5.1 Процессор и его функции

Центральный процессор - это устройство компьютера, которое выполняет операции по обработке данных и управляет периферийными устройствами.

Современные процессоры выполняются в виде микропроцессоров.

Физически **микропроцессор** представляет собой **интегральную схему** — тонкую пластинку кристаллического кремния прямоугольной формы площадью всего несколько квадратных миллиметров, на которой размещены схемы, реализующие все функции процессора.

В большинстве процессоров реализованы принципы фон Неймана в следующем виде:

• оперативная память (ОП) организована как совокупность машинных слов (МС) фиксированной длины или разрядности (имеется в виду количество двоичных единиц или бит, содержащихся в каждом МС).

• ОП образует единое адресное пространство, адреса МС возрастают от младших к старшим;

• в ОП размещаются как данные, так и программы, причем в области данных одно слово, как правило, соответствует одному числу, а в области программы — одной команде (машинной инструкции — минимальному и неделимому элементу программы);

• команды выполняются в естественной последовательности (по возрастанию адресов в ОП), пока не встретится команда управления (условного/безусловного перехода, или ветвления — branch), в результате которой естественная последовательность нарушится;

• ЦП может произвольно обращаться к любым адресам в ОП для выборки и/или записи в МС чисел или команд.

В обобщенном виде функционирование процессора может быть представлено как циклическое чередование двух этапов – (1) *выборки (чтения) команд* из памяти и их *дешифрации*, и (2) *выполнения команд.*

Выборка (чтение) команд является **автоматическим** процессом, происходящим под воздействием импульсов от генератора тактовых импульсов (ГТИ), и не зависит от программиста в смысле механизма реализации, который жестко определяется аппаратной структурой процессора.

Дешифрация команды представляет собой процесс формирования последовательности управляющих сигналов для всех узлов процессора и других блоков вычислителя на основе информации (т.е. кода), содержащегося в команде.

Действия, выполняемые в соответствии с командой, могут представлять собой арифметическую или логическую обработку данных, пересылку данных, формирование адреса следующей команды или изменение режимов работы процессора. В любом случае эти действия определяются программистом в рамках имеющейся в его

распоряжении *системы команд* конкретного процессора. После выполнения действий, задаваемых командой, процессор автоматически переходит к выборке следующей команды из памяти.

Современные микропроцессоры существенно различаются набором функциональных блоков и связями между ними. Тем не менее, в структуре любого процессора можно выделить основные элементы, определяющие специфику процессора как управляющего центра вычислителя. Прежде всего, речь идет о двух блоках: блоке управления и арифметико-логическом устройстве (рис. 5.1.1).

