Операционные системы

Основные понятия

Операционная система является посредником между приложениями, утилитами и пользователями, с одной стороны, и аппаратным обеспечением – с другой. Операционная система обслуживает пользователей, обращаясь при этом к ресурсам аппаратного обеспечения, в состав которого входит один или несколько процессоров. Она управляет вторичной памятью и устройствами ввода – вывода. Поэтому прежде чем приступить к исследованию операционных систем, важно иметь представление о компьютерных системах, на которых работают операционные системы.

На макроуровне компьютер состоит из процессора, памяти и устройств ввода – вывода. Чтобы компьютер мог выполнять свое основное назначение, состоящее в выполнении программ. Различные компоненты должны иметь возможность взаимодействовать между собой. Структурно можно выделить четыре компонента компьютера:

Процессор. *Осуществляет контроль за действиями компьютера и выполняет функцию обработки данных. Если в системе имеется только один процессор, то, чаще всего, он называется центральным процессором.*

Основная память. *Предназначена для хранения данных и программы. Временная или оперативная(первичная).*

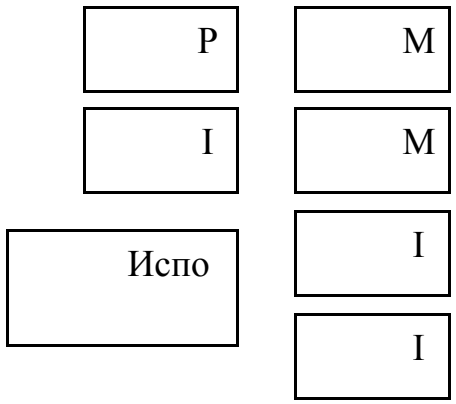
Устройства ввода – вывода. *Служат для передачи данных между компьютером и внешним окружением, состоящим из различных периферийных устройств.*

Системная шина. *Определенные структуры и механизмы, которые обеспечивают взаимодействие между процессором, основной памятью и устройствами ввода – вывода.*

Все вышеупомянутые компоненты схематично можно изобразить так:

