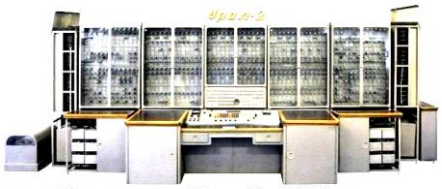
Глава 13. Понятие о мультипрограммном режиме работы

… Компьютеры, видимо, представляют собой наиболее интеллектуальное из всех человеческих изобретений, поэтому и ремесло программиста является самым сложным из всех известных.

Джоэл Бирнбаум,
отдел экспериментальных систем IBM,
затем вице-президент Hewlett-Packard

Надо отметить, что самые первые программисты работали непосредственно за пультом ЭВМ, в своеобразном диалоговом режиме. Правда, в отличие от персональных ЭВМ, работа была не в привычном нам сейчас графическом или текстовом интерфейсе, а непосредственно в двоичном коде. Лампочки на пульте ЭВМ высвечивали значения всех регистров (горит – единица, не горит – ноль), для удобства лампочки разбивались на группы по три или четыре, что позволяло «в уме» работать с машинными словами в 8-й или 16-й системе счисления. Специальная кнопка включала трассировку программы, при этом машина останавливалась после выполнения каждой команды. На особом регистре можно было набрать двоичный адрес останова (break point), при попадании в который машина останавливала свою работу и т.д. Всё это очень напоминало работу в первых отладчиках.


Компьютер Урал-2, 1959 г

Время было весёлое, ЭВМ содержала порядка 20000 электронных ламп, в среднем каждые полчаса одна лампа перегорала и приходилось звать дежурного инженера для ремонта 😐. Из-за этого каждые 15-20 минут нужно было делать так называемую контрольную точку, т.е. запоминать текущее состояние программы (на перфокартах или, в лучшем случае на магнитной ленте), чтобы можно было возобновить счёт с этой точки при аварийном останове ЭВМ. Кроме того, из за частых сбоев в работе самой аппаратуры приходилось каждый такой участок программы просчитывать дважды (а в важных рассчётах и трижды), сверяя результаты между собой.

Затем из экономических соображений такой режим работы стал нерациональным, так как программисты слишком долго искали и исправляли ошибки в своих программах или анализировали результаты расчётов, и в это дорогая ЭВМ простаивала. Тогда и появился новый, пакетный режим работы, и программистов перестали пускать в машинный зал за пульт ЭВМ 😐.[1]

Пакетный режим (batch mode) подразумевал, что подлежащие счёту программы собираются в некоторый «пакет», для ЭВМ первых поколений это был деревянный или металлический ящик, наполненный программами, каждая программа была в отдельной пачке картонных перфокарт, скрепленной резинками ⚠.[2] Операторы ЭВМ брали эти ящики, несли в машинный зал и по очереди вводили программы в компьютер. Программистам, «отлучённым» от пульта ЭВМ, теперь приходилось вставлять в свои программы многочисленные отладочные выдачи на печать. Анализируя потом эти выдачи, они пытались найти и исправить ошибки, после чего опять отдавали свою программу на счёт. За день многие ухитрялись просчитать свою программу несколько раз (надо было иметь хорошие отношения с операторами ЭВМ 🙂). А ночью на счёт ставились уже отлаженные программы…

Сейчас пакетный режим используется в основном на супер-ЭВМ, где запущенные программы выполняются на выделенных им ресурсах (ядрах) пока не закончатся или не исчерпают отведённое им время, без того, чтобы эти ресурсы передавались другим программам.

Одним из принципов фон Неймана, как известно, является принцип последовательного выполнения команд программы. Более того, архитектура машин фон Неймана предполагает, что последовательно выполняются не только команды текущей программы, но также и сами эти программы. Другими словами, пока одна программа полностью не заканчивается, следующая программа не загружа-

---

[1] Иногда исключения делались в ночное время, когда компьютер был более свободным, так что у программистов были ночные смены с недовольным и заспанным дежурным инженером для ремонта ЭВМ.

[2] Большие программы могли состоять из порядка тысячи перфокарт, так что для них требовался отдельный ящий 🙂.

ется в память и не начинает выполняться. Именно так и работали первые ЭВМ (вспомните, как работала описанная ранее учебная ЭВМ УМ-3). Сейчас познакомимся с весьма важным понятием – мультипрограммным (иногда говорят, многопрограммным) режимом работы ЭВМ.

