3.4. Типы данных

Основными типами данных в компьютерах являются байты, слова, двойные слова и квадрослова (рис.2.12.).

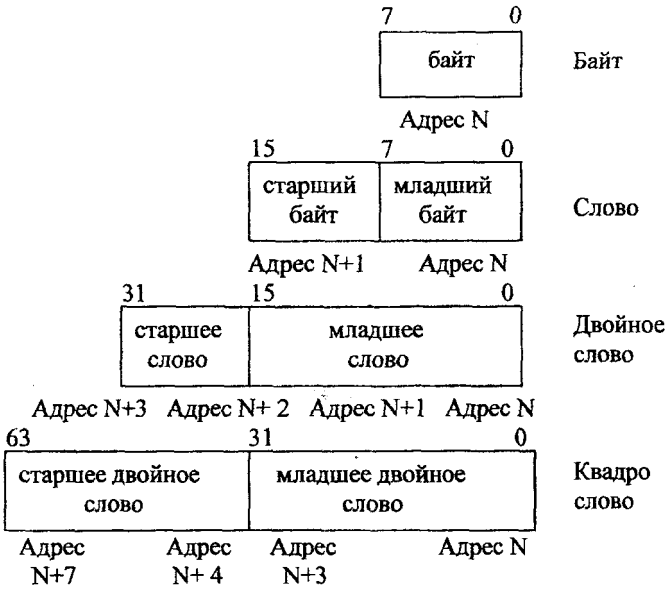


Рис. 2.12. Основные типы данных

Каждый из представленных на рис. 2.12 типов данных может начинаться с любого адреса: это означает, что слово не обязано начинаться с четного адреса; двойное слово - с адреса, кратного 4 и т.д. Таким образом достигается максимальная гибкость структур данных и эффективность использования памяти.

Однако обмен данными между процессором и памятью осуществляется в Pentium через 64-битовую ШД (i486 - 32 р.) и для достижения максимальной производительности этого обмена желательно выравнивать слова по чётным адресам, двойные слова - по адресам, кратным 4 и т.д.

На базе основных типов данных строятся все остальные типы, распознаваемые командами процессора.

Данные со знаком

На рис. 2.13 приведены 4 формата данных со знаком с фиксированной точкой.

Представление таких данных и выполнение операций производится в дополнительном коде.

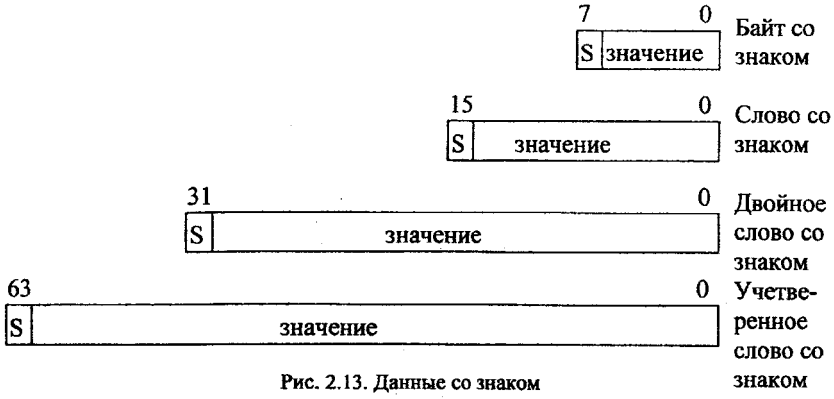


Рис. 2.13. Данные со знаком

Данные без знака

На рис. 2.14 показаны три формата данных без знака-

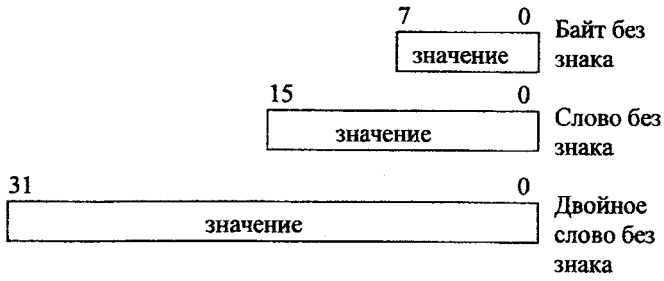


Рис. 2.14. Данные без знака

Данные в формате с плавающей точкой

Формат включает три поля: знака, мантиссы и порядка (рис. 2.15). Поле мантиссы содержит значащие биты числа, а поле порядка содержит степень 2 и определяет масштабирующий множитель для мантиссы. Поддерживаются блоком FPU.

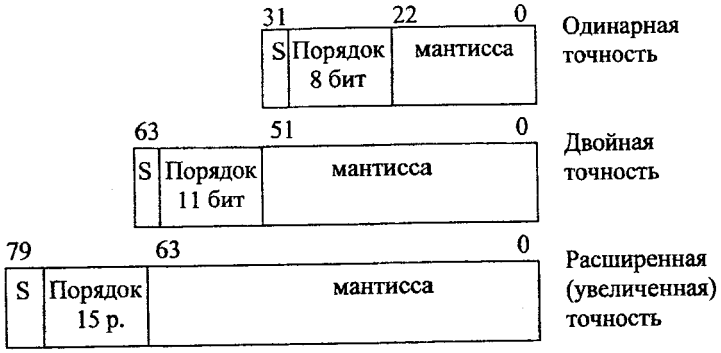


Рис.2.15. Форматы данных с плавающей точкой

Двоично-десятичные данные (BCD)

На рис. 2.16 приведены форматы двоично-десятичных данных.

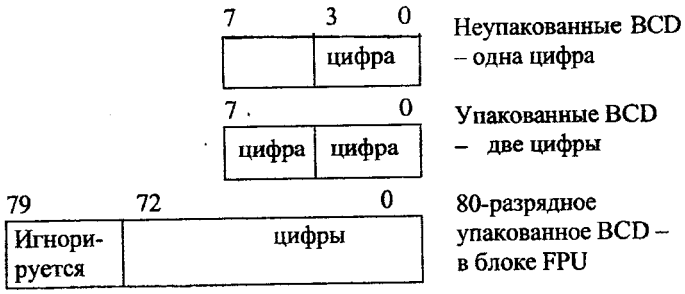


Рис.2.16. Форматы двоично-десятичных данных

Данные типа строка

Строка представляет собой непрерывную последовательность бит, байт, слов или двойных слов (рис. 2.17). Строка бит может быть длиной до 1 Гби-та, а длина остальных строк может составлять от 1 байта до 4 Гбайтов. Поддерживается АЛУ.

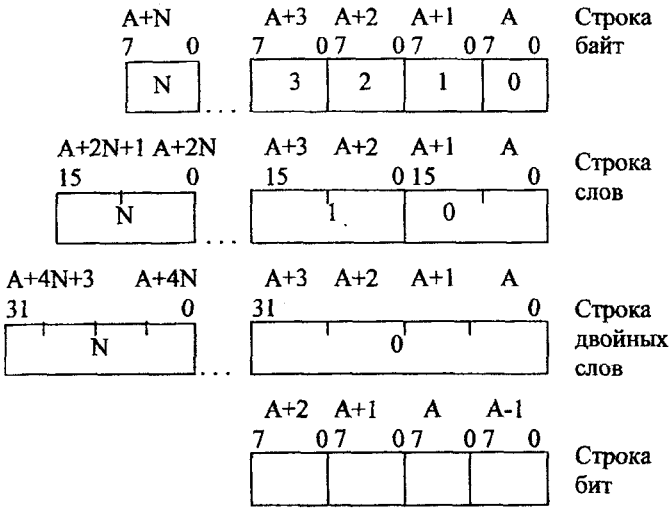


Рис.2.17. Данные типа строка

Символьные данные

Поддерживаются строки символов в коде ASCII и арифметические операции (сложение, умножение) над ними (рис. 2.18). Поддержка осуществляется блоком АЛУ.

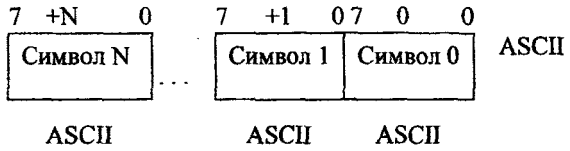


Рис.2.18. Символьные данные

Данные типа указатель

Указатель содержит величину, которая определяет адрес фрагмента данных. Поддерживается два типа указателей, приведенных на рис. 2.19.

Диапазон представления целых чисел лежит в интервале от -2^{64} до 2^{64}. Диапазон нормализованных чисел с двойной точностью - от $\pm 2{,}23 \times 10^{-308}$ до $\pm 1{,}79 \times 10^{-308}$, а с расширенной точностью - от $\pm 3{,}37 \times 10^{-4932}$ *до* $\pm 1{,}18 \times 10^{4932}$.