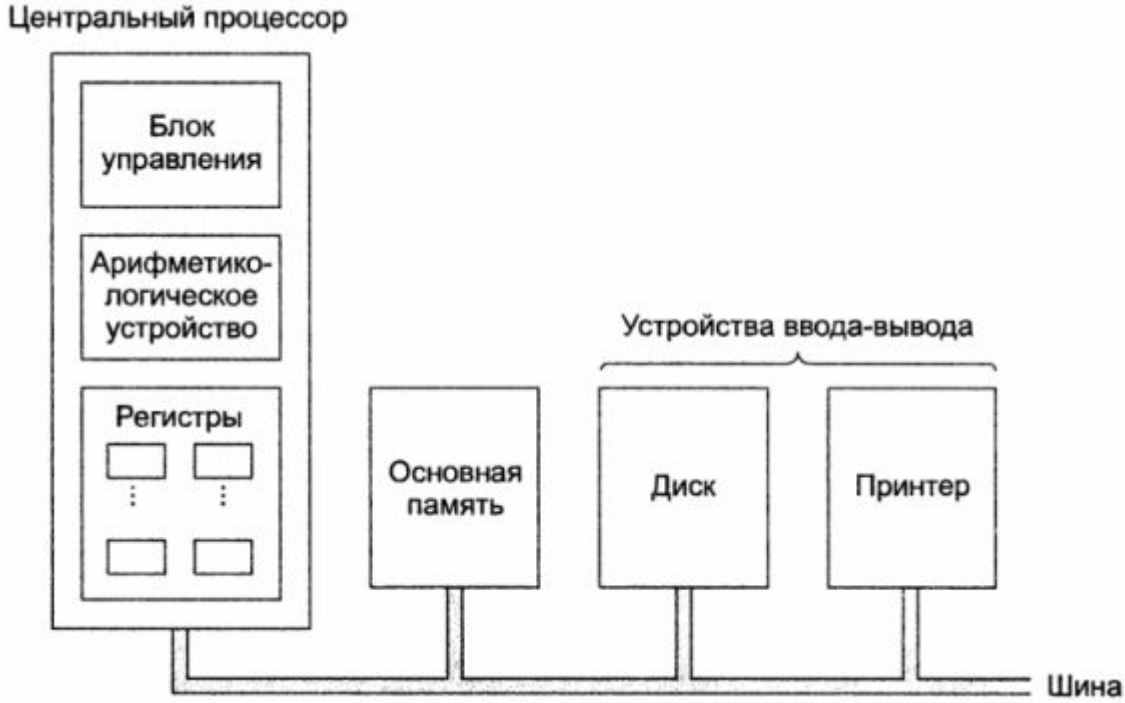

Рис. 5.1.1 Обобщенная структура процессора

Блок управления предназначен для реализации выборки команд, их дешифрации, и на основе этого – для управления обменом и обработкой информации путем генерации последовательности управляющих сигналов.

Управляющий блок выдает последовательность сигналов, которые обеспечивают выполнение данной команды. Информационные сигналы зависят не только от исходных значений обрабатываемых данных, но и от результатов получаемых в процессе обработки.

Порядок функционирования устройства базируется на следующих положениях:

1) Любая машинная команда рассматривается, как некоторое сложное действие, которое состоит из последовательности элементарных действий над словами информации микроопераций.

2) Порядок следования микроопераций зависит не только от значений преобразуемых слов, но также от их информационных сигналов, вырабатываемых операционным автоматом. Примерами таких сигналов могут быть признаки результата операции, значения отдельных битов данных и т.п.

3) Процесс выполнения машиной команды описывается в виде некоторого алгоритма в терминах микроопераций и логических условий. Описание информационных сигналов – микропрограмма.

4) Микропрограмма служит не только для обработки данных, но и обеспечивает управление работой всего устройства в целом – принцип микропрограммного управления.

Операционный блок обеспечивает выполнение определенного набора микроопераций и вычисление необходимых логических условий.

Операционный блок, согласно заданной машинной команде, генерирует необходимую последовательность сигналов, инициирующих соответствующие микрооперации, согласно микропрограмме и значениями логических условий, формируемых операционным в ходе обработки по микропрограмме. Таким образом, микропрограммы выступают, с одной стороны, в роли закона по которому выполняется обработка, с другой стороны, закон, по которому работает управляющий блок.

Арифметико-логическое устройство служит для обработки цифровой информации (арифметические и логические операции, сдвиги, анализ чисел и т.п.).

АЛУ реализует важную часть процесса обработки данных. Она заключается в выполнении набора простых операций. Арифметической операцией называют процедуру обработки данных, аргументы и результат которой являются числами (сложение, вычитание, умножение, деление). Логической операцией именуют процедуру, осуществляющую построение сложного высказывания (операции И, ИЛИ, НЕ, ...). АЛУ состоит из регистров, сумматора с соответствующими логическими схемами и блока управления выполняемым процессом. Устройство работает в соответствии с сообщаемыми ему именами (кодами) операций, которые при пересылке данных нужно выполнить над переменными, помещаемыми в регистры.

АЛУ классифицируются следующим образом:

1. По способу действий над операндами:

• АЛУ последовательного действия;

• параллельного действия.

В последовательных АЛУ действия над операндами производятся последовательно разряд за разрядом начиная с младшего. В параллельных АЛУ все разряды операндов обрабатываются одновременно.

2. По виду обрабатываемых чисел АЛУ могут производить операции над двоичными числами с фиксированной или плавающей запятой и над двоично-десятичными числами.

3. По организации действий над операндами:

• блочные;

• многофункциональные АЛУ.

В блочных АЛУ отдельные блоки предназначены для действий над двоично-десятичными числами, отдельно для действий над числами с фиксированной запятой, отдельно с плавающей запятой.

В многофункциональных АЛУ одни и те же блоки обрабатывают числа с фиксированной запятой, плавающей запятой и двоично-десятичные числа

4. То структуре:

• АЛУ с непосредственными связями;

• многосвязные.

В многосвязных АЛУ входы и выходы регистров приемников и источников информации подсоединяются к одной шине. Распределение входных и выходных сигналов происходит под действием управляющих сигналов.