Мультипрограммный режим работы (multiprogramming) означает, что в оперативной памяти компьютера одновременно находятся несколько *независимых* друг от друга и *готовых к счёту* программ пользователей, времена выполнения которых могут перекрываться.[1] Независимость программ означает, что они автоматически не обмениваются между собой данными в процессе счёта (через общие секции данных, файлы или как-нибудь ещё), т.е. они одновременно не решают одну общую задачу. Стоит также заметить, что при мультипрограммном режиме работы в памяти ЭВМ одновременно могут находиться не только программы разных пользователей, но и несколько независимых программ одного пользователя.

Как Вы уже наверно знаете, компьютеры принято делить на поколения. Мультипрограммный режим работы появился только на ЭВМ, начиная с конца 2-го и начала 3-го поколения, на первых компьютерах его не было [3]. Сначала предстоит разобраться, а для чего вообще может потребоваться, чтобы в памяти одновременно находилось несколько программ пользователей. Этот вопрос вполне естественный, так как раньше у большинства компьютеров был только один (центральный) процессор, так что одновременно могли выполняться команды только одной программы, а остальные программы в это время будут просто занимать место в оперативной памяти и «ничего не делать».

Частично уже была обоснована необходимость присутствия в оперативной памяти нескольких программ, когда изучалась система прерываний. Как правило, при возникновении прерывания процессор производит автоматическое переключение на некоторую другую программу, которая тоже, конечно, должна при этом находиться в оперативной памяти. Здесь, однако, можно возразить, что все программы, на которые производится автоматическое переключение при прерывании, являются системными программами (входят в операционную систему), а при определении мультипрограммного режима работы особо подчеркивалось, что в оперативной памяти могут одновременно находиться несколько разных программ обычных пользователей.

На ЭВМ первых поколений «обычным» пользователям разрешалось писать свои собственные процедуры-обработчики прерываний, однако в операционных системах современных ЭВМ это, как правило, запрещено. Причина такого запрета будет понятна из дальнейшего изложения мультипрограммного режима работы ЭВМ. Вместо этого, программа пользователя может попросить операционную систему при возникновении исключительной ситуации в своей программе не вызывать стандартную процедуру ОС обработки этого события, а предварительно вызвать свою собственную процедуру, которая попытается сама разобраться в возникшей ошибке и попробует исправить её (см. разд. 6.13.1).

Рис. 13.1. Одновременное нахождение в оперативной памяти нескольких программ разных пользователей.

Следует указать две основные причины, по которым может понадобиться мультипрограммный режим работы. Во-первых, может потребоваться одновременно выполнять несколько программ. На-

[1] Вообще говоря, физически эти программы, могут присутствовать в оперативной памяти не целиком. Во-первых, они могут использовать изученную ранее схему динамической загрузки библиотек, и, во-вторых, работать на так называемой виртуальной памяти, при этом некоторые части программы могут временно отсутствовать в оперативной памяти, находясь в специальном файле подкачки (swap file) на внешней памяти.

пример, это могут быть программы, которые в диалоговом режиме работают с разными пользователями (программисты Вася и Петя одновременно с разных терминалов (разных консолей), подключенных к одной ЭВМ, отлаживают свои программы, см. рис. 13.1).

Правда, здесь имеет место уже упомянутая ранее трудность: так как процессор на компьютере может быть только один, то в каждый момент времени может выполняться или программа Васи, или программа Пети (ну, или служебная программа операционной системы при обработке прерывания). Эта трудность преодолевается введением специального режима работы ЭВМ – режима разделения времени (Time-sharing), который является частным случаем мультипрограммного режима.

Во многих операционных системах режим разделения времени называется режимом *вытесняющей* (preemptive) *многозадачности*. Имеется ввиду, что некоторый процесс (задача) по истечению кванта времени принудительно вытесняет (заменяет) текущий процесс, выполняющийся в настоящее время. Более ранние ОС работали в режиме *не вытесняющей* многозадачности, при этом новый процесс начинал выполняться, только если прежний добровольно отдавал управление (например, ожидая ввода из файла) или заканчивался.

В режиме разделения времени, используя сигналы прерывания от встроенных в компьютер часов (таймера), служебная программа-диспетчер переключает процессор с одной задачи пользователя на другую по истечении определенного кванта времени (time slice), обычно порядка единиц или десятков миллисекунд. В таком режиме разделения времени (в русскоязычной литературе этот режим иногда метко называли *коммунальным* использованием ресурсов ЭВМ) и у Васи, и у Пети создаётся иллюзия, что только его программа все время считается на компьютере (правда, почему-то медленно 🙁). Механизм переключение задач подробно разбирался в разд. 7.1.