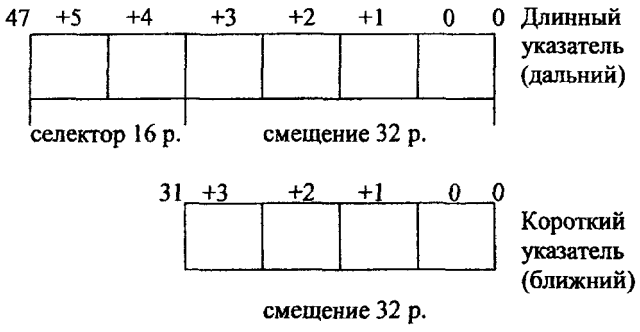


Рис.2.19. Данные типа указатель

3.5. Теги и дескрипторы. Самоопределяемые данные

Одним из эффективных средств совершенствования архитектуры современных ЭВМ является теговая организация памяти, при которой каждое хранящееся в памяти (или регистре) слово снабжается указателем - **тегом** (рис. 2.20,а). Последний определяет тип

данных - целое двоичное число, число с плавающей точкой, десятичное число, адрес, строка символов, дескриптор и т.д.

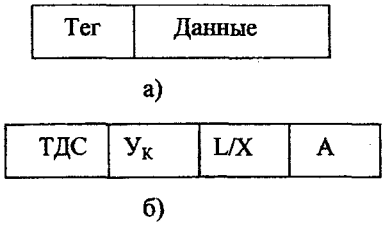


Рис.2.20. Структура описания данных: а) с теговой организацией памяти; б) дескриптор данных

В поле тега обычно указывается не только тип, но и длина (формат) и некоторые другие его параметры. Теги формируются компилятором.

Наличие тегов придает хранящимся в машине данным свойство самоопределяемости, вносящее принципиальные особенности в архитектуру и функционирование ЭВМ.

Отметим, что ЭВМ с теговой памятью (самоопределяемыми данными) выходит за рамки модели вычислительной машины фон Неймана, где тип (характер) данного определяется командой, использующей данное в качестве операнда. В обычных ЭВМ, соответствующих классической модели фон Неймана, тип данных — операндов и **их** формат задаются кодом операции команды, а в раде случаев размер (формат) определяется следующими полями команды.

Теговая организация памяти позволяет достигнуть инвариантности команд относительно типов и форматов операндов, что приводит к значительному сокращению набора команды машины. Это упрощает и делает более регулярной структуру процессора, облегчает работу программиста, в том числе при отладке программ, упрощает компиляторы и сокращает затраты времени на компиляцию (отпадает необходимость выбора типа команды в зависимости от типа данных), облегчает обнаружение ошибок, связанных с некорректным заданием типов данных (например, при попытке сложить адрес с числом с плавающей точкой).

Теговая организация памяти способствует реализации принципа независимости программ от данных.

И, наконец, нечто неожиданное. Использование тегов приводит к экономии памяти, так как в программах обычных машин имеется большая информационная избыточность на задание типов и размеров операндов при их использовании несколькими командами.

В качестве недостатка теговой организации памяти можно указать на некоторое замедление работы процессора из-за того, что установление соответствия типа команды типу данных, в обычных ЭВМ выполняемое на этапе компиляции, при использовании тегов переносится на этап выполнения программы.

В архитектуре некоторых ЭВМ используются **дескрипторы** — служебные слова, содержащие описание массивов данных и команд, причем дескрипторы могут употребляться как в машинах с теговой организацией памяти, так и без тегов.

Дескриптор содержит сведения о размере массива данных, его местоположении (в ОП или внешней памяти), адресе начала массива, типе данных, режиме защиты данных (например, запрет записи в ячейки массива) и некоторых других параметрах данных. Отметим, что задание в дескрипторе размера массива позволяет контролировать выход за границу массива при индексации его элементов. На рис. 2.20,6 в качестве примера представлен один из видов дескрипторов - дескриптор данных.

Дескриптор содержит специфический тег — ТДС, указывающий, что данное слово является дескриптором определенного вида; Ук — группа указателей; А — адрес начала массива данных; L — длина массива; Х — индекс.

Использование в архитектуре ЭВМ дескрипторов подразумевает, что обращение к

информации в памяти производится через дескрипторы, которые при этом можно рассматривать как дальнейшее развитие аппарата косвенной адресации.

Адресация информации в памяти может осуществляться с помощью цепочки дескрипторов, при этом реализуется многоступенчатая косвенная адресация. Более того, сложные многомерные массивы данных (таблицы и т. п.) эффективно описываются древовидными структурами дескрипторов.

4. ФУНКЦИОНАЛЬНАЯ И СТРУКТУРНАЯ ОРГАНИЗАЦИЯ ЦЕНТРАЛЬНОГО ПРОЦЕССОРА ЭВМ

В области вычислительной техники различают процессоры центральные, специализированные, ввода-вывода, передачи данных и коммуникационные.

4.1. Назначение и структура центрального процессора

Центральный процессор — основное устройство ЭВМ, которое наряду с обработкой данных выполняет функции управления системой: инициирование ввода-вывода, обработку системных событий, управление доступом к основной памяти и т.п.

Организация центрального процессора (ЦП) определяется архитектурой и принципами работы ЭВМ (состав и форматы команд, представление чисел, способы адресации, общая организация машины и её основные элементы), а также технико-экономическими показателями.

Логическую структуру ЦП представляет ряд функциональных средств (рис. 3.1): средства обработки, средства управления системой и программой, локальная память, средства управления вводом-выводом и памятью, системные средства.

Средства обработки обеспечивают выполнение операций с фиксированной и плавающей точкой, операций с десятичными данными и полями переменной длины. Локальная память состоит из регистров общего назначения и с плавающей точкой, а также управляющих регистров. К средствам управления памятью относятся средства управления доступом к ОП и предвыборкой команд, буферная память, средства защиты памяти. Средства управления вводом-выводом обеспечивают приоритетный доступ программ через контроллеры (каналы) к периферийному оборудованию. К системным средствам относятся средства службы времени: часы астрономического времени, таймер, коммутатор и т.д.

Существует обязательный (стандартный) минимальный набор функциональных средств для каждого типа центрального процессора. Он включает в себя: регистры общего назначения, средства выполнения стандартного набора операций и средства управления вычислительным процессом. Конкретная реализация ЦП может различаться составом средств, способом их реализации, техническими параметрами.

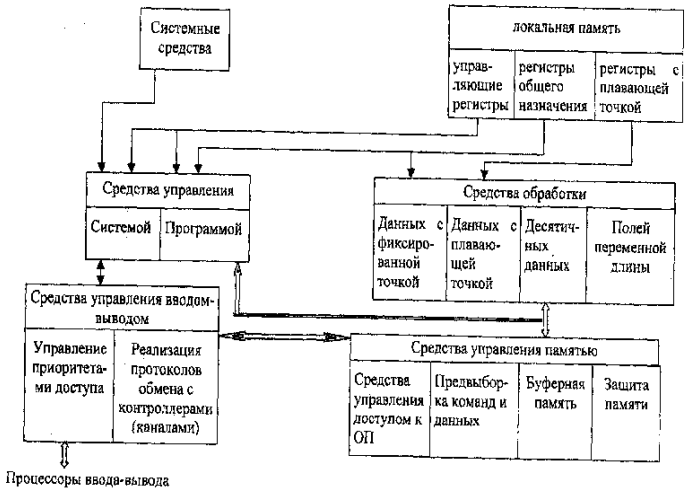


Рис.3.1. Логическая организация ЦП

Структурно все функциональные средства разбиваются на следующие устройства (рис. 3.2): центральное устройство управления (ЦУУ), арифметико-логическое устройство (АЛУ), устройство управления памятью (УУП), сверхоперативное запоминающее устройство (СОЗУ), устройство предвыборки команд и данных (УП) и интерфейс магистрали (ИМ).

Центральное устройство управления включает дешифратор команд, блок управления и блок прерываний. Дешифратор команд дешифрирует команды, которые поступают из блока предварительной выборки (очереди команд). Блок управления (БУ) формирует последовательности управляющих сигналов, которые поступают на все блоки процессора, обеспечивающие выполнение очередной команды и переход к следующей. Блок прерываний проводит анализ запросов на прерывания, формирует сигнал прерывания работы процессора и код (вектор) запроса с наивысшим приоритетом.

Арифметико-логическое устройство выполняет все арифметические и логические операции набора команд ЭВМ. В состав устройства входят традиционные арифметико-логические блоки, специализированные аппаратные средства (блок ускоренного умножения), буферные и рабочие регистры, иногда собственный блок управления. Во многих случаях выполнение операций с плавающей точкой осуществляется в отдельном блоке (процессоре), который имеет собственные регистры данных, управления и работает параллельно с центральным процессором.

Сверхоперативное ЗУ (регистровый файл) содержит регистры общего назначения (РОН), в которых хранятся данные и адреса.

Устройство управления памятью (диспетчер памяти) предназначено для сопряжения центрального процессора и подсистемы ввода-вывода с оперативной памятью. Оно состоит из блока сегментации и блока страничной адресации, осуществляющих двухступенчатое формирование физического адреса ячейки памяти: сначала в пределах сегмента, а затем в пределах страницы. Наличие блоков сегментации и страничной адресации, их одновременное функционирование обеспечивают максимальную гибкость проектируемой системы. Сегментация полезна для организации в памяти локальных модулей и является инструментом программиста, в то время как страницы нужны системному программисту для эффективного использования физической памяти системы.

