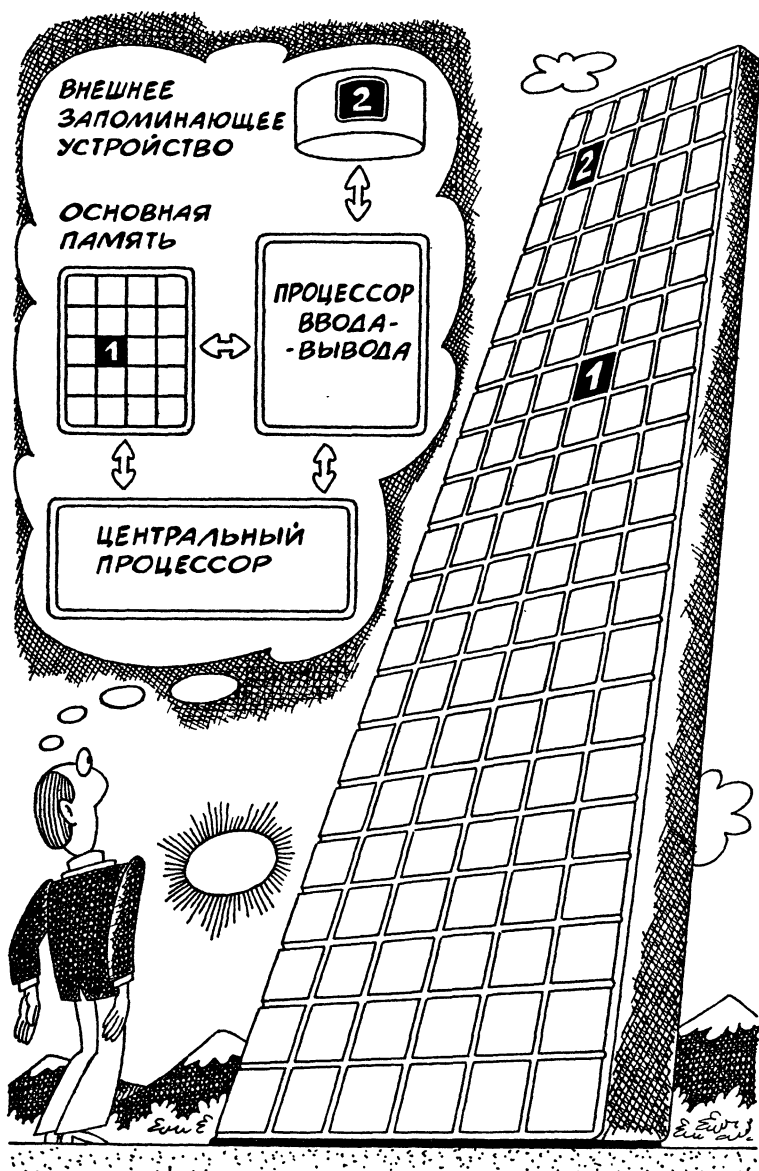


Рис. 112

## Сверхбыстродействующие вычислительные системы

Процессор Ввода-Вывода значительно видоизменил классическую неймановскую архитектуру вычислительной машины (см. раздел «Архитектура ЭВМ»). Теперь Центральный Процессор освобожден от необходимости выполнять операции ввода-вывода и лишь инициирует их выполнение. Тем самым увеличилась роль основной памяти ЭВМ как организационного центра вычислительной системы. Реализованный аппаратно параллелизм, о котором шла речь в рассказе об архитектурных особенностях БЭСМ-6, ограничивался выполняемой командой и несколькими следующими по программе командами (просмотр вперед). Появление в архитектуре ЭВМ Процессоров Ввода-Вывода означало распараллеливание более крупных функций, выполняемых машиной в процессе обработки информации. По сути дела, используя общую основную память, по различным программам в машине одновременно работают несколько процессоров (рис. 113). Естественным развитием этой идеи было объединение в рамках одного комплекса не только нескольких Процессоров Ввода-Вывода, руководимых одним Центральным Процессором, но и других центральных процессоров. Так появились многопроцессорные вычислительные системы. Первоначально многопроцессорные системы создавались для повышения надежности вычислительных систем, управляющих технологическими процессами и различными установками, от состояния которых зависели здоровье и безопасность человека в таких прикладных областях, как ядерная энергетика, космические исследования, химическая промышленность и т. п. Следующим этапом было появление таких многопроцессорных вычислительных комплексов, которые обеспечивали не только высокую надежность работы, но и предельную производительность. В качестве примера рассмотрим архитектуру многопроцессорного вычислительного комплекса МВК «Эльбрус» (рис. 113). Построенный по модульному принципу «Эльбрус» включает модули центральных процессоров, оперативной памяти, процессоров ввода-вывода, процессоров приема-передачи данных, ВЗУ на магнитных барабанах, дисках и лентах, а также разнообразные УВВ. Модули центральных процессоров, оперативной памяти и процессоров ввода-вывода связаны между собой при помощи центрального коммутатора. Процессоры