На рис. 13.2 показана временна́я диаграмма переключения работы процессора ЭВМ между находящимися в памяти программами.

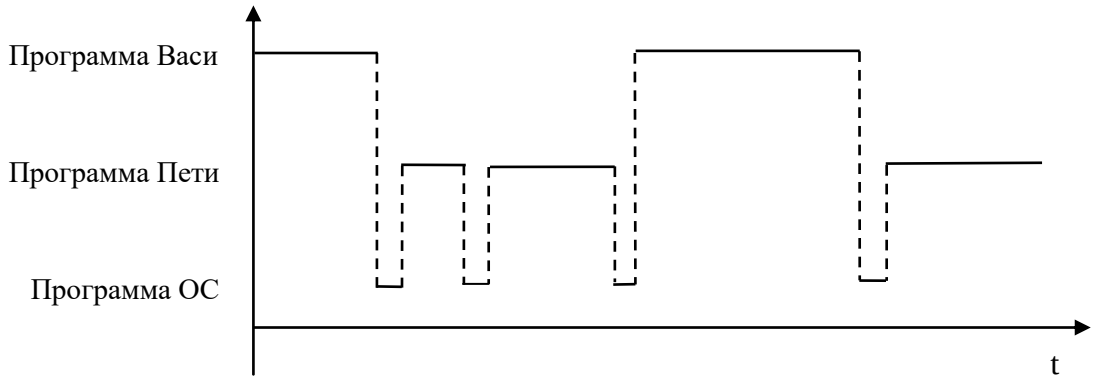


Рис. 13.2. Диаграмма загрузки процессора ЭВМ (t – время)

Если отвлечься от несколько надуманного примера с Васей и Петей, то можно отметить, что потребность в таком псевдо-одновременном счёте нескольких программ на компьютере с одним центральным процессором весьма распространена. Пусть, например, наш компьютер предназначен для управления несколькими различными химическими реакторами на каком-нибудь заводе, или обслуживает запросы сразу многих абонентов в библиотеке и т.д.[1] Заметим, что даже если центральных процессоров на компьютере несколько (сейчас даже на персональных ЭВМ, планшетах и смартфонах широко распространены многоядерные процессоры), но одновременно находящихся в памяти задач может быть много больше, чем процессоров, поэтому этот вопрос по-прежнему актуален.[2]

Другая причина широкого распространения мультипрограммного режима заключается в следующем. Наряду с главной частью – процессором и оперативной памятью – в компьютере существует и большое количество так называемых периферийных (внешних) устройств, это диски, клавиатура, мышь, печатающие устройства, сетевые карты и т.д. (см. рис. 13.3). Все эти периферийные устройства предназначены для связи центральной части машины с «внешним миром», и работают значительно медленнее, чем процессор и оперативная память. Имеется в виду, что все они значительно медленнее манипулируют данными. Например, за то время, за которое лазерный принтер напечатает на

[1] Предполагается, что экономически нецелесообразно использовать в этом случае отдельные ЭВМ.

[2] Вызвав диспетчер задач операционной системы можно увидеть, что задач (процессов) много больше, чем ядер (правда, бóльшая часть из них системные и находятся в состоянии ожидания).

бумаге всего один символ, оперативная память способна выдать процессору миллионы байт, а сам процессор способен за это время выполнить миллионы команд.

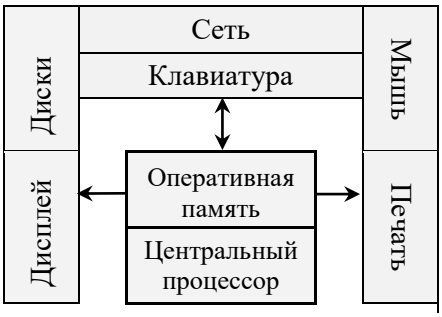


Рис. 13.3. Центральная и периферийная части компьютера.

Из этих соображений, очевидно, что в то время, когда по запросу некоторой программы производится обмен данными с медленными внешними устройствами, процессор, как правило, не сможет выполнять команды этой программы, т.е. будет простаивать. Например, рассмотрим случай, когда в программе Васи, написанной на Паскале, выполняются операторы:

```
Read(MyFile,X); Y:=X+1
```

Очевидно, что оператор присваивания `Y:=X+1` не сможет начать выполняться, пока из файла не будет прочитано значение переменной `X`. При описании языка высокого уровня обычно говориться, что при выполнении операции ввода/вывода вычислительный процесс *блокируется*, т.е. переводится в состояние ожидания завершения этого ввода/вывода (это блокирующий ввод/вывод).