Устройство предвыборки команд и данных включает блок предвыборки команд и внутреннюю кэш-память (кэш-память первого уровня). Первый осуществляет заполнение очереди команд, причем выборка из памяти производится в промежутках между магистральными циклами команд. Внутренняя кэш-память позволяет существенно повысить производительность процессора за счет буферизации в ней часто используемых

команд и данных, сокращения числа обращений к оперативной памяти.

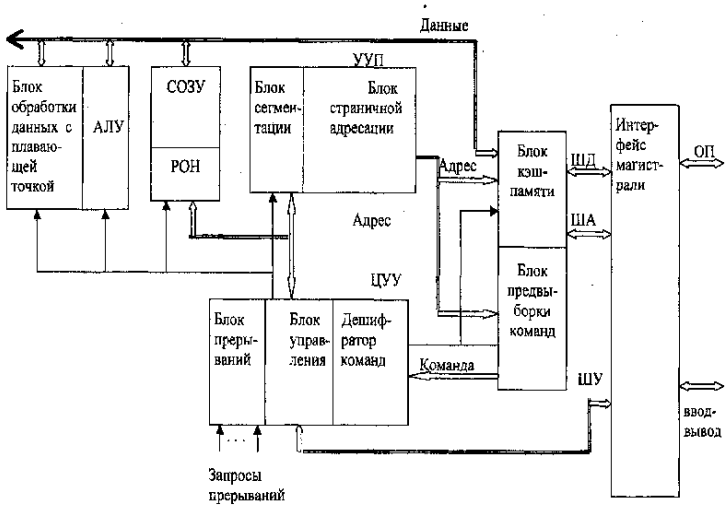


Рис.3.2.Структурная схема процессора

Интерфейс магистрали реализует протоколы обмена центрального процессора с памятью, контроллерами (каналами) ввода-вывода, другими активными устройствами системы. Обмен осуществляется с помощью шин данных, адреса и управления. Состав линий управления, тактовая сетка, магистральные циклы обмена существенно отличаются у различных типов процессоров.

В современных суперскалярных процессорах используется целый ряд параллельно функционирующих исполнительных устройств (от 2 до 6 устройств). В их состав могут входить:

- несколько целочисленных устройств;
- устройство плавающей точки;
- устройство выполнения переходов;
- устройство загрузки/записи.

Устройство выполнения переходов обрабатывает команды условных переходов. Если условия перехода доступны, то решение о направлении перехода принимается немедленно, в противном случае выполнение последующих команд продолжается по предположению (спекулятивно).

Пересылки данных между кэш-памятью данных, с одной стороны, и регистрами общего назначения и регистрами плавающей точки, с другой стороны, обрабатываются устройством загрузки/записи.

4.2. Регистровые структуры центрального процессора

Набор регистров и их структуры рассмотрим на примере процессоров с интеловской архитектурой. Можно выделить следующие группы регистров:

1. Основные функциональные регистры:

- регистры общего назначения (РОНы);
- указатель команд;
- регистр флагов;
- регистры сегментов.

2. Регистры процессора обработки чисел с плавающей точкой (FPU):

- регистры данных;
- регистр тегов;

- регистр состояния;
- регистр указателей команд и данных FPU;
- регистр управления FPU.

3. Системные регистры:

- регистры управления микропроцессора;
- регистры системных адресов.

4. Регистры отладки и тестирования.

Все 16-разрядные регистры микропроцессоров 8086, 80186, 80286 входят в состав набора 32-разрядных регистров. Регистры первых двух групп используются при выполнении прикладных программ, третьей группы _ системных, четвертой — при отладке и тестировании.

4.2.1. Основные функциональные регистры

Содержимое этих регистров определяется текущей задачей, т.е. в эти регистры автоматически загружается новое значение при переключении задач.

Регистры общего назначения

Восемь 32-разрядных регистров предназначены для хранения данных и адресов. Они поддерживают работу с данными разрядностью 1, 8, 16,32 и 64 бита, битовыми полями длиной от 1 до 32 бит и адресами размером 16 и 32 бита. Младшие 16 разрядов этих регистров (рис. 3.3) доступны отдельно при использовании соответствующего имени, например регистр EAX (имя AX для 16 разрядов).

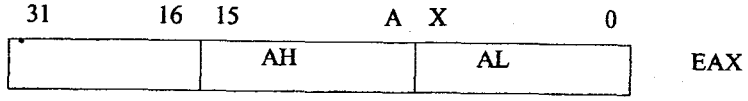


Рис. 3.3. Структура регистра общего назначения EAX

При операциях с байтами можно отдельно обращаться к младшему байту (разряды 0 - 7) и старшему байту (8-15) по именам AL и AH. Доступ к отдельным байтам обеспечивает дополнительную гибкость при операциях с данными.

Регистры сегментов и дескрипторы сегментов

Шесть 16-разрядных сегментных регистров (CS, SS, DS, ES, FS, GS) содержат значения селекторов сегментов, указывающих на текущие адресуемые сегменты памяти. С каждым из них связан программно-недоступный регистр дескриптора сегмента (рис. 3.4).

В защищенном режиме каждый сегмент может иметь размер от 1 байта до 4 Гбайт, в режиме реальных адресов максимальный размер сегмента составляет 64 Кбайта.

Селектор в CS обеспечивает обращение к текущему сегменту команд, селектор в SS — к текущему сегменту стека, селекторы в DS, ES, FS, GS — к текущим сегментам данных. Каждый регистр дескриптора содержит 32-разрядный размер сегмента и другие необходимые атрибуты.

Когда в регистр сегмента загружается новое значение селектора, содержимое соответствующего регистра дескриптора автоматически корректируется. В реальном режиме базовый адрес сегмента получается путем сдвига значения селектора на 4 разряда влево (20 разрядов), максимальный размер и атрибуты сегмента в реальном режиме имеют фиксированные значения.

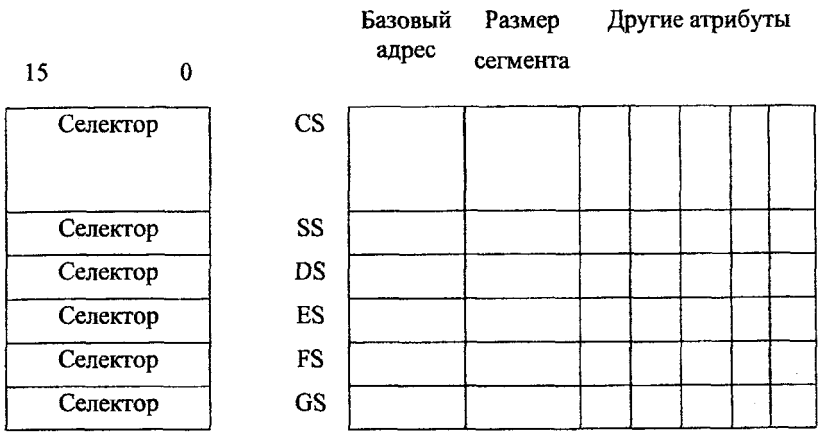


Рис. 3.4. Регистры сегментов и соответствующие регистры дескрипторов

Указатель команд

Указатель команд (рис. 3.5) представляет собой 32-разрядный регистр с именем EIP, содержимое которого используется в качестве смещения при определении адреса следующей выполняемой команды. Смещение задается относительно базового адреса сегмента команд CS. Младшие 16 бит (0 — 15) содержат 16-разрядный указатель команд с именем IP, который используется при 16-разрядной адресации.

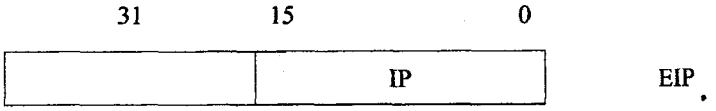


Рис. 3.5. Структура регистра указателя команд

Указатель команд непосредственно программисту недоступен. Его содержимое изменяется при выполнении команд передачи управления и прерываний.

Регистр флагов

Регистр флагов является 32-разрядным, имеет имя EFLAGS. Его разряды содержат признаки результата выполнения команды, управляют обработкой прерываний, последовательностью вызываемых задач, вводом/выводом и рядом других.

4.2.2. Регистры процессора обработки чисел с плавающей точкой

Набор регистров, входящих в блок (FPU), изображен на рис. 3.6.

Рис. 3.6. Регистры блока FPU

Регистр тегов FPU. Он содержит 16-разрядное слово, включающее восемь двухбитовых тегов. Каждый тег (признак) характеризует содержимое одного из регистров данных.

Тег определяет, является ли регистр пустым (незаполненным) - код 11 или в него введено конечное число — 00 (достоверное значение), или нуль -01, неопределенное значение (бесконечность) — 10 (нет числа и неподдерживаемый формат). Слово тегов позволяет оптимизировать функционирование FPU посредством идентификации пустых и непустых регистров данных, проверить содержимое регистра без сложного декодирования хранящихся в нем данных.

4.2.3. Системные регистры

Системные регистры управляют функционированием микропроцессора в целом и режимами работы отдельных внутренних блоков: процессора с плавающей точкой, кэш-памятью, диспетчера памяти.

Эти регистры доступны только в защищенном режиме для программ.

Набор системных регистров включает три регистра управления (CRO, CR2, CR3) и четыре регистра системных адресов и сегментов.