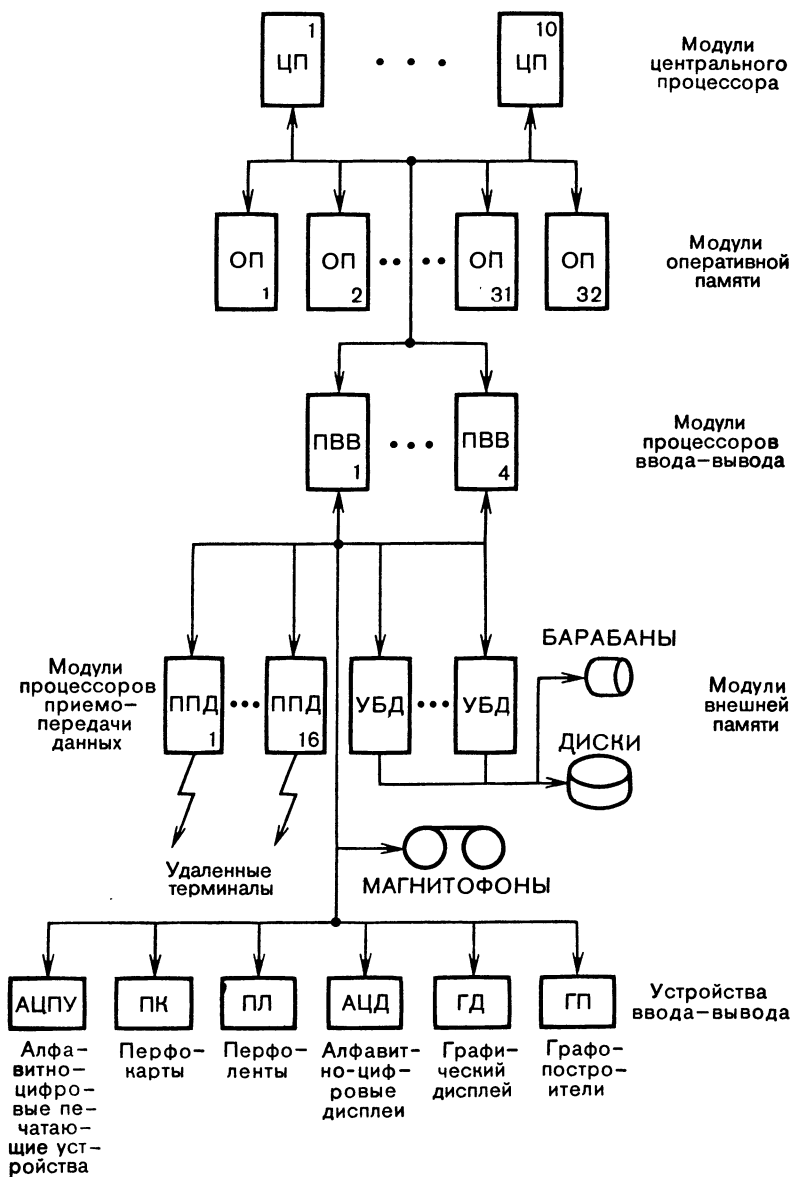


Рис. 113

приема-передачи данных, ВЗУ и УВВ подключаются к центральной части системы через процессоры ввода-вывода. Информация от удаленных терминалов (комплекс допускает подключение 2560 терминалов) поступает в центральную часть МВК через процессоры приема-передачи данных. Модули МВК «Эльбрус» функционируют параллельно и независимо друг от друга. Однотипность модулей позволяет Операционной Системе при появлении хотя бы одиночной ошибки в работе оборудования отключить неисправный модуль без ощутимого ущерба для результативности МВК в целом. После устранения неисправностей модуль может быть снова включен Операционной Системой в рабочую конфигурацию. МВК «Эльбрус» имеет практически неограниченную виртуальную память. Для него характерна полная автономия ввода-вывода. В основу организации вычислений положен стек. Это позволяет вызывать операнды в самый последний момент перед выполнением операций и освобождать память сразу же после завершения вычислений. Информационными обменами как между оперативной памятью и ВЗУ, так и между парами УВВ управляют процессоры ввода-вывода. Они имеют буферную память, свои АЛУ и УУ и работают на фоне деятельности Центральных Процессоров, которые совместно с Операционной Системой составляют заявки на обращение к ВЗУ или УВВ. Выполняя очередную заявку, процессор ввода-вывода запускает УВВ или ВЗУ, реализует передачу данных, завершает обмен и корректирует список заявок. УВВ и ВЗУ могут работать с любым из четырех процессоров ввода-вывода и с любым соответствующим устройством управления (контроллером). Если существует несколько путей обмена информацией, то процессор ввода-вывода сам выбирает направления обмена по мере освобождения оборудования. Все основные функции, в том числе поиск и выбор свободного и исправного канала, реализуется в процессорах ввода-вывода аппаратно. Это позволяет экономить оперативную память и существенно разгружает Центральные Процессоры от прерываний со стороны УВВ и ВЗУ. В процессорах приема-передачи данных в отличие от процессоров ввода-вывода алгоритмы обмена, поиска и выбора реализуются программно, поскольку число терминалов, подключаемых к процессорам приема-передачи данных, может быть велико, а их типы разнообразны. Высокая эффективность использования оборудования в многопроцессорных вы-

числительных комплексах достигается тогда, когда удастся распараллелить выполнение отдельных частей работающей программы. Это совершенно необходимо при обработке векторов или матриц, при многократных обращениях к базам данных. Распознаванию параллельных участков программ и оптимальному распараллелива-