CPU

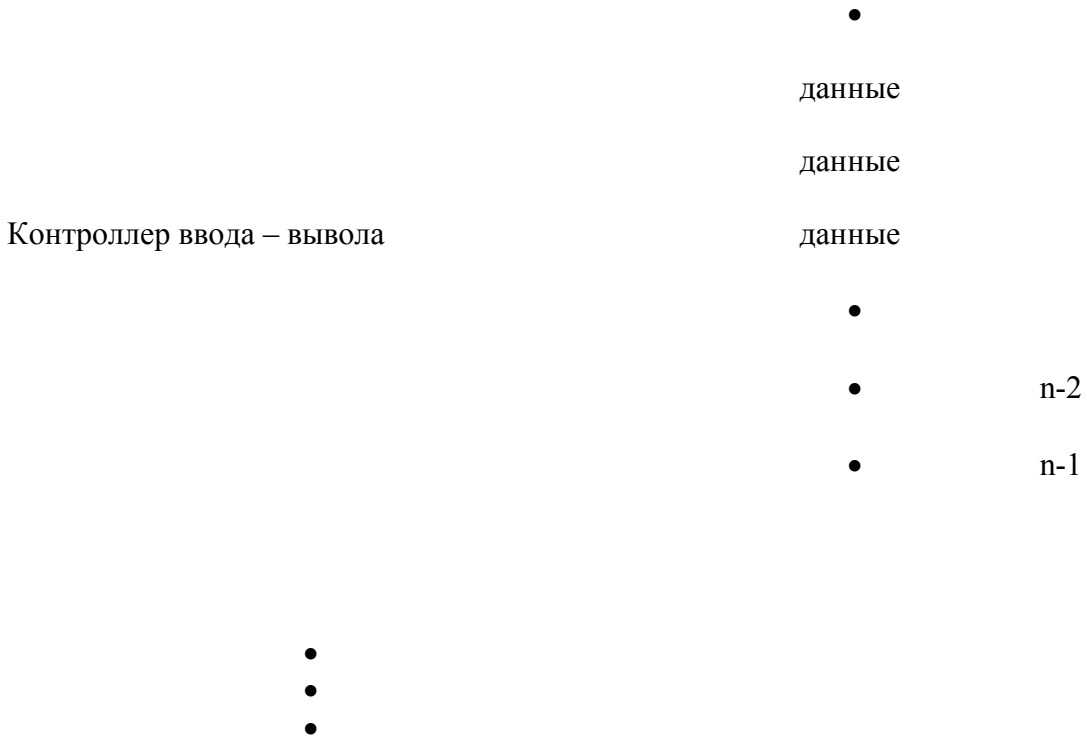
Основная память



- 0
- 1
- 2

команда

- команда
- команда
-
-



Одной из основных функций процессора является обмен данными с памятью. Для этого он обычно использует два внутренних (по отношению к процессору0 регистра: регистр адреса памяти (MAR – memory address register), куда заносится адрес ячейки памяти, в которой будет производиться операция чтения – записи, и регистр буфера памяти (MBR - memory buffer register), куда заносятся данные, предназначенные для записи в память, или те, которые были прочитаны из нее. Аналогично, номер устройства ввода – вывода задается в регистре адреса ввода – вывода (I/O address register –I/O AR). Регистр буфера ввода – вывода (I/O buffer register –I/O BR) служит для обмена данными между устройствами ввода – вывода и процессором. Модуль памяти состоит из множества пронумерованных ячеек. В каждую ячейку может быть записано двоичное число, которое интерпретируется либо как команда, либо как данные. Модуль ввода – вывода служит для передачи данных от внешних устройств как в процессор и память, так и в обратном направлении. Для временного хранения данных в нем есть свои внутренние буферы.

В процессоре имеется набор регистров, представляющих собой область памяти быстрого доступа, но намного меньшей емкости, чем основная память. Регистры процессора выполняют две функции.

- Регистры, доступные пользователю. Эти регистры позволяют программисту сократить число обращений к основной памяти, оптимизируя использование регистров с помощью машинного языка или ассемблера. В состав языков высокого уровня входят оптимизирующие компиляторы, построенные на алгоритмах, которые, в частности, позволяют определить, какие переменные следует заносить в регистры, а какие — в основную память. Некоторые языки высокого уровня, такие, как C, предоставляют программисту возможность предложить компилятору хранить те или иные данные в регистрах.
- Регистры управления и регистры состояния. Используются в процессоре для контроля над выполняемыми операциями; с их помощью привилегированные программы операционной системы могут контролировать ход выполнения других программ.

Для разделения регистров на эти две категории не существует определенного признака. Например, на некоторых машинах оператор имеет возможность следить за состоянием программного счетчика, а на других — нет. Однако такое разделение удобно при дальнейшем рассмотрении.

Регистры, доступные пользователю

К доступным регистрам пользователь может обращаться с помощью команд машинного языка. К этим регистрам, как правило, имеют доступ все программы — как приложения, так и системные. Обычно среди доступных регистров есть регистры данных, адресные регистры и регистры кода условия.

Регистры данных.

Программист может применять их в различных целях. В ряде случаев они имеют общее назначение и могут использоваться любой машинной командой для операций с данными. Однако зачастую при этом накладываются определенные

ограничения. Например, некоторые регистры предназначены для операций над числами с плавающей точкой, в то время как остальные — для хранения целых чисел.

Адресные регистры.

В них заносятся адреса команд и данных в основной памяти; в этих регистрах может быть записана только часть адреса, использующаяся при вычислении полного или эффективного адреса. Рассмотрим следующие примеры.

- Индексный регистр. Используется в обычном режиме адресации, когда адрес получается в результате сложения содержимого индексного и базового регистра.

- Сегментный регистр. При сегментной адресации память разделяется на блоки (сегменты), состоящие из различного количества машинных слов. Адрес ячейки памяти складывается из адреса сегмента и смещения относительно начала сегмента. При этом режиме адресации базовый адрес сегмента (его начало) хранится в одном из регистров. Таких регистров может быть несколько; например, один — для операционной системы (т.е. использующийся при выполнении процессором кода операционной системы), другие — для исполняющихся в данный момент приложений.