Регистры управления 32-разрядные, служат для фиксации общего состояния процессора. Эти регистры вместе с регистрами системных адресов хранят информацию о состоянии процессора, которое затрагивает все задачи.

4.2.4. Регистры отладки и тестирования

Микропроцессор i486, например, имеет одиннадцать регистров отладки и тестирования (все они 32-разрядные). Из них 6 программно-доступных регистров (DRO — DR3, DR6, DR7) поддерживают процесс отладки программ. Пять программно-доступных регистров (TR3 — TR7) поддерживают тестирование внутренних блоков: TR3 — TR5 используются для проверки кэш-памяти; TR6, TR7 — для тестирования механизма быстрого формирования адресов страниц.

4.3. Назначение, классификация и организация ЦУУ

Центральное устройство управления — это комплекс средств автоматического управления процессом передачи и обработки информации. ЦУУ вырабатывает управляющие сигналы (УС), необходимые для выполнения всех операций, предусмотренных системой команд, а также координирует работу всех узлов и блоков ЭВМ. В связи с этим можно считать ЦУУ преобразователем первичной командной информации, представленной программой решения задачи, во вторичную командную информацию, представляемую управляющими сигналами.

В общем случае ЦУУ формирует управляющие сигналы для реализации следующих функций:

- выборки из памяти кода очередной команды;
- расшифровки кода операции и признаков выбранной команды;
- выборки операндов и выполнения машинной операции;
- обеспечения прерываний при выполнении команд;
- формирования адреса следующей команды;
- учета состояний других устройств машины;
- инициализации работы контроллеров (каналов) ввода-вывода;
- организации контроля работоспособности ЭВМ.

Для дальнейшего рассмотрения характеристик и способов организации ЦУУ введем

ряд определений.

Элементарное машинное действие, выполняемое по одному УС, называют **микрооперацией.** Набор микроопераций, выполняемых параллельно в одном машинном такте, называют **микрокомандой** Последовательность микрокоманд, обеспечивающих выполнение некоторой операции, предписанной командой, называют **микропрограммой.**

К основным характеристикам ЦУУ следует отнести:

- принцип формирования и развертывания временной последовательности УС;
- способ построения цикла работы ЦУУ и ЭВМ в целом;
- общая организация управления ЭВМ;
- способ синхронизации узлов и блоков ЭВМ.

По принципу формирования и развертывания временной последовательности УС различают ЦУУ:

- **аппаратного (схемного) типа**, выполненным в виде управляющего автомата с жесткой логикой, в котором функции переходов и выходов реализуются набором логических элементов, а требуемое количество состояний автомата задается множеством запоминающих элементов;
- **микропрограммного типа**, в которых блок управления реализован как блок микропрограммного управления (БМУ).

По способу построения рабочего цикла различают ЦУУ:

- с прямым циклом, когда на первом этапе производится выборка из памяти команды, а затем следуют этапы выполнения машинной операции;
- с обращенным циклом, когда сначала выдаются УС микроопераций для выполнения машинной операции по коду команды, поступившей в ЦУУ на предыдущем цикле (предвыборка команд), а затем из памяти выбирается код команды, которая будет исполняться в следующем цикле;
- с совмещением во времени циклов выполнения нескольких команд (конвейером команд).

По общей организации управление может быть центральным и смешанным. В первом случае в БУ вырабатываются все УС микроопераций для всех команд, выполняемых процессором ЭВМ. Во втором случае, кроме БУ центрального устройства управления, операционные и другие устройства процессора имеют собственные блоки местного управления. В последнем случае БУ вырабатывает сигналы для запуска в работу блоков местного управления.

По способу синхронизации работы ЭВМ в зависимости от числа тактов в цикле выполнения команды, различают ЦУУ с постоянным или переменным числом тактов. В микропрограмме рабочего цикла выделяют общую и специальную части. К общей относятся микрокоманды, исполняемые в цикле любой команды: выборки команды, анализа запросов на прерывание работы процессора. Они выполняются за постоянное число тактов. К специальной части относятся микрокоманды, по которым вырабатываются УС в зависимости от содержания операционной части исполняемой команды. В этом случае количество тактов будет переменным для различных команд. В современных ЭВМ с различной структурой используемых команд число тактов в рабочем цикле зависит от формата выбираемой команды, структуры ее адресной части и длины операндов.

По принципу организации циклов различают ЦУУ:

- синхронного типа, в которых время цикла может быть постоянным или переменным;
- асинхронного типа, в которых продолжительность цикла определяется фактическими затратами времени на выполнение каждой операции. В этом случае необходимо вырабатывать сигналы об окончании операции;
- смешанного типа, где частично реализуются оба предыдущих принципа организации

циклов.

4.3.1. Центральное устройство управления микропрограммного типа

Микропрограммный принцип управления обеспечивает реализацию одной машинной команды путем выполнения определенного набора микропрограмм, интерпретирующего алгоритм выполнения данной операции. Совокупность микропрограмм, необходимая для реализации системы команд ЭВМ, хранится в специальной памяти микропрограмм. Каждая микропрограмма состоит из определенной последовательности микрокоманд, которые после выборки из памяти преобразуются в набор управляющих сигналов.

Микрокоманда (МК) имеет операционно-адресную структуру. В операционной части МК размещается информация о микрооперациях (МО), одновременно выполняемых в блоках ЭВМ под управлением данной МК. В адресной части МК находится информация, необходимая для формирования адреса следующей микрокоманды.

Существуют различные способы организации операционной части МК:

- горизонтальное микропрограммирование;
- вертикальное микропрограммирование;
- смешанное микропрограммирование.

В первом случае операционная часть МК содержит столько разрядов, сколько различных МО выполняется в ЭВМ (число управляющих точек). Каждому разряду ставится в соответствие определенный УС, под действием которого выполняется соответствующая микрооперация. Таким образом, нет необходимости в преобразовании операционной части МК в управляющие сигналы. За счет этого сокращаются затраты времени на формирование УС. Недостатком данного метода является большая длина операционной части МК, что ведет к значительным затратам памяти микропрограмм.

По второму способу из всего множества M микроопераций выделяются подмножества, содержащие не более H совместно выполняемых в каждом такте МО. Номера МО кодируются двоичным кодом, разрядность которого определяется по формуле $m \geq \log_2 M$. Операционная часть МК должна содержать H полей, каждое из которых имеет разрядность m и определяет код номера микрооперации. В результате использования данного метода уменьшается длина МК, сокращаются затраты микропрограммной памяти, но возникает необходимость в дешифрировании полей операционной части МК, что приводит к увеличению затрат времени на выработку УС.

В настоящее время наибольшее распространение получил третий способ — **смешанное микропрограммирование**, в котором сочетаются первые два способа. В этом случае операционная часть МК содержит как коды номеров микроопераций, так и сами УС, соответствующие отдельным МО.

Адресная часть МК используется для определения адреса следующей МК.

Существуют два способа адресации микрокоманд:

- принудительная адресация;
- естественная адресация.

Принудительная адресация МК заключается в том, что в каждой МК указывается адрес следующей МК. Адрес следующей МК может задаваться безусловно, независимо от значений признаков (осведомительных сигналов) или выбираться по условию, определяемому текущими значениями осведомительных сигналов, которые, в свою очередь, отображают текущее состояние операционных блоков процессора. Для этого в адресную часть МК кроме адресных полей включаются поля для задания условий (осведомительных сигналов).

При естественной адресации адрес следующей МК принимается равным увеличенному на единицу адресу предыдущей МК. В этом случае отпадает необходимость во введении адресной части в каждую МК. Если микрокоманды идут в естественном порядке, то процесс адресации реализуется счетчиком адреса МК. Для

организации безусловных или условных переходов в микропрограмму включаются дополнительные управляющие МК.

Обобщенная структура блока микропрограммного управления (БМУ) представлена на рис. 3.7. Узел ФАМ предназначен для формирования адреса очередной МК с учетом значений адресной части (АЧ) предыдущей МК и множества {x} осведомительных сигналов. Микропрограммная память (МПП) хранит микропрограммы операций и по сформированному адресу в каждом такте выдает значение очередной МК, которое записывается в регистр микрокоманд (РМК). Поля операционной части (ОЧ), выбранной МК, при необходимости дешифрируются для выработки управляющих сигналов {y}. Первоначальное обращение к какой-либо микропрограмме осуществляется по начальному адресу (НА), который соответствует коду операции выполняемой команды.

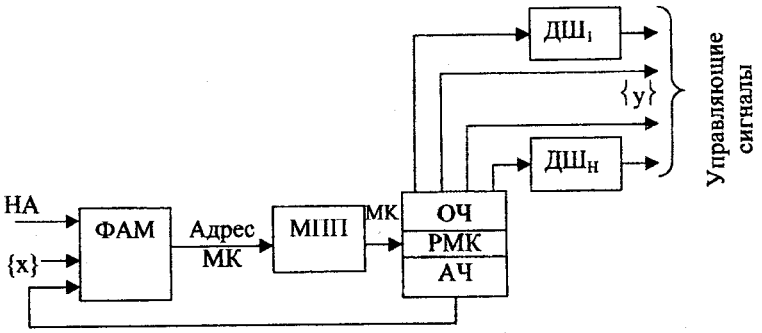


Рис.3.7. Обобщенная структура БМУ

С точки зрения физической реализации управления МПП делится на два вида: память с постоянно записанной информацией и память, допускающая перезапись информации. Память с постоянно записанной информацией (ПЗУ) работает только на чтение информации и, как правило, является более быстродействующей и простой по управлению, нежели память с перезаписью. В то же время память, допускающая перезапись, предоставляет больше дополнительных возможностей для повышения эффективности работы процессора за счет постоянного совершенствования алгоритмов выполнения операций.