Вот здесь нам и пригодится способность программы-диспетчера переключаться на выполнение других программ пользователей, тоже расположенных в оперативной памяти. Теперь, пока одна программа пользователя выполняет свои команды на центральном процессоре, другая может выводить свои данные на принтер, третья – читать массив с диска в оперативную память, четвертая – ждать ввода символа с клавиатуры и т.д. Правда, для того, чтобы обеспечить такую возможность, мало наличия на компьютере одной системы прерываний. Прежде всего, необходимо научить периферийные устройства компьютера работать параллельно и относительно независимо от центрального процессора. Действительно, вспомните, что в машине фон Неймана всеми операциями с внешними устройствами управлял именно центральный процессор по командам ввода/вывода, посылая им особые управляющие сигналы, которые изображались на схеме одинарными стрелками. Естественно, занимаясь этой работой (т.е. выполняя команду ввода/вывода), центральный процессор уже не мог выполнять команды какой-либо другой программы.

Итак, обоснована полезность режима мультипрограммирования. Как уже говорилось, на первых ЭВМ этого режима работы не было. Сейчас будут сформулированы необходимые требования, которые предъявляются к аппаратуре компьютера, чтобы на нём было возможно реализовать мультипрограммный режим работы.

Сначала надо заметить, что требование параллельной работы процессора и периферийных устройств не является совершенно необходимым для режима разделения времени, который, как уже говорилось, является частным случаем мультипрограммного режима работы. Поэтому это требование не будет включаться в перечень *обязательных* свойств аппаратуры ЭВМ для обеспечения работы в мультипрограммном режиме. Надо, однако, отметить, что параллельная работа периферийных устройств и процессора сильно повышает производительность компьютера и реализована практически на всех современных ЭВМ.

13.1. Требования к аппаратуре для обеспечения возможности работы в мультипрограммном режиме

Прежде чем что-то требовать, подумай, как ты с этим справишься.

Галина Гончарова.
«Средневековая история.
Домашняя работа»

Итак, сформулируем необходимые требования к аппаратуре ЭВМ для обеспечения возможности мультипрограммной работы. Надо особо подчеркнуть, что это требования именно к аппаратуре ЭВМ, а не к её программному обеспечению.

Система прерываний. Система прерываний необходима как для режима разделения времени, так и для обеспечения параллельной работы процессора и периферийных устройств, так как она обеспечивает саму возможность реакции на события и автоматического переключения с одной программы на другую.

Механизм защиты памяти. Этот механизм обеспечивает безопасность одновременного нахождения в оперативной памяти нескольких независимых программ. Защита памяти гарантирует, что одна программа не сможет случайно или же предумышленно обратиться в память другой программы (по выполнению, по записи или даже по чтению данных). Очевидно, что без такого механизма мультипрограммный режим просто невозможен. Даже если не принимать во внимание «вредных» программистов, которые специально захотят испортить или незаконно прочитать данные других программ, всегда существует больша́я вероятность таких действий из-за семантических ошибок в программах даже у «добропорядочных» программистов (например, при выходе индекса за границу массива). Незаконное обращение к чужим ресурсам (в частности, к чужой оперативной памяти) «по научному» называется несанкционированным доступом.

Защита оперативной памяти на современных ЭВМ устроена весьма сложно, и часто связана с механизмом, так называемой (сегментно-страничной) виртуальной памяти, который в полном объёме изучается в курсе, посвященном операционным системам. Сейчас будет рассмотрена одна из простейших реализаций механизма защиты памяти, так эта защита была сделана на некоторых первых простых ЭВМ 3-го поколения, способных работать в мультипрограммном режиме.

В процессор добавляются два новых регистра защиты памяти, обозначим их $A_{нач}$ и $A_{кон}$. На каждый из этих регистров можно загрузить любой адрес оперативной памяти. Предположим теперь, что после загрузки некоторой программы в оперативную память она занимает сплошной участок памяти с адресами от 200000_{10} до 500000_{10} включительно. Тогда загрузчик, перед передачей управления на первую команду программы (у нас это часто была команда с меткой `Start`), присваивал регистрам защиты памяти соответственно значения

mov $A_{нач}$,200000 и **mov** $A_{кон}$,500000

Далее, в процессор добавлена способность, перед *каждым* обращением в оперативную память по физическому адресу $A_{физ}$ автоматически проверять условие

$$A_{нач} \le A_{физ} \le A_{кон}$$

Если условие истинно, т.е. программа обращается в свою область памяти, выполняется требуемое обращение к памяти по записи или чтению команд и данных. В противном случае доступ в оперативную память не производится, и процессор вырабатывает сигнал прерывания по событию «попытка нарушения защиты памяти».