- Регистр стека. Стек размещается в основной памяти в виде последовательности ячеек. Он похож на стопку бумаг, в которой листы с данными можно брать и класть только сверху При стековой адресации выделяется специальный регистр, в котором размещен указатель на вершину стека. Этот режим адресации позволяет использовать некоторые команды, в которых отсутствует поле адреса, например push и pop.

В некоторых машинах вызов процедуры или подпрограммы приводит к автоматическому сохранению содержимого всех доступных пользователю регистров, чтобы по возвращении их можно было восстановить. Процедура сохранения и восстановления, являющаяся составной частью команды вызова и возврата, выполняется процессором. Такой подход позволяет процедурам

использовать регистры независимо друг от друга. В других машинах операция сохранения содержимого регистров, выполняемая при вызове процедуры, является обязанностью программиста, который должен включить в программу необходимые команды. Таким образом, в зависимости от типа машины функция сохранения и восстановления может выполняться либо аппаратно, либо программно.

Управляющие регистры и регистры состояния

Для контроля над работой процессора используются различные регистры. В большинстве машин эти регистры в основном не доступны пользователю. Некоторые из них могут быть доступны для машинных команд, исполняемых в так называемом режиме управления или режиме операционной системы.

Конечно, у разных типов машин организация регистров отличается; для их классификации также используется различная терминология. Кроме ранее упомянутых регистров, MAR, MBR, I/OAR и I/OBR, важными для выполнения команд являются следующие:

- **Программный счетчик** (program counter — PC). Содержит адрес команды, которая должна быть выбрана из памяти.
- **Регистр команд** (instruction register — IR). Содержит последнюю выбранную из памяти команду.

В состав всех процессоров входит также регистр (или набор регистров), известный под названием регистра слова состояния программы (program status word — PSW). В нем, как правило, содержатся коды условий и другая информация о состоянии, например, бит разрешения/запрещения прерываний или бит режима системный/пользовательский.

Коды условий (известные также как флаги) — это последовательность битов, устанавливаемых или сбрасываемых процессором в зависимости от результата выполненных операций.

Например, в результате выполнения арифметического действия может получиться положительное число, отрицательное, ноль, или может произойти

переполнение. В дополнение к сохранению полученного значения в памяти или регистре в результате арифметических операций устанавливаются также соответствующие коды условий. Впоследствии они могут быть проверены условной операцией ветвления. Биты кодов условий группируются в один или несколько регистров (обычно они составляют часть регистра управления). Вообще говоря, есть машинные команды, позволяющие прочитать содержимое этих битов с помощью явных обращений к регистрам; однако изменять их содержимое явным образом нельзя, поскольку эти биты предназначены для отображения результатов выполнения команд.

В машинах, в которых используются различные виды прерываний, может быть предусмотрено несколько регистров прерываний с указателями на каждую программу обработки прерываний. Если при реализации некоторых функций (например, вызова процедуры) используется стек, процессор должен иметь регистр — указатель стека. Для аппаратного обеспечения управления памятью нужны свои регистры. И, наконец, регистры часто используются при управлении операциями ввода-вывода.

На устройство и организацию управляющих регистров и регистров состояния влияют несколько факторов. Одним из них является поддержка операционной системы. Различные виды управляющей информации используются операционной системой по-разному. Если разработчик процессора имеет ясное представление об операционной системе, которая будет работать с этим процессором, он сможет так спланировать организацию регистров, чтобы обеспечить аппаратную поддержку ряда возможностей, таких, как защита памяти или переключение пользовательских программ.

Еще одним ключевым конструкторским решением является распределение управляющей информации между регистрами и памятью. Общепринятым подходом является выделение для нее нескольких первых сотен или тысяч слов памяти. Конструктор должен решить, какая часть информации будет находиться в

более дорогих, но более быстрых регистрах, а какая часть — в более дешевой, но медленной основной памяти.

Программа, которую выполняет процессор, состоит из набора хранящихся в памяти команд. В простейшем виде обработка команд проходит в две стадии: процессор считывает *(выбирает)* из памяти, а затем запускает очередную команду. Исполнение программы сводится к повторению процесса выборки команды и ее исполнения. Для выполнения одной команды может потребоваться несколько операций; их число определяется природой самой команды.