Таким образом, использование в составе центрального устройства управления БМУ приводит к двухуровневому принципу управления процессом обработки данных. Первый уровень — это система команд ЭВМ (программное управление), второй — микропрограммное управление. Возникает задача организации перехода от одного уровня к другому. На рис. 3.8 приведена упрощенная структура процессора, в котором решается эта задача. По содержимому счетчика адреса команд (СЧАК) из памяти программ (кэш-памяти) выбирается команда и записывается в регистр команд (РК). Код операции из РКОП подается на дешифратор начального адреса (ДШНА), который на выходе формирует адрес первой микрокоманды микропрограммы, соответствующей данному коду операции. ДШНА реализуется на ПЗУ или ПЛМ (программируемой логической матрице). Под управлением микрокоманд выполняются все последующие действия. Адрес операнда из РА передается в память данных, осуществляется выборка операнда и занесение его в регистр общего назначения (СОЗУ) или в АЛУ. В АЛУ выполняется определенная микропрограммой операция, результат записывается в РОН или память данных.

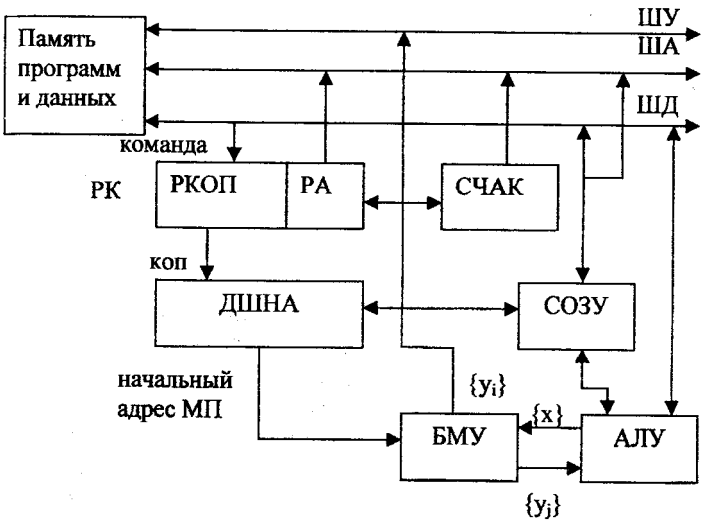


Рис.3.8. Процессор с микропрограммным управлением

Анализ аппаратурной (схемной) и микропрограммной реализации устройства управления указывает на зависимость стоимости управления от величины набора команд и их сложности. Для сокращенного набора простых команд выгоднее использовать схемное управление, что и реализуется в RISC-процессорах. При расширенном составе сложных команд (как в CISC-процессорах) наиболее эффективно, с точки зрения затрат оборудования, микропрограммное управление. Однако оно приводит к увеличению затрат времени на выработку управляющих воздействий. Основным же преимуществом микропрограммного управления является его гибкость, которая позволяет повышать эффективность серийно выпускаемых и эксплуатируемых машин за счет введения новых средств математического обеспечения, использующих дополнительный набор команд и новые функции процессора. Модернизация алгоритмов или реализация дополнительных команд легко осуществляется путем изменения содержимого микропрограммной памяти. Наглядным примером использования данной возможности является технология MMX, разработанная фирмой Intel. В серийно выпускаемый процессор Pentium были добавлены 57 новых команд для параллельной обработки видео- и аудиоинформации. Аппаратурные средства процессора остались прежними, изменению подверглась лишь микропрограммная память.

4.3.*2*. Процедура выполнения команд

Стандартные фазы работы ЦП включают в себя выборку команды, вычисление адреса и выборку операндов, выполнение команды и запись результатов, обработку прерывания, изменение состояния процессора и системы в целом.

Выборка команд (ВК) — передача содержимого счетчика команд в регистр адреса памяти, считывание команды из основной памяти в регистр команды, модификация содержимого счетчика команд для выборки следующей команды.

Выборка операнда (ВО) — вычисление адреса и обращение в основную память или к регистру локальной памяти. Операнд считывается и принимается в регистр АЛУ.

Арифметическая операция (АО) — инициализация (кодом операции) цикла работы устройства управления, которое, в свою очередь, управляет работой АЛУ, регистров и схем сопряжения. Результат выполнения операции передается в локальную или основную память и процессор переходит к выборке и выполнению следующей команды.

На рис. 3.9 показаны временные диаграммы обработки команды с разбиением на этапы (фазы) выполнения (а): последовательная обработка команд (б); обработка команд в режиме совмещения — конвейер команд (в).

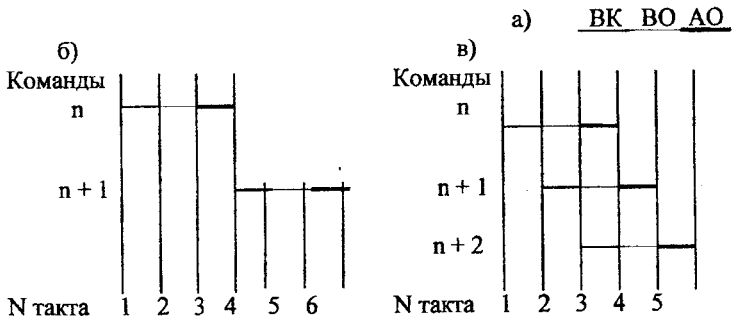


Рис. 3.9. Временные диаграммы обработки команд в процессоре:

а) этапы выполнения команды;

б) последовательное выполнение команд;

в) совмещенное выполнение команд (конвейеризация)

Совмещенные принципы обработки (конвейер команд) существенно увеличивают пропускную способность процессора, однако эффективность их использования зависит от управления (синхронизации), числа уровней обработки.

Приостановка работы конвейера вызывает любая команда условного перехода в программе или взаимозависимость команд, т. е. использование следующей командой результатов предыдущей команды.

Следует учитывать, что совмещение обработки увеличивает объем оборудования и усложняет схемы управления тем сильнее, чем больше число уровней совмещения.

Все эти обстоятельства приходится учитывать при выборе числа уровней совмещения в каждом конкретном случае для получения заданных параметров и прежде всего удельных затрат (отношение производительности к стоимости). Опыт разработки ЭВМ общего назначения и проведенные исследования показывают, что технически и экономически целесообразной является совмещенная обработка 5-6 команд.

Для обеспечения непрерывности вычислительного процесса и сглаживания влияния логической зависимости команд в структуре ЦП используется блок прогнозирования ветвлений или устройство выполнения переходов.

В большинстве современных компьютеров используется конвейер команд.

4.3.3. Принципы организации системы прерывания программ

Во время выполнения ЭВМ текущей программы внутри машины и в связанной с ней внешней среде (технологический процесс, управляемый ЭВМ) могут возникать события, требующие немедленной реакции на них со стороны машины.

Реакция состоит в том, что машина прерывает обработку текущей программы и переходит к выполнению некоторой другой программы, специально предназначенной для данного события. По завершению этой программы ЭВМ возвращается к выполнению прерванной программы.

Рассматриваемый процесс, называемый прерыванием программ, поясняется на рис. 3.10.

Принципиально важным является то, что моменты возникновения событий, требующих прерывания программ, заранее не известны и поэтому не могут быть учтены при программировании.

Каждое событие, требующее прерывания, сопровождается сигналом, который называют **запросом прерывания.**

Программу, затребованную запросом прерывания, называют **прерывающей**

программой, противопоставляя ее **прерываемой программе,** выполнявшейся в ЭВМ до появления запроса.

Запросы на прерывания могут возникать внутри самой ЭВМ и в ее внешней среде. К первым относятся, например, запросы при возникновении в ЭВМ таких событий, как появление ошибки в работе ее аппаратуры, переполнение разрядной сетки, попытка деления на 0, выход из установленной для данной программы области памяти, затребование периферийным устройством операции ввода-вывода, завершение операции ввода-вывода периферийным устройством или возникновение при этой операции особой ситуации и др. Хотя некоторые из указанных событий порождаются самой программой, моменты их появления, как правило, невозможно предусмотреть. Запросы во внешней среде могут возникать от других ЭВМ, от аварийных и некоторых других датчиков технологического процесса и т.п.

Таким образом, запросы прерывания генерируются несколькими развивающимися параллельно во времени процессами, которые в некоторые моменты требуют вмешательства процессора.

К этим процессам, в частности, относится процесс выполнения самой программы, процесс контроля правильности работы ЭВМ, операции ввода-вывода, технологический процесс в управляемом машиной объекте и др.