В зависимости от набора и порядка выполнения команд процессоры подразделяются на четыре класса, отражающих также последовательность развития ЭВМ. Ранее других появились процессоры CISC. Затем, с целью повышения быстродействия процессоров были разработаны процессоры RISC, которые характеризуются сокращенным набором быстро выполняемых команд. Ряд редко встречающихся команд процессора CISC выполняется последовательностями команд процессора RISC. Позже появилась концепция процессоров MISC, использующая минимальный набор длинных команд. Вслед за ними возникли процессоры VLIW, работающие со сверхдлинными командами.

CISC (complex instruction set computer) есть традиционная архитектура, в которой ЦП использует микропрограммы для выполнения исчерпывающего набора команд. Они могут иметь различную длину, методы адресации и требуют сложных электронных цепей для декодирования и исполнения. В течение долгих лет производители компьютеров разрабатывали и воплощали в изделиях все более сложные и полные системы команд. Однако анализ работы процессоров показал, что в течение примерно 80 % времени выполняется лишь 20 % большого набора команд. Поэтому была поставлена задача оптимизации выполнения небольшого по числу, но часто используемых команд.

RISC (Redused Instuction Set Computer) — процессор, функционирующий с сокращенным набором команд. Так, в процессоре CISC для выполнения одной команды необходимо в большинстве случаев 10 и более тактов. Что же касается процессоров RISC, то они близки к тому, чтобы выполнять по одной команде в каждом такте. Между тем, в сокращенный набор RISC вошли только наиболее часто используемые команды.

Современные процессоры RISC характеризуются следующим:

• упрощенный набор команд, имеющих одинаковую длину;

• большинство команд выполняются за один такт процессора;

• отсутствуют макрокоманды, усложняющие структуру процессора и уменьшающие скорость его работы;

• взаимодействие с оперативной памятью ограничивается операциями пересылки данных;

• резко уменьшено число способов адресации памяти (не используется косвенная адресация);

• используется конвейер команд, позволяющий обрабатывать несколько из них одновременно;
• применяется высокоскоростная память.

Процессор **MISC** — MISC processor, работающий с минимальным набором длинных команд.

Увеличение разрядности процессоров привело к идее укладки нескольких команд в одно слово (связку, bound) размером 128 бит.

Оперируя с одним словом, процессор получил возможность обрабатывать сразу несколько команд. Это позволило использовать возросшую производительность компьютера и его возможность обрабатывать одновременно несколько потоков данных. Процессор MISC, как и процессор RISC, характеризуется небольшим набором чаще всего встречающихся команд. Вместе с этим принцип команд VLIW обеспечивает выполнение группы команд за один цикл работы процессора.

Процессор **VLIW** — процессор, работающий с системой команд сверхбольшой разрядности.

Идея технологии VLIW заключается в том, что создается специальный компилятор планирования, который перед выполнением прикладной программы проводит ее анализ, и по множеству ветвей последовательности операций определяет группу команд, которые могут выполняться параллельно. Каждая такая группа образует одну сверхдлинную команду. Это позволяет решать две важные задачи.

Во-первых, в течение одного такта выполнять группу коротких («обычных») команд. И, во-вторых, упростить структуру процессора.

2.12 Программная модель (регистровая структура) процессора

Регистровая структура процессора включает в себя набор программно доступных регистров. В соответствии с этим, этот аспект достаточно часто называют программной моделью процессора. Фактически, рассмотрение этого аспекта связано с перечислением программно доступных регистров и описанием их назначения для использования (рис. 5.1.2).