Набор действий, требующихся для реализации одной команды, называется ее *циклом.* На рис. 1.2 показан процесс обработки команд процессором в такой упрощенной схеме, включающей два этапа. Эти этапы называются *циклом выборки* и *циклом исполнения.* Прекращение работы программы происходит при выключении машины, в случае возникновения какой-либо фатальной (неисправимой) ошибки, или если в программе имеется команда останова.

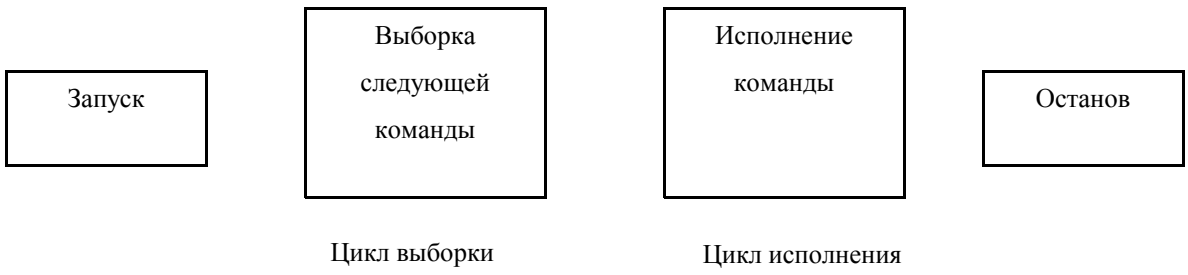


Рис. 1.2. Базовый цикл исполнения программы

Выборка и исполнение команды

В начале каждого цикла процессор выбирает из памяти команду. Обычно адрес ячейки, из которой нужно извлечь очередную команду, хранится в программном счетчике (PC). Если не указано иное, после извлечения каждой команды процессор увеличивает значение программного счетчика на единицу. Таким образом, команды выполняются в порядке возрастания номеров ячеек памяти, в которых они хранятся. Рассмотрим, например, упрощенный компьютер, в котором каждая команда занимает одно 16-битовое слово памяти. Предположим, что значение программного счетчика установлено равным 300. Это значит, что следующая команда, которую должен извлечь процессор, находится в 300-й ячейке. При успешном завершении цикла команды процессор перейдет к извлечению команд из ячеек 301, 302, 303 и т.д. Однако, эта последовательность может быть изменена.

Извлеченные команды загружаются в регистр команд (IR). Команда состоит из последовательности битов, указывающих процессору, какие именно действия он должен выполнить. Процессор интерпретирует команду и выполняет требуемые действия. Все действия можно разбить на четыре категории:

- **Процессор — память**. Данные передаются из процессора в память или обратно.
- **Процессор — устройства ввода-вывода**. Данные из процессора поступают на периферийное устройство через устройство ввода-вывода. Возможен и обратный процесс.
- **Обработка данных**. Процессор выполняет с данными различные арифметические или логические операции.
- **Управление**. Команда может задавать изменение последовательности выполнения команд. Например, если процессор извлекает из ячейки 149 команду, которая указывает, что следующей по очереди должна быть исполнена команда из ячейки 182, то процессор устанавливает значение

программного счетчика равным 182. Таким образом, в следующем цикле выборки команда извлекается не из ячейки 150, а из ячейки 182.

Для выполнения команды может потребоваться последовательность, состоящая из комбинации вышеперечисленных действий.

Функции ввода-вывода

До сих пор мы рассматривали операции компьютера, управляемые процессором, основное внимание, обращая на взаимодействие процессора и памяти.

Процессор может не только читать данные из памяти и записывать их туда, обращаясь по адресу к определенной ячейке, но также читать и записывать данные в устройство ввода-вывода. Таким образом, устройство ввода-вывода (например, контроллер диска) обменивается данными с процессором. При этом процессор должен идентифицировать устройство, которое будет управляться определенным устройством ввода-вывода. Из команд ввода-вывода можно сформировать такие же последовательности, как последовательности команд обращения к памяти.