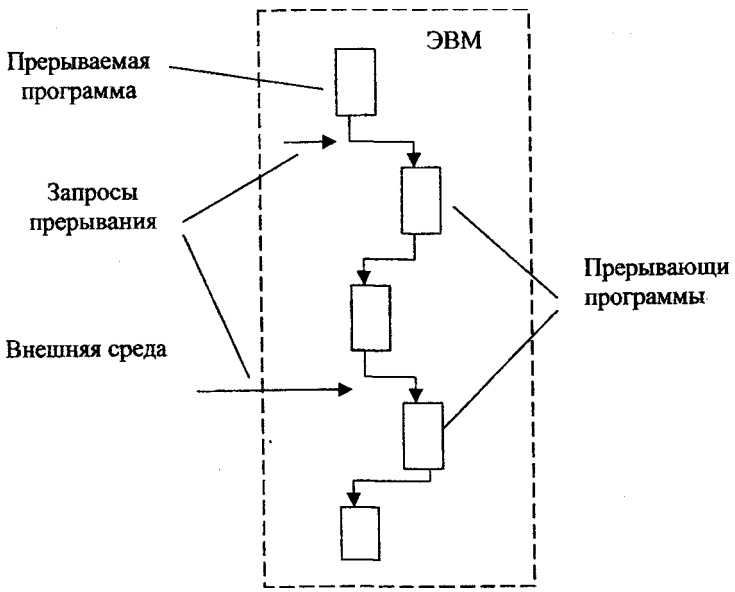


Рис.3.10. Процесс прерывания программы

Возможность прерывания программ - важное архитектурное свойство ЭВМ, позволяющее эффективно использовать производительность процессора при наличии нескольких, протекающих параллельно во времени, процессов, требующих в произвольные моменты времени управления и обслуживания со стороны процессора. В первую очередь это относится к организации параллельной во времени работы процессора и периферийных устройств машины, а также к использованию ЭВМ для управления в реальном времени технологическими процессами,

В некоторых машинах, наряду или вместо прерывания с переключением управления на другую программу, используется примитивное прерывание - так называемая приостановка, когда по соответствующему запросу приостанавливается выполнение программы и выполняется аппаратурными средствами некоторая процедура без изменения содержания счетчика команд, а по ее окончании продолжается выполнение

приостановленной программы.

Чтобы ЭВМ могла, не требуя больших усилий от программиста, реализовывать с высоким быстродействием прерывания программ, машине необходимо придать соответствующие аппаратурные и программные средства, совокупность которых получила название **системы прерывания программ.** В качестве аппаратных средств используется **контроллер прерывания** (блок прерывания).

Основными функциями системы прерывания являются:

- запоминание состояния прерываемой программы и осуществление перехода к прерывающей программе;
- восстановление состояния прерванной программы и возврат к ней. При наличии нескольких источников запросов прерывания между ними должны быть установлены **приоритетные соотношения,** определяющие, какой из нескольких поступивших запросов подлежит обработке в первую очередь, и устанавливающие: имеет право или нет данный запрос (прерывающая программа) прерывать ту или иную программу.

Характеристики системы прерывания

Для оценки эффективности систем прерывания могут быть использованы следующие характеристики.

1. Общее число запросов прерывания (входов в систему прерывания).

2. Время реакции — время между появлением запроса прерывания и моментом прерывания текущей программы. На рис. 3.11 приведена упрощенная временная диаграмма процесса прерывания.

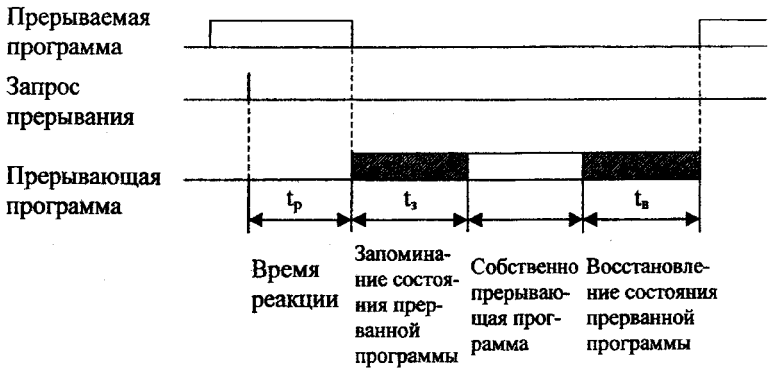


Рис.3.11. Упрощенная временная диаграмма процесса прерывания

Для одного и того же запроса задержки в исполнении прерывающей программы зависят от того, сколько программ со старшим приоритетом ждут обслуживания, поэтому время реакции определяют для запроса с наивысшим приоритетом ($t_р$).

Время реакции зависит от того, в какой момент допустимо прерывание. Большей частью прерывание допускается после окончания текущей команды. В этом случае время реакции определяется в основном длительностью выполнения команды.

Это время реакции может оказаться недопустимо большим для ЭВМ, предназначенных для работы в реальном масштабе времени. В таких машинах часто допускается прерывание после любого такта выполнения команды (микрокоманды). Однако при этом возрастает количество информации, подлежащей запоминанию и восстановлению при переключении программ, так как в этом случае необходимо сохранять также и состояние в момент прерывания счетчика тактов, регистра кода операции и некоторых других узлов, поэтому такая организация прерывания возможна только в машинах с быстродействующей сверхоперативной памятью.

Имеются ситуации, в которых желательно немедленное прерывание. Если аппаратура контроля обнаружила ошибку, то целесообразно сразу же прервать операцию, пока

ошибка не оказала влияние на следующие такты работы программы.

3. Затраты времени на переключение программ (издержки прерывания) равны суммарному расходу времени на запоминание и восстановление состояния программы (рис. 3.11):

$$t_{изд} = t_з + t_в.$$

4. Глубина прерывания - максимальное число программ, которые могут прерывать друг друга. Если после перехода к прерывающей программе и вплоть до ее окончания прием запросов прекращается, то говорят, что система имеет глубину прерывания, равную 1. Глубина равна n, если допускается последовательное прерывание до n программ. Глубина прерывания обычно совпадает с числом уровней приоритета в системе прерывания. На рис. 3.12 показаны процессы прерывания в системах с различной глубиной прерывания (предполагается, что приоритет каждого последующего запроса выше предыдущего). Система с большим значением глубины прерывания обеспечивает более быструю реакцию на срочные запросы.

Если запрос окажется не обслуженным к моменту прихода нового запроса от того же источника, то возникает так называемое **насыщение системы прерывания.**

В этом случае предыдущий запрос от данного источника будет машинально утерян, что недопустимо.

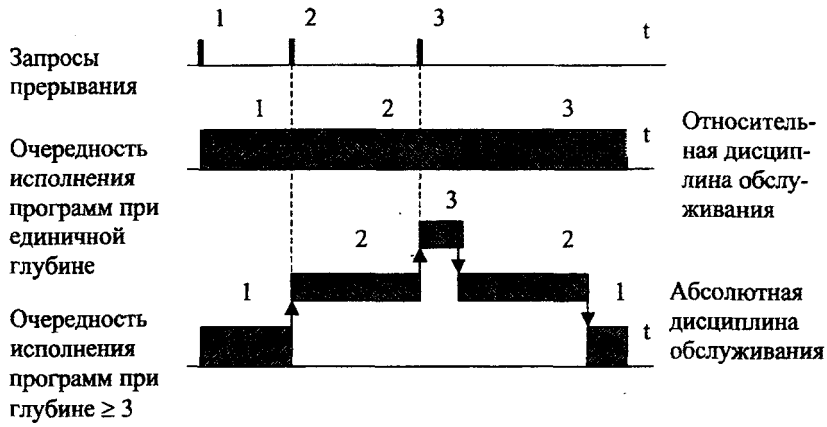


Рис.3.12. Процессы прерывания с различной глубиной прерывания и дисциплиной обслуживания

5. Число классов (уровней) прерывания. В ЭВМ число различных запросов (причин) прерывания может достигать нескольких десятков или сотен. В таких случаях часть запросов разделяют на отдельные классы или уровни.

Совокупность запросов, инициирующих одну и ту же прерывающую программу, образует класс или уровень прерывания (рис. 3.13).

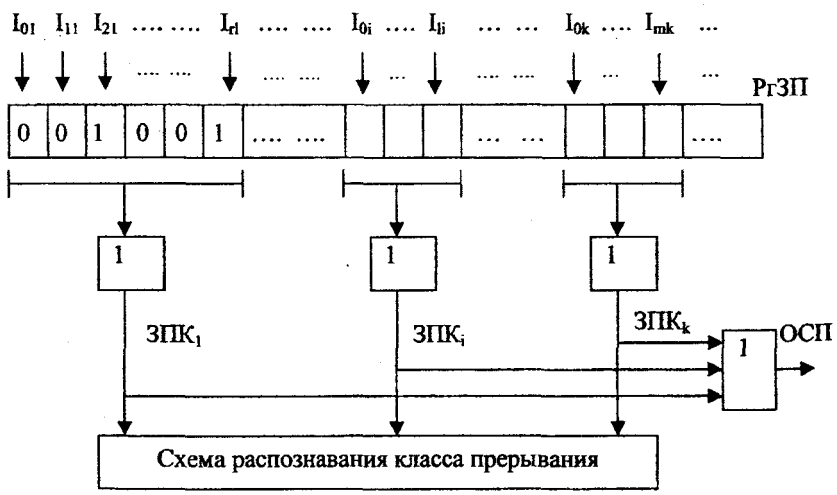


Рис.3.13. Разделение запросов на классы прерывания

Запросы всех источников прерывания поступают на регистр запросов прерывания РгЗП, устанавливая соответствующие его разряды в состояние 1, указывающее на наличие запроса прерывания определенного источника. Запросы классов прерывания $ЗПК_1$-$ЗПК_k$ формируются элементами ИЛИ, объединяющими разряды РгЗП, относящиеся к соответствующим классам (уровням). Еще одна схема ИЛИ формирует общий сигнал прерывания ОСП, поступающий в устройство управления процессора.