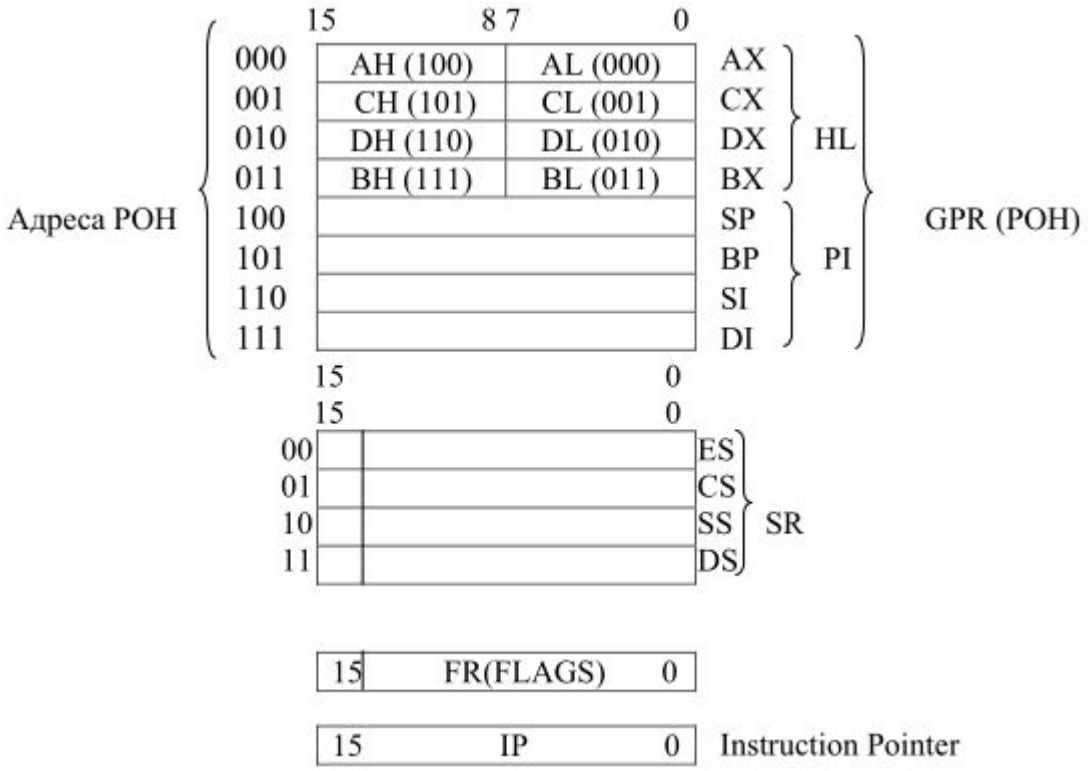

рис. 5.1.2 Регистровая структура процессора

Программная доступность регистров означает, что со стороны программы, с использованием некоторых машинных команд, может осуществляться обращение к этому регистру, либо по чтению, либо по записи.

Важной характеристикой регистра является его разрядность. Как правило, именно разрядность внутренних регистров и определяет разрядность самого процессора (Intel 8086 – 16 –разрядный). Разрядность регистра определяет количество бит информации, которое можно представить (хранить) в данном регистре.

Любой процессор в современной ЭВМ содержит собственную внутреннюю память для хранения, в основном, операндов и адресов, а также результатов выполняемых операций. Эту внутреннюю память называют регистровой памятью, или сверхоперативной памятью, чтобы подчеркнуть значительно большее её быстродействие по сравнению с оперативной (основной) памятью. Быстродействие памяти определяется так называемым временем доступа (обращения). Время доступа к регистровой памяти – единицы наносекунд, а к оперативной памяти - десятки.

В состав регистровой памяти любого процессора входят как программно доступные, так и программно недоступные регистры. Типичным примером программно недоступного регистра может служить регистр команд, в который производится выборка машинной команды из памяти перед её выполнением. Программно доступные регистры, в свою очередь разделяются на прикладные (доступные как прикладным, так и системным программам) и системные (доступные только системным программам).

Системные регистры появляются в процессорах семейства Intel 80X86, начиняя с модели i286, в которой впервые был введён защищённый режим.

В старших моделях процессора Intel используются следующие группы системных регистров.

• Управляющие регистры CR - Control Registers

• Регистры управления памятью

• Регистры отладки DR - Debug Registers

• Регистры проверки TR - Test Registers (аппаратная поддержка механизмов тестирования внутренних блоков)

Программная модель базового процессора Intel 8086 включает в себя 14 шестнадцатиразрядных регистров, 8 из них входят в состав Регистров Общего Назначения (РОН) - General Purpose Registers (GPR). Группа этих регистров предназначена как для хранения операндов (результатов), так и адресов.

В принципе, существует два диаметрально противоположных подхода к использованию регистров процессора:

1) Полная специализация регистров, когда каждый регистр используется только по одному специальному назначению.

2) Полная универсализация, когда каждый регистр можно использовать по любому назначению.