Описанный механизм защиты памяти очень легко реализовать, однако он обладает существенным недостатком: каждая программа может занимать только один сплошной участок в оперативной памяти. В современных ЭВМ это ограничение несущественно, так как на них реализован уже упоминавшийся механизм виртуальной памяти, который позволяет выделять для каждой программы любые участки адресов памяти, независимо от того, заняты ли эти, как говорят, логические адреса другими программами или нет. С другой стороны, если реализован механизм виртуальной памяти, то на его базе легко сделать и другой, более совершенный механизм защиты памяти.

Аппарат привилегированных команд. Сейчас рассмотрим ещё одно необходимое свойство аппаратуры, без которого невозможно реализовать мультипрограммный режим работы ЭВМ. Это свойство иногда называется аппаратом привилегированных команд, а иногда – защищённым режимом работы процессора, и заключается оно в следующем: все команды, которые может выполнять процессор, разбиваются на две части. Команды из одной части называются обычными командами или командами пользователя, а команды из другой части – привилегированными или запрещёнными командами.

Далее, в процессоре на каком-то регистре располагается признак режима работы (CPU mode). Значение этого признака и определяет тот режим, в котором в данный момент работает процессор: обычный режим пользователя (user mode) или привилегированный режим. Привилегированный режим часто называют режимом супервизора (supervisor mode) или режимом ядра ОС (kernel mode). В архитектуре x86 текущий режим работы процессора обозначается CPL (**C**urrent **P**rivilege **L**evel), он находится в двух младших битах в сегментного регистра `CS`.

В привилегированном режиме процессору разрешается выполнять все команды языка машины, а в режиме пользователя – только обычные (не привилегированные) команды.[1] При попытке выполнить привилегированную команду в пользовательском режиме процессором вырабатывается сигнал прерывания, а сама команда, естественно, не выполняется. Из этого правила выполнения команд легко понять и другое название для привилегированных команд – запрещённые команды, так как их выполнение запрещено в режиме пользователя. Объясним теперь, почему без аппарата привилегированных команд невозможно реализовать мультипрограммный режим работы ЭВМ.

Легко понять, что, например, команды пересылки, которые для рассмотренного выше механизма защиты памяти заносят на регистры защиты $A_{нач}$ и $A_{кон}$ новые значения, должны быть привилегированными. Действительно, если бы это было не так, то любая программа могла бы занести на эти регистры адреса начала и конца всей оперативной памяти, после чего получила бы возможность записывать данные в любые области памяти. Ясно, что при этом и описанный выше механизм защиты памяти становится совершенно бесполезным.

Привилегированными должны быть и все команды, которые обращаются к внешним (периферийным) устройствам. Например, нельзя разрешать запись на диск в режиме пользователя, так как диск – это тоже общая память для всех программ, только внешняя, и одна программа может испортить на диске данные (файлы), принадлежащие другим программам. То же самое относится и к печатающему устройству: если разрешить всем программам бесконтрольно выводить свои данные на печать, то, конечно, разобраться в том, что же получится на бумаге, будет чаще всего невозможно. Поэтому, если работающие в мультипрограммном режиме программы Васи и Пети производят вывод на общий принтер, то на самом деле данные, которые печатает каждая программа пользователя, не выводятся сразу на печать, а записываются в специальный файл, который обычно будет выводиться на печать только после полного завершения этой программы. Таким образом, выводимые на печать данные Васи и Пети не перепутаются.

Итак, в мультипрограммном режиме программе пользователя запрещается выполнять многие «опасные» команды, в частности команды, работающие с внешними устройствами (дисками, принтерами, линиями связи и т.д.).[2] Как же тогда быть, если программе необходимо, например, считать данные из своего файла на диске в оперативную память? Выход один – программа пользователя должна обратиться к служебным подпрограммам, с просьбой выполнить для неё ту работу, которую сама программа пользователя сделать не в состоянии. Эти служебные подпрограммы, естественно, должны работать в привилегированном режиме. Перед выполнением запроса из программы пользователя, такая служебная подпрограмма проверяет, имеет ли эта программа пользователя право на запрашиваемое действие, например, что эта программа имеет необходимые полномочия на чтение из указанного файла.