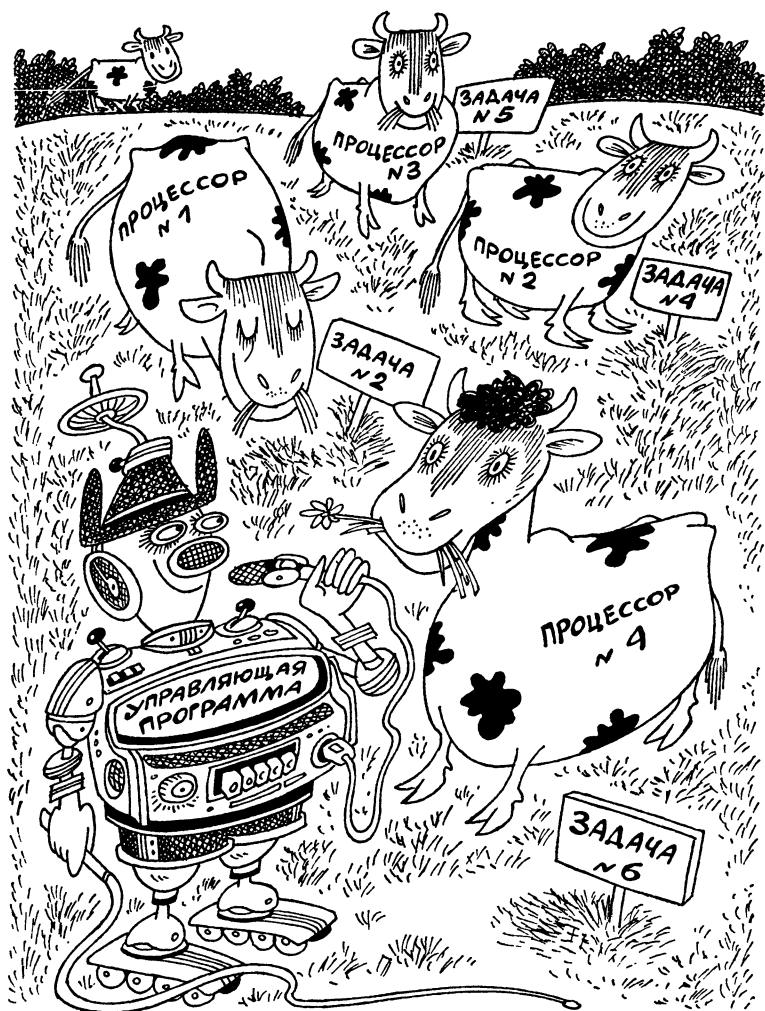


Рис. 114



нию сейчас посвящаются многочисленные исследования, и все же белых пятен в этой области, к сожалению, пока еще много. Впрочем, МВК работают эффективно, даже не используя распараллеливания программ. В этих условиях высокая загрузка каждого процессора обеспечивается общим большим пакетом программ, каждая из которых выполняется последовательно. Оцениваемая таким образом интегральная производительность МВК «Эльбрус» в полной его конфигурации достигает сотни миллионов операций в секунду. Один известный ученый как-то сказал, что он видит перспективу развития многопроцессорных вычислительных машин как огромное поле памяти, на котором, как коровы, пасутся процессоры. Большое стадо процессоров... (рис. 114).

Приведенная здесь классификация средств обработки информации, целью которой было рассмотрение архитектурных особенностей «гномов» и «великанов» мира ЭВМ, довольно условна. Многопроцессорные вычислительные комплексы имеют свою классификацию, связанную со способами объединения процессоров в комплексы.

## ЭВОЛЮЦИЯ ЭВМ

**С**о времени появления первых ЭВМ прошли десятилетия. За этот сравнительно короткий период конструкторы вычислительных машин, разработчики программного обеспечения и технологии микроэлектроники добились значительного повышения производительности ЭВМ, расширили возможности их применения в народном хозяйстве и улучшили условия работы пользователей. Количество усилий, прилагаемых создателями вычислительных машин, переходило в качество примерно один раз в 7 — 9 лет. Каждый из специалистов по различным проблемам вычислительной техники по-своему отмечал эти качественные переходы. Но все они называют четыре главных этапа развития ЭВМ — четыре поколения. Инженер-системотехник, занимающийся разработкой аппаратной части ЭВМ, охарактеризовал бы четыре поколения вычислительных машин так:

1-е поколение — это машины, построенные на электронных лампах, с быстродействием 10—20 тысяч операций в секунду;