В процессорах фирмы Intel используется промежуточный подход. Это означает, что, в принципе, за каждым регистром закреплена его определенная функциональная специализация. Например, функционально специализированный регистр CX - Counter Register. Его специализация проявляется при выполнении команд циклов (LOOP), команд сдвигов (SAR) или команд обработки строк (MOVS, CMPS). Эта специализация отражается в наименовании регистра. Однако наличие специализации у регистров не мешает их использованию для других целей (не по прямому назначению). Например, в регистр CX может быть помещён операнд для какой - либо арифметической команды.

Использование регистров по их прямому назначению позволяет существенно сократить длину машинного (объектного) кода программы за счёт использования неявной адресации (операнд или адрес не задаётся, а подразумевается по умолчанию).

Следующая группа регистров программной модели - 4 Сегментных Регистра - Segment Registers (SR). С использованием этих регистров реализуется простейшая модель сегментирования памяти. В сегментных регистрах содержатся базовые (начальные) адреса 4 сегментов памяти по наименованию регистров:

• Code Segment (сегмент кода)

• Stack Segment(сегмент стека)

• Data Segment(сегмент данных)

• Extra Segment(дополнительный сегмент)

Модель памяти в процессоре Intel 8086 предполагает формирование физического адреса как суммы двух компонент: базовый адрес сегмента и

Offset (внутрисегментное смещение). При суммировании компонент первая составляющая сдвигается влево на 4 разряда, в итоге получается двадцатиразрядная сумма, представляющая собой физический адрес, который и выставляется на внешнюю шину адреса. В процессоре Intel 8086 используется мультиплексированная шина адрес/данные, т. е. по одним и тем же проводам, но в разные моменты времени передаются адрес и данные. В соответствии с принципами формирования физического адреса, в сегментных регистрах находятся старшие 16-ти разрядные компоненты 20-ти разрядных базовых адресов сегментов.

В соответствии с этим подходом, границы сегментов физической памяти выравниваются на 16-ти байтную границу (4 младших нуля в адресе), которую принято называть границей параграфа.

Выбор внутрисегментного смещения (Offset) определяется видом обращения к памяти. Например, при выборке команды в качестве компонент адреса используется пара CS - IP.

Регистр IP (Instruction Pointer).

Любой процессор, входящий в состав компьютера с неймановской архитектурой, в качестве обязательного элемента, содержит так называемый счётчик команд, который иначе называется программным счётчиком PC (Program Counter) или указатель команд.

Содержимое IP используется процессором при выборке очередной команды из памяти. В момент выполнения машинной команды, содержимое IP определяет адрес следующей команды.

Понятие“следующая”, в отношении команды, характеризует последовательность команд не столько в смысле их выполнения, сколько в смысле их положения в памяти. Так, например, при выполнении команд перехода, вызовов, возвратов, содержимое IP изменяется на значение адреса перехода, вызова или возврата.

Регистр FR (Flag Register).

Содержимое этого регистра используется, во-первых, для фиксации так называемых признаков результата (арифметические флаги), а так же для управления режимом работы процессора (флаги управления). Содержимое арифметических флагов используется при выполнении команд условных переходов. Значения арифметических флагов изменяются при выполнении большинства арифметических и логических команд, к флагам управления относятся:

- IF (Interrupt Flag) - Флаг Прерывания
- TF (Trace Flag) - Флаг Трассировки
- DF (Direction Flag) - Флаг Направления.

2.13 Система прерываний

По выражению Питера Нортона (Peter Norton — американский предприниматель, программист и филантроп, создатель операционной оболочки Norton Commander), прерывания являются движущей силой

компьютера. В связи с этим прерывания следует рассматривать не только и не столько как реакцию процессора на аномальные явления, а как естественный процесс, с помощью которого реализуется поддержка большинства необходимых механизмов, в частности, таких как ввод, / вывод, виртуальная память, режим разделения памяти и т.д.

Система прерываний является неотъемлемой частью любого компьютера и предназначена для обеспечения быстрой реакции процессора на ряд ситуаций, требуемых его внимания, которые могут возникнуть не только в самом компьютере, но и за его пределами.

Система прерываний является комплексом аппаратных и программных средств.

Прерывания принято разделять на два основных типа: программные; аппаратные.

Программные прерывания связаны с выполняемой программой и являются синхронными по отношению к этой программе.