Информация о действительной причине прерывания, породившей запрос данного класса, содержится в коде прерывания, который отражает состояние разрядов РгЗП, относящихся к данному классу прерывания.

После принятия запроса прерывания на исполнение и передачу управления прерывающей программе соответствующий триггер РгЗП сбрасывается. Объединение запросов в классы прерывания позволяет уменьшить объем аппаратуры. но приводит к замедлению работы системы прерывания.

Программно-управляемый приоритет прерывающих программ

Относительная степень важности программ, их частота повторения, относительная степень срочности в ходе вычислительного процесса могут меняться, требуя установления новых приоритетных отношений. Поэтому во многих случаях приоритет между прерывающими программами не может быть зафиксирован раз и навсегда. Необходимо иметь возможность изменять по мере необходимости приоритетные соотношения программным путем. Приоритет между прерывающими программами должен быть динамичным, т.е. программно управляемым.

В ЭВМ широко применяются два способа программно-управляемого приоритета прерывающих программ:

- использование порога прерывания;
- использование маски прерывания.

Порог прерывания. Этот способ позволяет в ходе вычислительного процесса программным путем изменить уровень приоритета процессора (а следовательно, и обрабатываемой в данный момент на процессоре программы) относительно приоритетов запросов источников прерывания (периферийных устройств), другими словами, задавать порог прерывания, т. е. минимальный уровень приоритета запросов, которым разрешается прерывать программу, идущую на процессоре.

Порог прерывания задается командой программы, устанавливающей в регистре порога прерывания код порога прерывания. Специальная схема выделяет наиболее приоритетный запрос, сравнивает его приоритет с порогом прерывания и, если он оказывается выше порога, вырабатывает общий сигнал прерывания, и начинается

процедура прерывания.

Маска прерывания представляет собой двоичный код, разряды которого поставлены в соответствие запросам или классам (уровням) прерываний. Маска загружается командой программы в регистр маски (рис. 3.14).

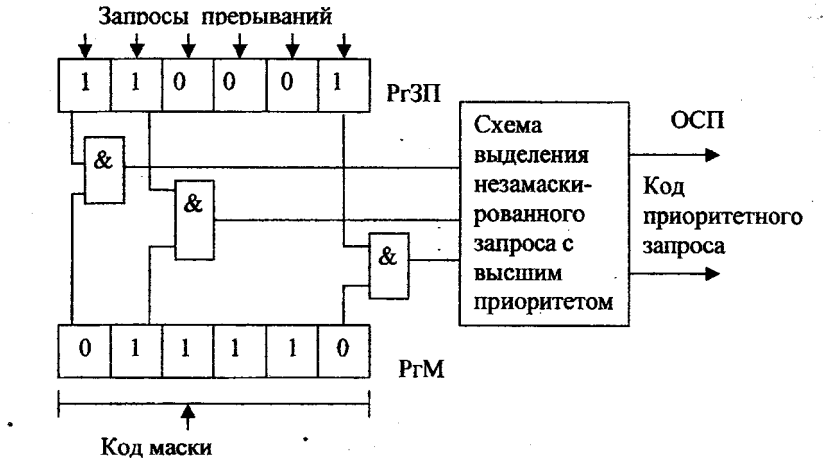


Рис.3.14. Маскирование прерываний

Состояние 1 в данном разряде регистра маски (РгМ) разрешает, а состояние 0 запрещает (маскирует) прерывание текущей программы от соответствующего запроса. Таким образом, программа, изменяя маску в регистре маски, может устанавливать произвольные приоритетные соотношения между программами без перекоммутации линий, по которым поступают запросы прерывания. Каждая прерывающая программа может установить свою маску. При формировании маски 1 устанавливаются в разряды, соответствующие запросам (прерывающим программам) с более высоким, чем у данной программы, приоритетом. Схемы И выделяют поступившие незамаскированные запросы прерывания, из которых специальная схема выделяет наиболее приоритетный и формирует код его номера.

С замаскированным запросом, в зависимости от причины прерывания, поступают двояким образом: или он игнорируется, или запоминается, с тем чтобы осуществить затребованные действия, когда запрет будет снят. Например, если прерывание вызвано окончанием операции в ПУ, то его следует, как правило, запомнить, так как иначе ЭВМ останется неосведомленной о том, что ПУ освободилось.

Прерывание, вызванное переполнением разрядной сетки при арифметической операции, следует при его маскировании игнорировать, так как запоминание этого запроса может оказать действие на часть программы или другую программу, к которым это переполнение не относится.

Организация перехода к прерывающей программе

Вектор начального состояния прерывающей программы называют **вектором прерывания.** Он содержит всю необходимую информацию для перехода к прерывающей программе, в том числе ее начальный адрес. Каждому запросу (уровню) прерывания соответствует свой вектор прерывания, способный инициировать выполнение соответствующей прерывающей программы. Векторы прерывания обычно находятся в специально выделенных фиксированных ячейках памяти (стеке).

Главное место в процедуре перехода к прерывающей программе занимает передача из соответствующего регистра (регистров) процессора в память (стек) на сохранение текущего вектора состояния прерываемой программы (чтобы можно было вернуться к ее исполнению) и загрузка в регистр (регистры) процессора вектора прерывания

прерывающей программы, к которой при этом переходит управление процессором.

Наиболее гибким и динамичным является **векторное прерывание,** при котором источник прерывания, выставляя запрос прерывания, посылает в процессор (выставляет на шины интерфейса) код адреса в памяти своего вектора прерывания.

При векторном прерывании каждому запросу прерывания или, другими словами, устройству — источнику прерывания, соответствует переход к начальному адресу соответствующей прерывающей программы, задаваемому вектором прерывания.

4.4. Назначение, классификация и организация АЛУ

Арифметико-логическое устройство (АЛУ) является одной из основных функциональных частей процессора, осуществляющей непосредственное преобразование информации.

Все операции, выполняемые в АЛУ, можно разделить на следующие группы:

- операции двоичной арифметики для чисел с фиксированной запятой;
- операции двоичной (или шестнадцатеричной) арифметики для чисел с плавающей запятой;
- операции десятичной арифметики (над числами, представленными в двоично-десятичном коде);
- операции адресной арифметики (при модификации адресов команд);
- операции специальной арифметики;
- логические операции;
- операции над алфавитно-цифровыми полями.

Современные универсальные ЭВМ обычно реализуют операции всех приведенных выше групп, а специализированные ЭВМ часто не имеют аппаратуры для обработки чисел с плавающей запятой, десятичных чисел и операций над алфавитно-цифровыми полями. В этом случае эти операции выполняются специальными подпрограммами.

Основными являются арифметические и логические операции. К арифметическим операциям относятся сложение, вычитание, вычитание модулей ("короткие операции"), умножение и деление ("длинные операции"). Группу логических операций составляют операции дизъюнкции (логическое ИЛИ) и конъюнкции (логическое И) над многоразрядными двоичными словами, сравнение кодов на равенство. Специальные арифметические операции включают в себя нормализацию, арифметический сдвиг (сдвигаются только цифровые разряды, знаковый разряд остается на месте), логический сдвиг (знаковый разряд сдвигается вместе с цифровыми разрядами). Обширна группа операций редактирования алфавитно-цифровой информации.

Для выполнения перечисленных операций в АЛУ включаются следующие функциональные узлы:

- сумматор для выполнения суммирования и других действий над кодами операндов;
- регистры для хранения кодов операндов на время выполнения действий над ними;
- сдвигатели для сдвига кода на один или несколько разрядов вправо или влево;
- преобразователи для преобразования прямого кода числа в обратный или дополнительный код;
- комбинационные схемы для реализации логических операций, мультиплексирования данных, управляемой передачи информации, формирования признаков результата и т.д.

Регистры и в некоторых случаях сумматоры имеют цепи управления приемом, выдачей и сбросом кодов операндов. Логические операции, операции сдвига и преобразования кодов могут выполняться не только специальными устройствами, но и с помощью дополнительных связей регистров и сумматора. В зависимости от типов используемых для суммирования базовых элементов различают комбинационные и накапливающие сумматоры.

Классификация АЛУ

По способу представления чисел различают АЛУ:

- для чисел с фиксированной запятой;
- для чисел с плавающей запятой;
- для десятичных чисел.