2-е поколение — транзисторные машины, или машины на твердых схемах (твердые схемы — предшествен-

ницы интегральных схем с быстродействием в сотни тысяч операций в секунду);

3-е поколение — это машины на интегральных схемах, их быстродействие достигает миллионы операций в секунду;

4-е поколение — многопроцессорные машины на больших интегральных схемах (БИС), обладающие быстродействием в десятки миллионов операций в секунду. Подтверждая основные направления в развитии элементной базы ЭВМ — «электронная лампа — транзистор, диод — интегральные схемы малой и средней степени интеграции — большие интегральные схемы», технолог отмечает последовательное возрастание функциональных возможностей элементов от поколения к поколению. Например, если с помощью транзистора можно было реализовать всего одну простейшую функцию (И, ИЛИ, НЕ), то интегральные схемы малой и средней степени интеграции оказались способными выполнять десятки простейших функций. Чаще всего с помощью этих простейших функций непосредственно в корпусе интегральной микросхемы выполняется одна из сложных функций (сумматор, регистр, дешифратор и т.д.). Наконец, большие интегральные схемы могут выполнять совокупности сложных функций (микропроцессор, микро-ЭВМ). Если бы в этом импровизированном интервью принял участие программист, ведущий разработку программного обеспечения, то в классификации поколений ЭВМ он подчеркнул бы, по-видимому, иные черты:

1-е поколение — это машины, для которых создавались программы преимущественно на машинных языках;

2-е поколение — машины, имевшие первые трансляторы с языков программирования высокого уровня;

3-е поколение характеризуется появлением семейств программно совместных ЭВМ с системами прерываний и развитыми операционными системами;

4-е поколение — это машины, обеспечивающие коммунальное использование вычислительной техники; связанные в единую вычислительную сеть такие машины позволяют сотням и тысячам пользователей работать в режиме разделения времени (рис.115).