Иногда желательно, чтобы обмен данными с памятью выполнялся непосредственно устройством ввода-вывода, а процессор в это время выполнял другие задания. В этом случае процессор передает устройству ввода-вывода полномочия для чтения из памяти и записи в память, что позволяет освободить процессор. Во время такой передачи данных устройство ввода-вывода читает или записывает команды в память, принимая на себя ответственность за этот обмен. Такой режим, известен под названием прямого доступа к памяти (direct memory access — DMA).

Прерывания

Во всех компьютерах предусмотрен механизм, с помощью которого различные устройства (ввода-вывода, памяти) могут прервать нормальную работу процессора. Основные общепринятые классы прерываний перечислены в табл. 1.1.

Таблица 1.1. Классы прерываний

Программное прерывание	Генерируется в некоторых ситуациях, возникающих в результате выполнения команд. Такими ситуациями могут быть арифметическое переполнение, деление на ноль, попытка выполнить некорректную команду и ссылка на область памяти, доступ к которой пользователю запрещен.
Прерывание по таймеру	Генерируется таймером процессора. Это прерывание позволяет операционной системе выполнять некоторые свои функции периодически, через заданные промежутки времени.
Прерывание ввода-вывода	Генерируется контроллером ввода-вывода. Сигнализирует о нормальном завершении операции или о наличии ошибок.
Аппаратное прерывание	Генерируется при возникновении таких аварийных ситуаций, как, например, падение напряжения в сети или ошибка контроля четности памяти.

Прерывания в основном предназначены для повышения эффективности работы. Например, большинство устройств ввода-вывода работают намного медленнее, чем процессор. Предположим, что процессор передает данные на принтер по схеме, показанной рис. 1.2. После каждой операции процессор вынужден делать паузу и ждать, пока принтер не примет данные. Длительность этой паузы может быть в сотни и даже тысячи раз больше длительности цикла команды, в которой участвуют обращения к памяти. Ясно, что подобное использование процессора является неэффективным.

Программа пользователя содержит ряд вызовов процедуры записи WRITE, в промежутках между которыми расположены другие команды. В отрезках между процедурами записи WRITE находятся последовательности команд кода, в которых не используется ввод-вывод. При вызове процедуры WRITE управление передается

системной утилите ввода-вывода, которая выполняет соответствующие операции. Программа ввода-вывода состоит из трех частей.

- Последовательность команд, которые служат для подготовки к собственно операциям ввода-вывода. В эту последовательность могут входить копирование выводимых данных в специальный буфер и подготовка набора параметров, необходимых для управления устройством.
- Собственно команды ввода-вывода. Если программа не использует прерываний, ей следует ждать, пока устройство ввода-вывода не выполнит требуемые операции (или периодически проверять его состояние путем опроса). При этом программе не остается ничего другого, как просто ждать, постоянно проверяя, завершилась ли операция ввода-вывода.
- Последовательность команд, которые служат для завершения операции. Эта последовательность может содержать в себе установку флагов, свидетельствующих об успешном или неудачном завершении операции.

Из-за того что для выполнения операции ввода-вывода может потребоваться сравнительно длительный промежуток времени, программа замедляет работу, ожидая завершения операции. Таким образом, там, где встречается вызов WRITE, производительность программы существенно уменьшается.

Прерывания и цикл команды

Благодаря прерываниям во время выполнения операций ввода-вывода процессор может быть занят обработкой других команд. Рассмотрим ход процесса. Вызвав процедуру WRITE, программа обращается к системе. При этом активизируется программа ввода-вывода, которая состоит из подготовительного кода и собственно команд ввода-вывода. После исполнения этих команд управление передается программе пользователя. Тем временем внешнее устройство занято приемом данных из памяти компьютера и их обработкой (например, если этим устройством является принтер, то под обработкой подразумевается

распечатка). Ввод-вывод происходит одновременно с выполнением команд программы пользователя.