Аппаратные прерывания могут возникать в произвольные моменты времени и являются асинхронными по отношению к выполняемой программе. С помощью аппаратного прерывания осуществляется взаимодействие процессора с периферийными устройствами (клавиатура, магнитные диски, таймер и т.п.), а также сообщается о различных ошибках аппаратуры (ошибка памяти, ошибка данных по шине и т.п.).

Реагируя на аппаратные прерывания, процессор должен идентифицировать его источник, сохранить минимальный контекст прерываемой программы и переключиться на специальную программу - обработчик прерывания (interrupt handler).

Действие обработчика прерывания, называемое обслуживанием прерываний, заключается в том, чтобы правильно отреагировать на прерывание от конкретного источника, например, поместить символ нажатой клавиши в буфер, считать сектор с диска, произвести инкремент системных часов и т.п. После завершения обслуживания прерывания процессор должен возвратиться к обслуживанию прерванной программы, и она должна продолжаться таким образом, как будто прерывания не было вовсе.

В настоящее время, особенно в зарубежной литературе, программные прерывания принято называть термином exception (особый случай, исключение). Для обработки особых случаев используются специальные программы-обработчики, называемые exception handler. В зависимости от способа возникновения особых случаев и возможности перезапуска (рестарта) процессора после их обработки с вызвавшей их команды принято рассматривать три вида особых случаев:

- Нарушение (отказ, ошибка) – fault;
- Ловушка – trap;
- Авария (выход из процесса) – abort.

1. *Нарушение (fault)* – это особый случай, который выявляется и обслуживается либо перед выполнением, либо во время выполнения «виновной» команды.

При обнаружении нарушения в качестве адреса возврата сохраняется адрес команды, вызвавшей это нарушение, а не адрес следующей команды. В соответствии с этим после выполнения обработчика этого нарушения осуществляется рестарт прерванной программы с команды, вызвавшей это нарушение. Типичным примером нарушения может служить не присутствие сегмента или страницы в основной памяти.

2. *Ловушка (trap)* - это особый случай, который возникает непосредственно после выполнения команды, вызвавшей этот особый случай.

В качестве адреса возврата прерванной программы используется адрес следующей программы по отношению к команде, вызвавшей срабатывание ловушки. Типичным примером ловушки может служить отладочное прерывание (ловушка пошагового режима).

3. *Авария (abort)* – является особым случаем, при котором невозможно точно локализовать вызвавшую его команду и, соответственно, осуществить рестарт (повторный запуск) программы. Авария используется для сообщения о крупных ошибках, таких как ошибки аппаратуры или ошибки в системных таблицах.

Ситуации, возникающие **внутри ЭВМ** (компьютера) и приводящих к прерыванию программы, можно разделить на следующие виды:

1. Особые случаи, возникающие при выполнении программы. К ним относятся:

а) Ошибки при выполнении арифметических операций. *Примеры:*

• переполнение при сложении и вычитании целых чисел;

• ошибка деления при выполнении одноименной операции над целыми числами;

• переполнение порядка или исчезновение переполнения порядка (антипереполнение) при выполнении арифметических операций над числами с плавающей запятой.

б) Различные некорректности, которые могут иметь место в машинных командах.

Примеры:

• некорректный код операции;

• некорректный адрес команды или операнда (например, адрес, не выровненный на целочисленную границу);

• некорректные данные (например, в качестве десятичной цифры используется двоичная тетрада, превышающая 1001).

в) Особые случаи, связанные с нарушением защиты памяти, точнее, защиты программ и данных, хранящихся в ней.

г) Особые случаи, связанные с организацией виртуальной памяти (например, отсутствие сегмента или страницы, к которым производятся обращения в основной памяти).

д) Выполнение специальных команд, имитирующих прерывание (в частности для процессоров Intel 80x86 к таким командам относится однобайтная INT, двухбайтная команда INTn с заданным типом прерывания,

INTo – прерывание по переполнению). Эти команды используются для вызова определенных функций операционных систем.

е) Особый случай, связанный с отладочным режимом выполнения программы. Например, в Intel 80x86 отладочный (пошаговый) режим выполнения программы имеет место при установке флага TF (флаг трассировки или ловушки). В старших моделях, начиная с Intel 80386, используются более мощные аппаратные средства поддержки механизма отладки программы в виде так называемых отладочных регистров DR0÷DR7(Debug Registers).

2. Запросы прерываний от внешних (периферийных) устройств или каналов (процессоров) ввода/вывода (КВВ).