Следует сказать, что некоторые ЭВМ, относимые по технологической классификации — по составу элементной базы — к машинам второго поколения, обладают всеми структурными особенностями машин третьего поколения. В известном смысле разделение мира ЭВМ на поколения

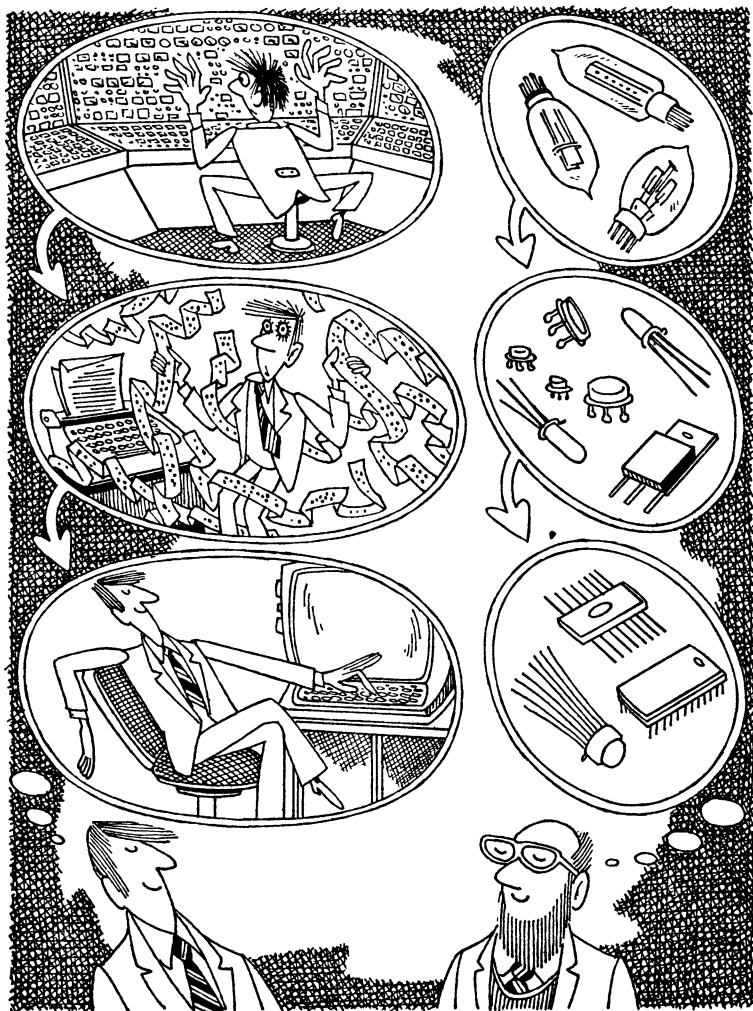


Рис. 115

достаточно условно. Типичные представители третьего поколения — машины из семейства ЕС ЭВМ. К четвертому поколению советских вычислительных машин относят многопроцессорный вычислительный комплекс «Эльбрус». Такая высокопроизводительная система, как МВК «Эльбрус», может служить для объединения вычислительных мощностей в едином центре, соединенном сов-

ременными линиями связи с многочисленными удаленными терминалами. Таким образом, речь идет о *коммунальном* использовании вычислительных машин, когда широкому кругу пользователей, как отдельных непрофессиональных потребителей информации, так и групп программистов-исследователей, предоставляются возможности одновременно связываться с центром обработки информации на любом расстоянии, выдавать машине задания и запросы и в срок получать выработанный ответ. Современной формой эксплуатации ЭВМ, ориентированной на коллективное использование машин, являются *вычислительные сети* — группы взаимосвязанных вычислительных машин, способных оперативно обмениваться информацией между собой. Такие сети организованы как совокупность соединенных друг с другом узлов, каждый из которых представляет собой одну или несколько ЭВМ. Поскольку входящие в сеть вычислительные машины могут связываться друг с другом, то пользователи конкретных ЭВМ получают доступ к ресурсам любых других ЭВМ сети, возможно даже не подозревая этого. Примерно так сейчас работает единая энергетическая система страны: включая вечером свет в своей комнате, мы не задумываемся над тем, откуда в наш дом пришла электроэнергия — с Красноярской ГЭС или с Нововоронежской атомной.