В тот момент, когда внешнее устройство освобождается и готово для дальнейшей работы, т.е. оно готово принять от процессора новую порцию данных, контроллер ввода-вывода этого устройства посылает процессору сигнал *запроса прерывания.* В ответ процессор приостанавливает выполнение текущей программы, переключаясь на работу с программой, обслуживающей данное устройство ввода-вывода (эту программу называют обработчиком прерываний). Обслужив внешнее устройство, процессор снова возобновляет прерванную работу. С точки зрения программы пользователя, прерывания — это не что иное, как нарушение обычной последовательности исполнения. После завершения обработки прерывания работа возобновляется. Таким образом, программа пользователя не должна включать в себя какой-нибудь специальный код, чтобы приспосабливаться к прерываниям. За приостановку программы пользователя и возобновление ее работы с того самого места, в котором она была прервана, отвечают процессор и операционная система.

Чтобы согласовать прерывание с программой, в цикл команды добавляется цикл прерывания. В цикле прерывания процессор проверяет наличие сигналов прерываний, свидетельствующих о происшедших прерываниях. При поступлении прерывания процессор приостанавливает работу с текущей программой и выполняет *обработчик прерываний.* Обработчики прерываний обычно входят в состав операционной системы. Как правило, эти программы определяют природу прерывания и выполняют необходимые действия. Например, обработчик должен определить, какой из контроллеров ввода-вывода сгенерировал прерывание; кроме того, он может передавать управление программе, которая должна вывести данные на устройство ввода-вывода. Когда обработчик прерываний завершает свою работу, процессор возобновляет выполнение программы пользователя с того места, где она была прервана.

Ясно, что этот процесс, включает в себя некоторые непроизводительные затраты. Для определения природы прерывания и принятия решения о последующих действиях обработчик прерываний должен выполнить дополнительные команды. Тем не менее, ввиду того, что для ожидания завершения операций ввода-вывода потребовался бы сравнительно большой отрезок времени, с помощью прерываний процессор можно использовать намного эффективнее.

Обработка прерываний

Прерывание вызывает ряд событий, которые происходят как в аппаратном, так и в программном обеспечении. На рис. 1.3 показана типичная последовательность этих событий. После завершения работы устройства ввода-вывода происходит следующее:

- Устройство посылает процессору сигнал прерывания.
- Перед тем как ответить на прерывание, процессор должен завершить исполнение текущей команды (см. рис. 1.7).
- Процессор производит проверку наличия прерывания, обнаруживает его и посылает устройству, приславшему это прерывание, уведомляющий сигнал об успешном приеме. Этот сигнал позволяет устройству снять свой сигнал прерывания.
- Теперь процессору нужно подготовиться к передаче управления обработчику прерываний. Сначала необходимо сохранить всю важную информацию, чтобы в дальнейшем можно было вернуться к тому месту текущей программы, где она была приостановлена. Минимальная требуемая информация — это слово состояния программы и адрес очередной выполняемой команды, который находится в программном счетчике. Эти данные заносятся в системный управляющий стек.