Переключение из привилегированного режима в режим пользователя обычно производится с помощью некоторой команды или изменением определённого флага в управляющем регистре ЭВМ. На нашем процессоре изменяется значение сегментного регистра `CS`.

Значительно сложнее обстоит дело с такой опасной операцией, как переключение процессора из обычного режима работы в привилегированный режим. Это переключение невозможно выполнить какой-либо машинной командой (чтобы это понять, достаточно задаться вопросом, должна ли сама эта команда переключения быть привилегированной, или нет). Обычно переключение в привилегированный режим производится автоматически при обработке процессором сигнала прерывания, в этом случае процедура-обработчик прерывания начинает свою работу уже в привилегированном ре-

---

[1] Как уже говорилось, привилегия на выполнение команд ввода/вывода (**`sti`**, **`cli`**, **`in`**, **`out`** и некоторых других) определяется не CPL, а полем IOPL (`I/O` **P**rivilege **L**evel – уровень привилегий ввода-вывода) в 12-ом и 13-ом битах регистра флагов. Таким образом, в нашей архитектуре существуют два класса привилегированных команд. Второй набор таких команд «менее привилегированные», иногда можно разрешить выполнять их и «обычным» программам.

[2] Как уже говорилось, некоторые периферийные устройства могут отдаваться в *монопольное использование* некоторой задаче, все номера портов этого устройства помечаются нулями в битовой карте разрешения ввода-вывода (I/O Permission Bit Map) этой задачи, о чём будет рассказано в другой главе.

жиме. Иногда переключение в привилегированный режим производится процессором при дальнем вызове системных процедур, которые имеют полномочия для работы в привилегированном режиме.[1]

Таймер. Встроенные в компьютер электронные часы (таймер) появились ещё до возникновения мультипрограммного режима работы. Тем не менее, легко понять, что без таймера мультипрограммный режим тоже невозможен. Действительно, это единственное внешнее устройство, которое гарантированно и периодически посылает процессору сигналы прерываний. Без таких сигналов некоторые программы могли бы войти в выполнение бесконечного цикла (зациклиться), и ничто не могло бы вывести компьютер из этого состояния.

Обычно при счёте в мультипрограммном режиме программа пользователя может сообщить операционной системе свое максимальное время счёта. Это не физическое время, а сумма всех квантов времени центрального процессора, выделяемых для этой задачи. Можно сказать, что программа заводит для себя «будильник», на котором выставляется отведённое ей время работы. По истечению этого максимального времени счёта программа пользователя получит соответствующий сигнал и может быть завершена.

Итак, были рассмотрены аппаратные средства, необходимые для обеспечения мультипрограммного режима работы ЭВМ. Остальные аппаратные возможности ЭВМ, которые часто называются при ответе на этот вопрос (такие, как большая оперативная память, высокое быстродействие процессора, большая емкость дисков и другие) являются, конечно, желательными, но не являются необходимыми.

Разумеется, кроме перечисленных аппаратных средств, для обеспечения мультипрограммной работы совершенно необходимы и специальные *программные средства*, прежде всего операционная система, поддерживающая режим мультипрограммной работы. Такая операционная система является примерно на порядок более сложной, чем её предшественницы – операционные системы, не поддерживающие мультипрограммный режим работы. Все это, однако, отдельная тема, а нам надо продолжить изучение архитектуры ЭВМ.

Вопросы и упражнения

Всякий человек имеет естественное желание знать.

Аристотель, IV век до н.э.

1. Что такое пакетный режим работы ЭВМ и для чего он нужен ?
2. Дайте определение мультипрограммного режима работы ЭВМ. Когда этот режим необходим ?
3. Что такое режим разделения времени и для чего он нужен ?
4. Почему полезна параллельная работа процессора и устройств ввода/вывода ?
5. Что такое аппарат привилегированных команд и почему он необходим для мультипрограммного режима работы ЭВМ ?
6. Какие команды машины необходимо делать привилегированными ?
7. Почему в языке машины не может существовать команды для переключения процессора из обычного режима работы в привилегированный режим ?
8. Что такое таймер и почему он необходим в мультипрограммном режиме работы ?
9. Объясните, почему на рис. 13.2 между счётом любых двух программ пользователей процессор обязательно на некоторое время переключается на программу операционной системы ?

[1] Как уже упоминалось, процессор производит это переключение при вызове процедуры-обработчика через шлюз задачи (Task Gate), или вызове системной функции через шлюз вызова (Call Gate).