Эти запросы могут иметь место в следующих случаях:

а) ВУ (внешнее устройство), готовое к обмену, требует реакции процессора для организации передачи данных;

б) завершение работы ВУ или КВВ по передаче данных;

в) особая (аварийная) ситуация в ВУ или КВВ. Например, отсутствие бумаги в принтере, нарушение контроля передаваемых данных и т.д.

3. Сбои аппаратуры, обнаруженные встроенными средствами аппаратного контроля.

4. Запросы прерывания от средств отсчета времени.

К основным ситуациям, возникающим **вне ЭВМ**, но, тем не менее, приводящим к прерываниям выполняемой программы относятся:

1. Запросы от объекта управления (например, от робота-манипулятора) -являются типичными для управляющих ЭВМ, которые функционируют в так называемом режиме реального времени (real time). Это означает, что темп их работы задается извне объектом управления.

2. Запросы прерывания от других процессоров (ЭВМ) для обеспечения синхронизации вычислительных процессов, протекающих в рамках многопроцессорной или многомашинной системы. Как частный случай можно рассматривать сопроцессорную конфигурацию компьютеров при наличии математического (арифметического) сопроцессора или сопроцессора ввода / вывода.

Аппаратные средства системы прерываний в персональных компьютерах реализуются в виде отдельной микросхемы, называемой программируемый контролер прерываний (PIC-Programmable Interrupt Controller). Одна микросхема PIC может обслуживать восемь источников прерываний. При большом количестве источников используется так называемое каскадное включение микросхем PIC, при этом одна из них выполняет функции ведущего контроллера и связана с микросхемой CPU, а остальные микросхемы – функции ведомых контроллеров (подключаются к ведущему контроллеру).

2.14 функции системы прерываний и их реализация на аппаратном и программном уровнях

Функции системы прерываний:

1. Прием и хранение запросов прерываний от многих источников.
2. Выделение наиболее приоритетного запроса из множества поступивших.
3. Проверка возможности обработки запроса центральным процессором (проверка замаскированности запросов или сравнение уровня приоритетности запросов с так называемым порогом прерываний).
4. Сохранение состояния (контекста) прерываемой программы.
5. Вызов соответствующего обработчика прерываний.
6. Обработка прерываний (выполнение программы обработчика прерываний).
7. Восстановление состояния (контекста) прерванной программой и возобновление ее выполнения.

Реализация организации системы прерываний на аппаратном и программном уровне

1. Прием и хранение запросов прерываний от многих источников.

Эта функция реализуется чисто на аппаратном уровне. Например, в микросхеме PIC имеется специальный регистр запроса прерываний, который является 8-разрядным (по числу обслуживаемых микросхем). Каждый бит этого регистра соответствует определенному источнику прерывания, и установка этого бита свидетельствует о наличии источника. Наличие запросов соответствующего типа фиксируется установкой в единицу соответствующего разряда регистра.

2. Выделение наиболее приоритетного запроса из множества поступивших.

Процедура опроса источников прерываний с целью определения наиболее приоритетного запроса обычно называется *poling*. Эта процедура в принципе может быть реализована как на аппаратном, так и на программном уровне.

Программный poling реализуется специальной программой, которая последовательно опрашивает триггера, объединенные, как правило, в единый регистр, с целью поиска одного организационного бита. В регистрах запроса биты отдельных запросов упорядочены по степени их важности (паритет). Для ускорения процесса программы poling, целесообразно использовать специальные команды типа BSF (Bit Scan Forward) и BSR (Bit Scan Reverse), с помощью которых можно выделить крайний левый (BSF) или крайний правый (BSR) организационный бит с фиксацией номера позиции или разряда этого бита.

Эти команды включены в систему команд процессора Intel, начиная с модели 80386.

Аппаратный poling может быть реализован на основе многотактной схемы, используемой в качестве основного элемента двоичный счетчик, либо с помощью однотактной схемы, называемой дейзи-цепочкой.

В микросхему PIC встроен механизм аппаратного полинга на основе дейзи-цепочки, но имеется принципиальная возможность реализации и программного полинга.

3. Проверка возможности обработки запроса центральным процессором. Отношение ЦП к поступившим запросам прерывания выражается с помощью одного из двух механизмов:

1. *Механизм масок* – используется в PC на базе процессоров Intel, а также в Main Frame (компьютер среднего класса) фирмы IBM.