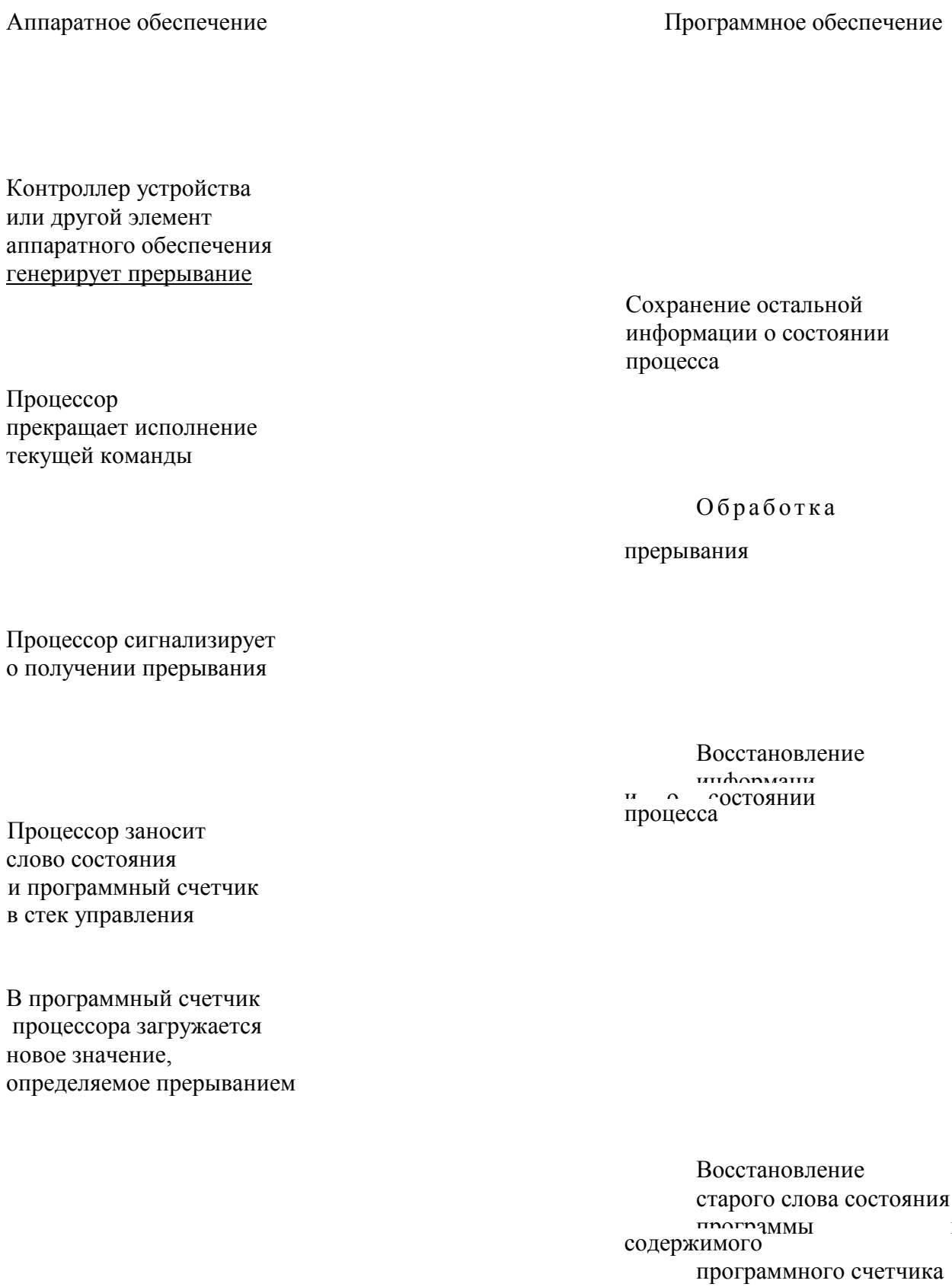


Рис. 1.3. Обработка простого прерывания

- Далее в программный счетчик процессора загружается адрес входа программы обработки прерываний, которая отвечает за обработку данного

прерывания. В зависимости от архитектуры компьютера и устройства операционной системы может существовать как одна программа для обработки всех прерываний, так может быть и своя программа обработки для каждого устройства и каждого типа прерываний. Если для обработки прерываний имеется несколько программ, то процессор должен определить, к какой из них следует обратиться. Эта информация может содержаться в первоначальном сигнале прерывания; в противном случае для получения необходимой информации процессор должен по очереди опросить все устройства, чтобы определить, какое из них отправило прерывание. Как только в программный счетчик загружается новое значение, процессор переходит к следующему циклу команды, приступая к ее извлечению из памяти. Так как команда извлекается из ячейки, номер которой задается содержимым программного счетчика, управление переходит к программе обработки прерываний. Исполнение этой программы влечет за собой следующие операции.

- Содержимое программного счетчика и слово состояния прерываемой программы уже хранятся в системном стеке. Однако это еще не вся информация, имеющая отношение к состоянию исполняемой программы. Например, нужно сохранить содержимое регистров процессора, так как эти регистры могут понадобиться обработчику прерываний. Поэтому необходимо сохранить всю информацию о состоянии программы. Обычно обработчик прерываний начинает свою работу с записи в стек содержимого всех регистров. Указатель стека при этом обновляется, указывая на новую вершину стека. Обновляется и программный счетчик, указывая на начало программы обработки прерывания.