По способу действия над операндами АЛУ делятся на последовательные и параллельные. В параллельных АЛУ операнды представляются параллельным кодом и операции совершаются параллельно во времени над всеми разрядами операндов. В последовательных АЛУ операнды представляются в последовательном коде, а операции производятся последовательно во времени над их отдельными разрядами. Такие АЛУ, как правило, используют конвейерный метод обработки, при котором совмещаются во времени фазы выполнения операции для различных разрядов операндов.

По выполняемым функциям АЛУ делятся на многофункциональные и функциональные (блочные). В блочном АЛУ операции над числами с фиксированной и плавающей запятой, десятичными и алфавитно-цифровыми полями, операции типа "умножение" выполняются в отдельных блоках. Такой подход позволяет увеличить скорость работы АЛУ за счет использования быстродействующих блоков, а также за счет организации параллельной работы этих блоков. Однако в этом случае значительно возрастают затраты оборудования.

В многофункциональных АЛУ всевозможные операции для всех форм представления чисел выполняются одними и теми же схемами, которые коммутируются нужным образом в зависимости от требуемого режима работы.

По структурной организации АЛУ можно разделить на устройства,
имеющие:

- регистровую структуру с непосредственными связями и закрепленной
логикой;
- магистральную структуру с сосредоточенной памятью и логикой.

Арифметико-логические устройства первого типа базируются на принципе закрепления логических схем, используемых для выполнения микроопераций, за каждым из регистров. Так, на рис. 3.15 регистры Р1 и Р2 выполняют функции приема, хранения и выдачи операндов, поступающих из регистров общего назначения (РОН) процессора или КЭШ-памяти данных. С регистром Р1 непосредственно связан преобразователь кода ПК1. Комбинационный сумматор КСМ объединен с регистром РЗ по схеме накапливающего сумматора, с которым непосредственно связаны ПК2 и комбинационная схема КС для мультиплексирования входных данных. На регистре РЗ выполняются микрооперации сдвига вправо или влево и сброс. Регистр Р4 выполняет микрооперации сдвига и непосредственно связан с преобразователем кода ПКЗ.

Таким образом, в данной структуре функции хранения и преобразования информации выполняются одним и тем же операционным блоком.

Магистральная структура АЛУ отличается тем, что в ней регистры и схемы для преобразования информации выделены в отдельные блоки, связанные между собой по входам и выходам. В этом случае блок регистров (БР) выполняет функции приема, хранения, выдачи операндов и результатов, а операционный блок (ОБ) выполняет весь необходимый набор микроопераций над словами, хранимыми в БР. В данной структуре блок регистров может быть реализован двумя способами: либо как совокупность отдельных регистров с индивидуальными схемами управления, либо как сверхоперативное адресное запоминающее устройство.

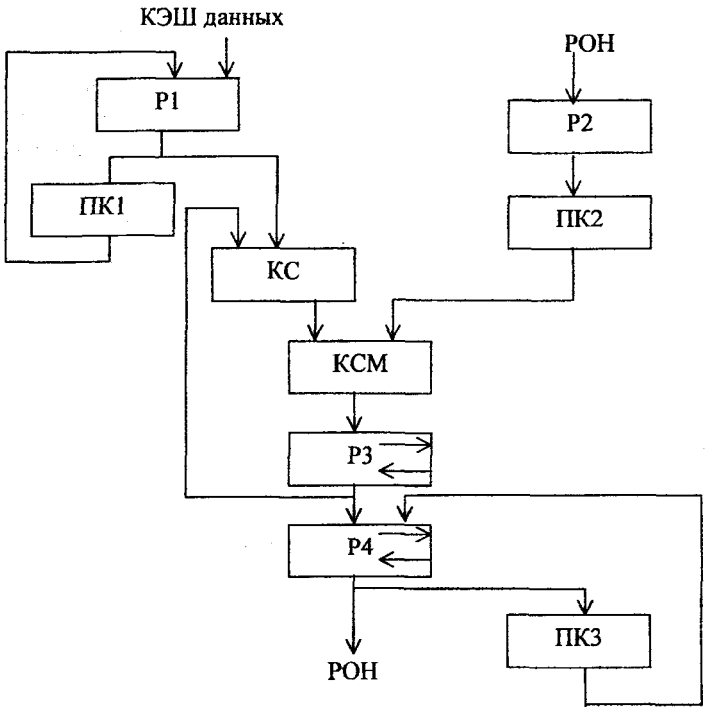


Рис.3.15. Регистровая структура с закрепленной логикой

Структура операционного блока имеет следующие модификации:

- последовательное соединение операционных узлов;
- параллельное соединение операционных узлов.

В первом случае (рис. 3.16) преобразователь кода ПК, комбинационный сумматор КСМ и сдвигатель СДВ соединены последовательно, причем входы ПК и КСМ связаны с выходными шинами блока регистров, а выход СДВ - с входной шиной БР. Такая организация операционного блока дает возможность выполнять с высокой скоростью последовательности микроопераций, обеспечивающей вычисление одного слова.

Во втором случае (рис. 3.17) операционные узлы (СМ, СДВ, ПК, КС) подсоединяются к входным и выходным шинам БР параллельно, что позволяет выполнять несколько микроопераций одновременно.

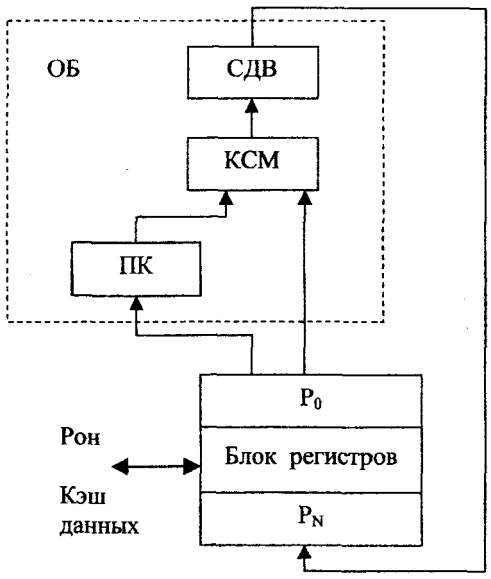
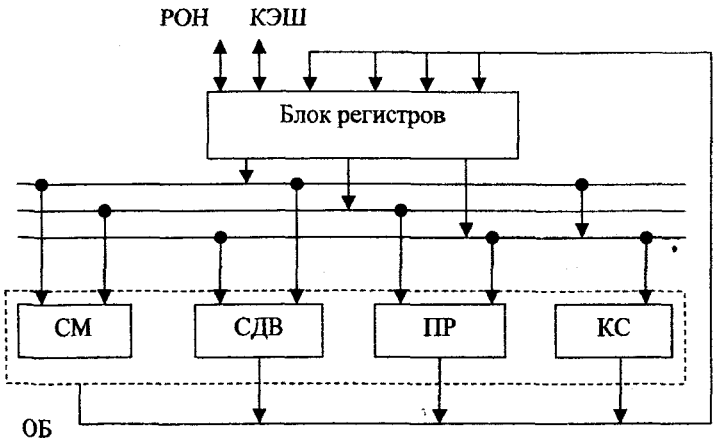


Рис. 3.16. Магистральная структура с последовательным соединением операционных узлов



Рнс.3.17. Магистральная структура с параллельным соединением операционных узлов

Обобщенная структурная схема АЛУ

Обобщенная структурная схема АЛУ (рис. 3.18) включает:

- блок регистров для приема и размещения операндов и результатов;
- операционный блок, в котором осуществляется преобразование операндов в соответствии с реализуемыми алгоритмами;
- схемы контроля, обеспечивающие непрерывный оперативный контроль и диагностирование ошибок;
- блок управления (БУ), в котором после приема кода операции (КОП) из центрального устройства управления формируются управляющие сигналы (УС), координирующие взаимодействие всех узлов АЛУ между собой и с другими блоками процессора.

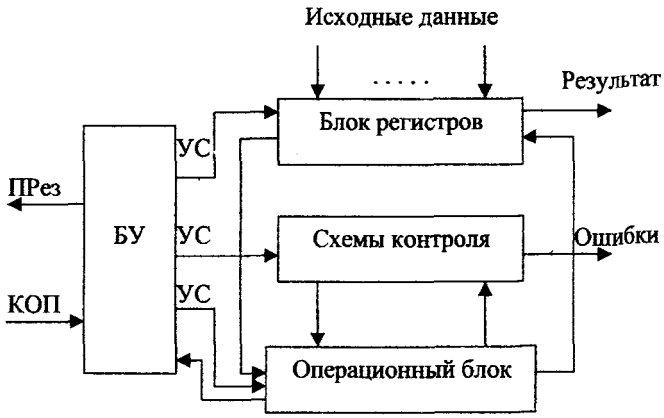


Рис.3.18. Обобщенная схема АЛУ

Блок регистров связан с ГОН центрального процессора и кэш-памятью данных.

Иногда АЛУ не содержит своего БР, в этом случае операционный блок непосредственно работает с регистрами общего назначения процессора. Для оперативного управления выполнением операции в ОБ на разных этапах анализируется преобразуемая информация и формируются сигналы признаков (флаги), которые используются в БУ для выработки и посылки в процессор сигнала признака результата (ПРез).

Для оценки АЛУ используются следующие характеристики: множество выполняемых операций, разрядность, время выполнения операций, надежностные и энергетические характеристики.