2. *Порог прерываний* – используется в миникомпьютерах с архитектурой DEC (Digital Equipment Corporation), а также PC на базе процессоров фирмы Motorola.

Механизм масок основан на использовании специального бита для каждого запроса прерывания, с помощью которого разрешается или запрещается его обработка этого запроса. Так, например, в микросхеме PIC имеется специальный 8-разрядный регистр маски MR: установка бита в 1 соответствует разрешению обработки этого запроса, а сброс в 0 – запрещению (маскированию) этого запроса. В качестве общей маски запросов прерываний от внешних источников может рассматриваться флаг IF.

Порог прерываний представляет собой собственный приоритет процессора, а точнее, уровень приоритета программы, выполняемой им в текущий момент времени. Этот порог фиксируется в специальном управляющем слове, называемом словом состояния процессора (PSW). Все запросы от внешних устройств распределяются по уровням приоритетов в зависимости от линий запросов, к которым они подключаются (каждой линии соответствует свой приоритет).

С помощью специальной схемы, называемой арбитром, производится выделение наиболее приоритетного запроса, уровень (приоритет) которого сравнивается с порогом прерывания. Если уровень поступившего запроса не больше порога прерывания, то его обслуживание откладывается до того момента, пока не произойдет снижение порога прерывания.

Дальнейшим развитием механизма масок является использование так называемой иерархии масок, т.е. разделение масок прерываний на ряд уровней (как правило, на два).

Типичным примером подобной иерархии масок может служить:

1) Маскирование запросов внешних прерываний. Глобальной маской является флаг IF, разрешающий (IF=1) или запрещающий (IF=0) обработку запросов от всех ВУ. Локальные маски от запросов ВУ находятся в регистре MR контроллера прерываний.

2) Маскирование особых случаев арифметического сопроцессора (ASP) или блока АСП (FPU). В управляемом регистре АСП (FPU) крайние правые шесть бит являются локальными масками различных особых случаев,

имеющих место при выполнении команд FPU. К ним относятся недействительная операция, денормализованный операнд, деление на ноль, переполнение, антипереполнение, потеря точности. Кроме того, в этом же регистре (CR) имеется глобальная маска (IEM), с помощью которой можно разрешить или запретить обработку всех особых случаев.

Эта функция (проверка возможности обработки запроса ЦП) реализуется чисто на аппаратном уровне.

4. Сохранение состояния (контекста) прерываемой программы. При сохранении контекста прерываний на аппаратном уровне сохраняется минимальная часть этого контекста, обеспечивающая возможность последующего возврата в прерываемую программу. В минимальную часть контекста прерываемой программы, как правило, включается адрес возврата, и, во-вторых, регистр состояний (флагов). Содержимое остальных регистров процессора, которые могут быть изменены при выполнении программы-обработчика, сохраняются на программном уровне. Действия, связанные с сохранением регистров составляют начальную фазу обработчика прерываний. В тех случаях, когда выход на обработку прерывания сопровождается переключением задач, сохранение всего контекста прерываемой программы реализуется на аппаратном уровне с использованием специально выделенной области памяти. В частности в процессорах фирмы Intel эта область называется TSS (Task State Segment) – сегмент состояния задачи. Как правило, для сохранения прерванной программы используется стек.

Эта функция (сохранение состояния (контекста) прерываемой программы) реализуется частично на аппаратном и частично на программном уровнях. Исключением является переключение задач при выходе на обработчик прерывания, когда сохранение всего контекста прерываемой программы реализуется на аппаратном уровне.

5. Вызов соответствующего обработчика прерываний. Вызов обработчика прерываний реализуется чисто на аппаратном уровне и

предполагает загрузку начального адреса программы-обработчика, обычно называемого вектором прерываний в соответствующие регистры ЦП (для процессоров INTEL такими регистрами являются CS и IP).

6. Обработка прерываний. Обработка прерываний реализуется на программном уровне путем выполнения соответствующей программы-обработчика.

7. Восстановление состояния (контекста) прерываемой программой и возобновление ее выполнения.

Эта функция является обратной функции (4) и обычно по аналогии с функцией (4) реализуется частично на аппаратном и частично на программном уровнях.

Восстановление регистров на программном уровне реализуется в завершающей части программы обработчика прерываний. При реализации обработчика в виде отдельной задачи эта функция целиком реализуется на

аппаратном уровне, т.к. возврат в прерванную программу реализуется обратным переключением задач.