- Теперь обработчик прерываний может начать свою работу. В процесс обработки прерывания входит проверка информации состояния, имеющая отношение к операциям ввода-вывода или другим событиям, вызвавшим

прерывание. Сюда может также входить пересылка устройствам ввода-вывода дополнительных инструкций или уведомляющих сообщений.

- После завершения обработки прерываний из стека извлекаются сохраненные ранее значения, которые вновь заносятся в регистры, возобновляя, таким образом, то состояние, в котором они пребывали до прерывания.

- Последний этап — восстановление из стека слова состояния программы и содержимого программного счётчика. В результате следующей будет выполняться команда прерванной программы.

- Из-за того, что прерывание не является подпрограммой, вызываемой из программы, для полного восстановления важно сохранить всю информацию состояния прерываемой программы. Однако прерывание может произойти в любой момент и в любом месте программы пользователя. Это событие непредсказуемо.

Множественные прерывания

До сих пор нами рассматривался случай возникновения одного прерывания. Представим себе ситуацию, когда может произойти несколько прерываний. Например, программа получает данные по коммуникационной линии и сразу же распечатывает результат. Принтер будет генерировать прерывание при каждом завершении операции печати, а контроллер коммуникационной линии — при каждом поступлении новой порции данных. Эта порция может состоять из одного символа или из целого блока, в зависимости от установленного порядка обслуживания. В любом случае возможна ситуация, когда коммуникационное прерывание произойдет во время обработки прерывания принтера.

В такой ситуации возможны два подхода. Первый — это запретить новые прерывания до тех пор, пока обрабатывается предыдущее. *Запрет прерываний* означает, что процессор может и должен игнорировать любой новый сигнал прерывания. Если в это время происходит прерывание, оно обычно остается в состоянии ожидания, и до него дойдет очередь, когда процессору вновь будет можно обрабатывать прерывания. Таким образом, если во время работы программы

пользователя происходит прерывание, на другие прерывания тут же накладывается запрет. После завершения работы программы обработки прерывания запрет снимается, и перед возвратом к исполнению прерванной программы процессор проверяет наличие других прерываний. Это удачный и простой подход, при котором прерывания обрабатываются в строго последовательном порядке. Однако недостатком такого подхода является то, что он не учитывает приоритет прерываний и те ситуации, в которых время является критическим параметром. Например, когда по коммуникационной линии приходит какая-то информация, может понадобиться быстро ее принять, чтобы освободить место для других входных данных. Если не обработать первый пакет входных данных перед получением второго пакета, данные могут потеряться вследствие загруженности и переполнения буфера устройства ввода-вывода.

При втором подходе учитывается приоритет прерывания, что позволяет приостановить обработку прерывания с более низким приоритетом в пользу прерывания с более высоким приоритетом.

Многозадачность

Бывает, что для эффективного использования процессора одних прерываний недостаточно. Например, если время, которое требуется для выполнения операций ввода-вывода, намного больше, чем время работы фрагмента пользовательского кода, который находится между вызовами ввода-вывода, то большую часть времени процессор будет простаивать. Чтобы решить эту проблему, нужно позволить нескольким программам пользователя быть активными в одно и то же время.

Предположим, процессору нужно выполнить две программы. Одна из них, читающая данные из памяти и выводящая их на внешнее устройство, достаточно проста; другая программа — некое приложение, выполняющее сложные вычисления. Процессор может начать работу с программой вывода, сгенерировать команду записи на внешнее устройство, а затем перейти к вычислениям, требующимся для выполнения другого приложения. Если процессор работает с

несколькими программами, то последовательность их выполнения зависит от относительного приоритета этих программ, а также от того, ожидают ли они завершения ввода-вывода. Если программа прервана с передачей управления обработчику прерываний, то после завершения обработки управление не обязательно сразу же передается программе пользователя, которая выполнялась до этого. Управление может быть передано какой-либо другой программе, которая находится в состоянии ожидания и обладает более высоким приоритетом. Прерванная программа пользователя возобновит работу, когда она будет обладать наиболее высоким приоритетом среди оставшихся программ. Эта концепция обработки нескольких программ, воплощенная на практике, называется многозадачностью.