Вычислительная сеть имеет ряд преимуществ перед простой группой автономно работающих ЭВМ:

- общая надежность сети намного выше надежности любой из ее частей;

- если один узел сети перегружен работой, то часть работы автоматически перераспределяется между другими менее загруженными узлами;

- сеть делает возможным обращение к большим быстродействующим машинам и уникальным УВВ, быть может удаленным на большое расстояние от пользователя.

Вычислительные сети — новая развивающаяся форма использования ЭВМ. Возможно, что современные сети ЭВМ являются предвестниками будущих распределенных вычислительных машин. Различие между сетью и распределенной машиной определяется, прежде всего, размерами, расположением и функциями узлов, а также степенью их взаимодействия. Если все компоненты системы малы, зависимы друг от друга и ограничены небольшим пространством, можно считать их частями одной вычис-

лительной машины. Такое видоизменение идеи вычислительной сети обязано своим появлением развитию технологии больших интегральных схем, в частности микропроцессоров и микро-ЭВМ. В том случае, когда компоненты вычислительной системы весьма велики и независимы, их можно рассматривать как разные ЭВМ. Возможно смешение форм — многомашинные и многопроцессорные системы, состоящие из ряда специализированных устройств. Вычислительная сеть, помимо обычных компонентов — универсальных ЭВМ, мини- и микромашин, может содержать специализированные системы, например процессоры для обработки изображений, процессоры для решения определенных классов задач математической физики, экономики и т. д. Такие вычислительные системы называют *неоднородными средами с распределенными вычислениями*.

Перспективу будущего компьютера нельзя нарисовать в виде единой универсальной модели ЭВМ: уж очень сильно различаются тенденции развития современной вычислительной техники. Народному хозяйству будут нужны и супер-ЭВМ, концентрирующие гигантские вычислительные мощности, и децентрализованные системы из большого числа территориально удаленных ЭВМ, объединенных в информационные сети, и настольные и даже карманные микро-ЭВМ с широкими возможностями решения прикладных задач — персональные компьютеры. Такая большая семья, состоящая, как и полагается в семье, из «старших» и «младших», будет представлять собою тщательно сбалансированную по архитектуре систему вычислительных машин, что позволит каждому из пользователей привести в действие ее огромный вычислительный потенциал. При этом вопросы рационального использования этого потенциала будут эффективно решать методами программирования. Таким путем удастся снизить трудоемкость работ по производству, обслуживанию и сопровождению ЭВМ.

В последней четверти двадцатого века индустрия обработки и хранения информации становится одним из наиболее важных национальных ресурсов наряду со всем тем, что нас «кормит, греет, одевает».

Эффективность использования информационных ресурсов во все большей степени будет определять экономическую мощь страны.

## ЧТО ЧИТАТЬ ДАЛЬШЕ?

**Е**сли, прочитав эту книгу, читатель заинтересовался проблемами вычислительной техники, устройством ЭВМ, историей их создания и перспективами развития, он наверняка возьмет в руки следующую книгу. К счастью, сейчас написано уже немало хороших книг о вычислительных машинах.

Все большее число людей стало знакомиться с основами вычислительной техники, а программирование постепенно превращалось из рабочего инструмента специалиста в элемент общей культуры современного человека. Вместе с тем история развития средств инструментального счета известна значительно в меньшей степени. Книга Р. С. Гутера и Ю. Л. Полунова «От абака до компьютера» (М.: Знание, 1981) популярно рассказывает о развитии идеи инструментальных вычислений, об истории дискретной техники.

Книга Л. Н. Королева «Структура ЭВМ и их математическое обеспечение» (М.: Наука, 1981) представляет собой систематический курс, излагающий устройство ЭВМ будущим специалистам в области вычислительной техники. Книга содержит обзор отечественных и зарубежных универсальных вычислительных машин. Здесь обсуждены архитектурные особенности некоторых наиболее интересных ЭВМ, их математическое обеспечение, а также влияние идей программирования на архитектурные решения в конструкции машины. Книга дает общее представление о тех направлениях, в которых идет развитие ЭВМ.

Широкие социальные проблемы внедрения вычислительной техники в жизнь человеческого общества описывает книга Г. Р. Громова «Национальные информационные ресурсы: проблемы промышленной эксплуатации» (М.: Наука, 1984). В центре этих проблем — роль персональных ЭВМ в современном мире.

