Экзаменационные вопросы по курсу «Микропроцессоры и микрокомпьютеры»

1. **История создания ЭВМ. Базовые операции ЦП на основе ТТЛ.**

Аналоговый компьютер – аналог вычислительной машины, которая представляет числовые данные с помощью аналоговых физических переменных (пример: скорость, длина, давление). Самая мощная машина – человеческий мозг.

Первый приём для подсчёта – пальцы. Абака – греческий предмет для подсчёта ( до него были палочки). Илана – предмет для подсчёта у инков. Юуан – китайские счёты, саробан – японские счёты и так далее.

Первая перфокарта была использована на ткацком станке, с её помощью можно было выполнять узоры. Арифмометры – механизмы из шестерёнок для умножения, сложения. Некоторые механизмы доходили до нескольких тонн.

От механизмов перейдём к электричеству. Сначала были аналоговые машины, они использовались в военных целях. Благодаря этому, были разработаны два принципа архитектуры: Неймановская и Гарвардская.

(Сеймур Крей создавал машины, которые сейчас именуются суперкомпьютерами).

Во времена второй мировой Алан Тьюринг разработал машину, с помощью которой можно было расшифровывать коды.

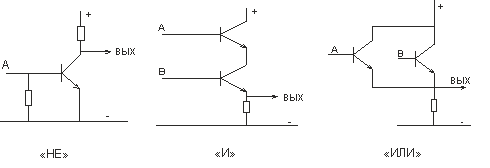
После войны, в 50х, были созданы транзисторы. От транзисто-ламповых – перешли к транзисторным машинам

В 1971 году компания Intel разработала микропроцессор 404: кремниевая пластина с тонким срезом, обработка – создание дорожек (транзистор мог выполнять такие операции, как И, ИЛИ и т.д.). 84 микропроцессор уже использовался для пк. Последняя частота процессора составляет примерно 2,2 ГГц.

Транзисторно-транзисторная логика (ТТЛ, TTL) — разновидность цифровых логических микросхем, построенных на основе биполярных транзисторов и резисторов. Название транзисторно-транзисторный возникло из-за того, что транзисторы используются как для выполнения логических функций (например, И, ИЛИ), так и для усиления выходного сигнала (в отличие от резисторно-транзисторной и диодно-транзисторной логики).

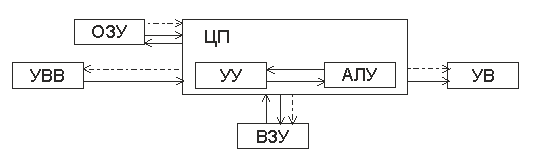
Простейший базовый элемент ТТЛ выполняет логическую операцию И-НЕ, в принципе повторяет структуру ДТЛ микросхем и в то же время за счёт использования многоэмиттерного транзистора, объединяет свойства диода и транзисторного усилителя что позволяет увеличить быстродействие, снизить потребляемую мощность и усовершенствовать технологию изготовления микросхемы.

Базовые операции:



1. **Структурная схема цифровой ЭВМ. Принцип программного управления.**

Простейшая схема:



УВВ/УВ – устройство ввода/вывода;

ОЗУ – основное запоминающее устройство;

УУ – устройство управления;

АЛУ – арифметико-логическое устройство;

ВЗУ – внешнее запоминающее устройство;

ЦП – центральный процессор.

Принцип программного управления (архитектура фон Неймана): Любая информация кодируется в цифровой форме и разделяется на единицы информации – слова. Разнотипные слова информации разделяются по способу применения, а не по способу кодирования, все кодируется с помощью 0 и 1. Слова размещаются в ячейках памяти ЭВМ, каждая ячейка имеет свой номер, который и является адресом данного слова. Алгоритм решения задачи представляется в форме управляющих слов, они определяют наименования операций, данные и так далее. Необходимость определять команду, данные и вычисления, сводятся к последовательным вычислениям команд (для переходов, циклов существуют специальные команды – условного и безусловного переходов).

1. **Представление информации в ЭВМ. Обработка данных посредством цепочечных команд.**

ASCII (англ. American Standard Code for Information Interchange) —стандартная кодировочная таблица для печатных символов и некоторых специальных кодов. ASCII представляет собой кодировку для представления десятичных цифр, латинского и национального алфавитов, знаков препинания и управляющих символов.

Юнико́д (англ. Unicode) — стандарт кодирования символов, позволяющий представить знаки почти всех письменных языков. Стандарт предложен в 1991 году некоммерческой организацией «Консорциум Юникода» (англ. Unicode Consortium, Unicode Inc.). Применение этого стандарта позволяет закодировать очень большое число символов из разных письменностей: в документах Unicode могут соседствовать китайские иероглифы, математические символы, буквы греческого алфавита, латиницы и кириллицы.

Стандарт состоит из двух основных разделов: универсальный набор символов (англ. UCS, universal character set) и семейство кодировок (англ. UTF, Unicode transformation format). Универсальный набор символов задаёт однозначное соответствие символов кодам — элементам кодового пространства, представляющим неотрицательные целые числа. Семейство кодировок определяет машинное представление последовательности кодов UCS. Коды в стандарте Юникод разделены на несколько областей. Область с кодами от U+0000 до U+007F содержит символы набора ASCII с соответствующими кодами. Далее расположены области знаков различных письменностей, знаки пунктуации и технические символы. