1.3. Обзор аппаратного обеспечения компьютера **43**

начала нас вполне устроит и эта модель. В следующих разделах будет дан краткий обзор отдельных компонентов и рассмотрены некоторые аспекты аппаратного обеспечения, представляющие интерес для разработчиков операционных систем. Наверное, излишне упоминать о том, что это будет очень краткое изложение. Компьютерному оборудо- ванию и его организации посвящено множество книг. Можно порекомендовать две довольно известные книги (Tanenbaum, 2012; Patterson and Hennessy, 2013).

**1.3.1. Процессоры**

Центральный процессор — это «мозг» компьютера. Он выбирает команды из памяти и выполняет их. Обычный цикл работы центрального процессора выглядит так: выбор- ка из памяти первой команды, ее декодирование для определения ее типа и операндов, выполнение этой команды, а затем выборка, декодирование и выполнение последую- щих команд. Этот цикл повторяется до тех пор, пока не закончится программа. Таким образом программы выполняются.

Для каждого типа центрального процессора существует определенный набор команд, которые он может выполнять. Поэтому x86 не может выполнять программы, написан- ные для ARM-процессоров, а те, в свою очередь, не в состоянии выполнять программы, написанные для x86. Поскольку доступ к памяти для получения команды или данных занимает намного больше времени, чем выполнение команды, у всех центральных про- цессоров есть несколько собственных регистров для хранения основных переменных и промежуточных результатов. Соответственно набор команд содержит, как правило, команды на загрузку слова из памяти в регистр и на запоминание слова из регистра в память. Другие команды объединяют два операнда из регистров, памяти или обоих этих мест для получения результата — например, складывают два слова и сохраняют результат в регистре или в памяти.

В дополнение к регистрам общего назначения, которые обычно применяются для хранения переменных и промежуточных результатов, у многих процессоров есть ряд специальных регистров, доступных программисту. Один из этих регистров, называ- емый **счетчиком команд**, содержит адрес ячейки памяти со следующей выбираемой командой. После выборки этой команды счетчик команд обновляется, переставляя указатель на следующую команду.

Другой специальный регистр, называемый **указателем стека**, ссылается на вершину те- кущего стека в памяти. Стек содержит по одному фрейму (области данных) для каждой процедуры, в которую уже вошла, но из которой еще не вышла программа. В стековом фрейме процедуры хранятся ее входные параметры, а также локальные и временные переменные, не содержащиеся в регистрах.

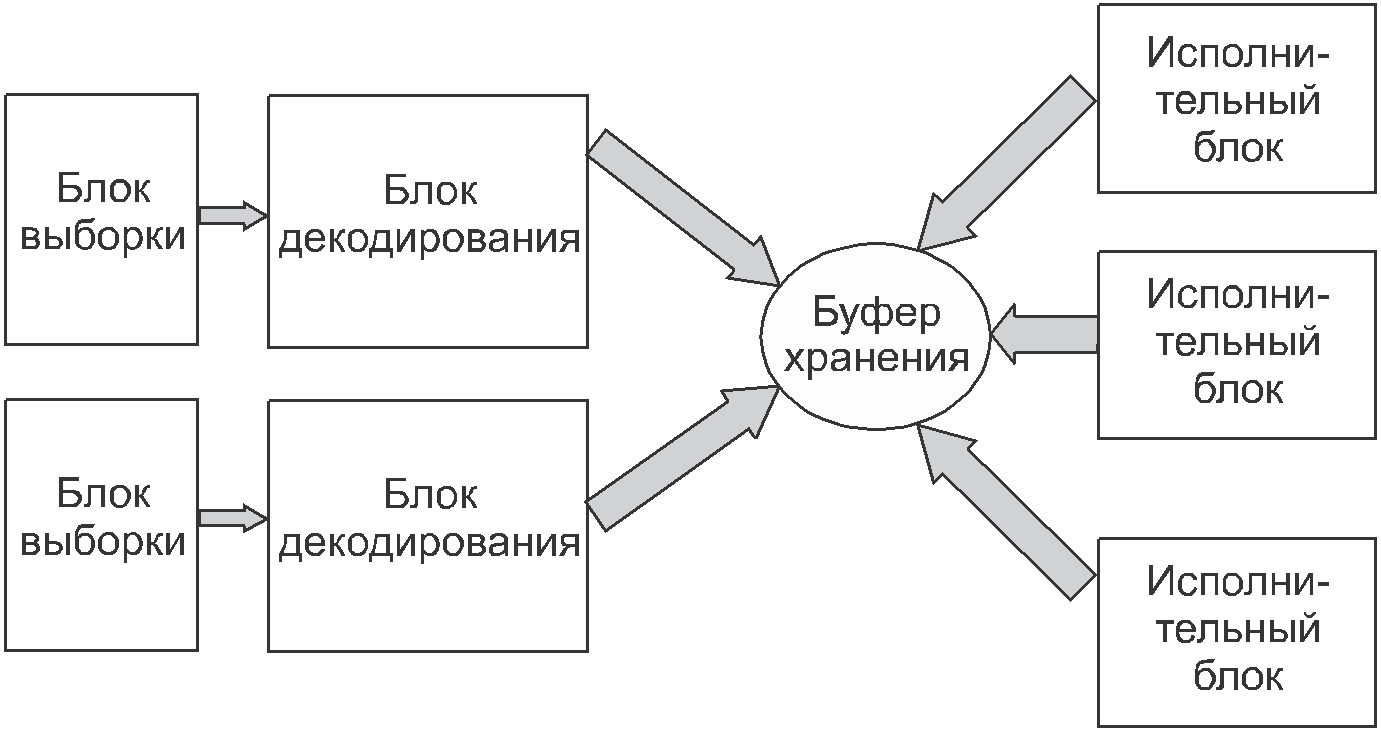
Еще один регистр содержит **слово состояния программы** — PSW (Program Status Word). В этом регистре содержатся биты кода условия, устанавливаемые инструкциями сравнения, а также биты управления приоритетом центрального процессора, режимом (пользовательским или ядра) и другие служебные биты. Обычно пользовательские программы могут считывать весь регистр PSW целиком, но записывать — только в некоторые из его полей. Регистр PSW играет важную роль в системных вызовах и операциях ввода-вывода.

Операционная система должна все знать о состоянии всех регистров. При временном мультиплексировании центрального процессора операционная система может часто

**44** Глава 1. Введение

останавливать работающую программу, чтобы запустить или возобновить работу дру- гой программы. При каждой остановке работающей программы операционная система должна сохранять состояние всех регистров, чтобы восстановить его при последующем возобновлении работы этой программы.

Для повышения производительности процессоров их разработчики давно отказались от простой модели извлечения, декодирования и выполнения одной команды за один цикл. Многие современные процессоры способны одновременно выполнять более одной команды. Например, у процессора могут быть отдельные блоки для выборки, декодирования и выполнения команд, тогда во время выполнения команды *n* он смо- жет декодировать команду *n* + 1 и осуществлять выборку команды *n* + 2. Подобная организация работы называется **конвейером**. На рис. 1.7, *а* показан конвейер с тремя стадиями обработки. Обычно используются более длинные конвейеры. В большинстве конструкций конвейеров, как только команда выбрана и помещена в конвейер, она должна быть выполнена, даже если предыдущая выбранная команда была условным ветвлением. Для разработчиков компиляторов и операционных систем конвейеры — это сплошная головная боль, обнажающая перед ними все сложности исходной маши- ны и заставляющая справляться с возникающими проблемами.



**Рис. 1.7.** Процессор: *а* — с конвейером с тремя стадиями; *б* — суперскалярный

Более совершенной конструкцией по сравнению с конвейерной обладает **суперскаляр- ный** процессор, показанный на рис. 1.7, *б*. Он имеет несколько исполнительных блоков, например: один — для целочисленной арифметики, другой — для арифметики чисел с плавающей точкой, третий — для логических операций. Одновременно выбираются две и более команды, которые декодируются и помещаются в буфер хранения, в ко- тором ожидают возможности своего выполнения. Как только исполнительный блок становится доступен, он обращается к буферу хранения за командой, которую может

1.3. Обзор аппаратного обеспечения компьютера **45**

выполнить, и если такая команда имеется, извлекает ее из буфера, а затем выполняет. В результате команды программы часто выполняются не в порядке их следования. При этом обеспечение совпадения конечного результата с тем, который получился бы при последовательном выполнении команд, возлагается в основном на аппаратуру. Однако, как мы увидим в дальнейшем, при этом подходе неприятные усложнения коснулись и операционной системы.

Как уже упоминалось, большинство центральных процессоров, за исключением самых простых, используемых во встраиваемых системах, имеют два режима работы: режим ядра и пользовательский режим (режим пользователя). Обычно режимом управляет специальный бит в слове состояния программы — PSW. При работе в режиме ядра процессор может выполнять любые команды из своего набора и использовать лю- бые возможности аппаратуры. На настольных и серверных машинах операционная система обычно работает в режиме ядра, что дает ей доступ ко всему оборудованию. На большинстве встроенных систем в режиме ядра работает только небольшая часть операционной системы, а вся остальная ее часть — в режиме пользователя.

Пользовательские программы всегда работают в режиме пользователя, который допу- скает выполнение только подмножества команд и дает доступ к определенному подмно- жеству возможностей аппаратуры. Как правило, в пользовательском режиме запрещены все команды, касающиеся операций ввода-вывода и защиты памяти. Также, разумеется, запрещена установка режима ядра за счет изменения значения бита режима PSW.

Для получения услуг от операционной системы пользовательская программа должна осуществить **системный вызов**, который перехватывается внутри ядра и вызывает операционную систему. Инструкция перехвата *TRAP* осуществляет переключение из пользовательского режима в режим ядра и запускает операционную систему. Когда обработка вызова будет завершена, управление возвращается пользовательской про- грамме и выполняется команда, которая следует за системным вызовом. Подробности механизма системного вызова будут рассмотрены в этой главе чуть позже, а сейчас его следует считать специальной разновидностью инструкции вызова процедуры, у которой есть дополнительное свойство переключения из пользовательского режима в режим ядра. В дальнейшем для выделения в тексте системных вызовов будет ис- пользоваться такой же шрифт, как в этом слове: *read*.

Конечно, компьютеры имеют и другие системные прерывания, не предназначенные для перехвата инструкции выполнения системного вызова. Но большинство других системных прерываний вызываются аппаратно для предупреждения о возникновении исключительных ситуаций, например попыток деления на нуль или исчезновении порядка при операции с плавающей точкой. Во всех случаях управление переходит к операционной системе, которая и должна решать, что делать дальше. Иногда работа программы должна быть прервана сообщением об ошибке. В других случаях ошибка может быть проигнорирована (например, при исчезновении порядка числа оно может быть принято равным нулю). Наконец, когда программа заранее объявила, что с некото- рыми из возможных ситуаций она собирается справляться самостоятельно, управление должно быть возвращено программе, чтобы она сама разрешила возникшую проблему.

**1.3.2. Многопоточные и многоядерные микропроцессоры**

Закон Мура гласит, что количество транзисторов на одном кристалле удваивается каждые 18 месяцев. Этот «закон», в отличие от закона сохранения импульса, не имеет