Комплексное изложение проблем автоматизированной обработки и хранения информации в машинном представлении содержит книга выдающегося советского ученого и организатора науки академика Виктора Михайловича Глушкова «Основы безбумажной информации» (М.: Наука, 1982). Книга может быть использована для первоначального ознакомления с вопросами применения вычислительной техники.

Вопросы практического применения микропроцессоров и микро-ЭВМ с достаточной для начинающего простотой и ясностью рассматриваются в следующих книгах:

1. В книге Д. Гивоне и Р. Россера «Микропроцессоры и микрокомпьютеры: вводный курс» (М.: Мир, 1983) подробно рассматривается архитектура микропроцессоров, даны примеры программ и интерфейсов. Это — систематическое и полное введение в логическую структуру и программирование микропроцессоров.

2. Двухтомник Дж. Уокерли «Архитектура и программирование микро-ЭВМ» (М.: Мир, 1984) замечателен тем, что в нем органически слиты программные и технические аспекты микропроцессорной техники. В книге рассматриваются основные типы современных микро-ЭВМ.

3. Простота изложения сложных вопросов теории цифровых устройств и методов цифровой обработки данных отличает книгу «Микро-ЭВМ» под редакцией А. Дирксена (М.: Энергоиздат, 1982), в которой излагаются общие принципы организации микропроцессоров и микро-ЭВМ.

4. Хорошо построена в методическом отношении, прекрасно иллюстрирована и снабжена большим количеством продуманных упражнений книга Ч. Гилмора «Введение в микропроцессорную технику» (М.: Мир, 1984).

# **СОДЕРЖАНИЕ**

## **ПРЕДИСЛОВИЕ**

**3**

## **АРХИТЕКТУРА ЭВМ**

**5**

## **ДВОИЧНАЯ АРИФМЕТИКА**

**11**

## **АЛГЕБРА БУЛЯ**

**17**

## **СИГНАЛЫ, ПОМЕХИ И ГОНЦЫ**

**23**

## **АЗБУКА ЦИФРОВОЙ ЭЛЕКТРОНИКИ**

**27**

## **ПАМЯТЬ**

**45**

## **КАК РАБОТАЕТ ПРОЦЕССОР?**

**58**

## **ВВОД-ВЫВОД ЭВМ**

**65**

## **ЭКСКУРСИЯ В СИСТЕМУ КОМАНД**

**89**

## **НЕКОТОРЫЕ ПОДРОБНОСТИ О СИСТЕМЕ КОМАНД ЕС ЭВМ**

**97**

## **СЛОВО СОСТОЯНИЯ ПРОГРАММЫ**

**104**

## **СИСТЕМА ПРЕРЫВАНИЙ**

**107**

## **ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ**

**115**

## **МИКРОПРОГРАММИРОВАНИЕ**

**131**

## **«ГНОМЫ» И «ВЕЛИКАНЫ» В МИРЕ ЭВМ**

**136**

## **ЭВОЛЮЦИЯ ЭВМ**

**152**

## **ЧТО ДЕЛАТЬ ДАЛЬШЕ?**

**157**



*АЛЕКСАНДР НИКОЛАЕВИЧ  
САЛТОВСКИЙ  
ЮРИЙ АБРАМОВИЧ ПЕРВИН*

## **КАК РАБОТАЕТ ЭВМ**

---

Зав. редакцией *Р. А. Хабиб*  
Редактор *Н. А. Песина*  
Мл. редактор *Л. Е. Козырева*  
Художественный редактор *Е. Н. Карасик*  
Художники *Г. А. Алексеев, Е. С. Шабельник*  
Технический редактор *М. М. Широкова*  
Корректор *О. Н. Дьячкина*

ИБ № 9374

Сдано в набор 14.02.86. Подписано к печати 06.08.86. Формат 84 × 108<sup>1</sup>/<sub>32</sub>. Бумага офсетная № 2. Гарнит. литер. Печать высокая. Усл. печ. л. 8,40. Усл. кр.-отт. 8,82. Уч.-изд. л. 8,46. Тираж 150 000 экз. Заказ № 1124. Цена 30 коп.

Ордена Трудового Красного Знамени издательство «Просвещение» Государственного комитета РСФСР по делам издательств, полиграфии и книжной торговли.  
129846, Москва, 3-й проезд Марьиной рощи, 41.

Ярославский полиграфкомбинат Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 150014, Ярославль, ул. Свободы, 97.

30 к.

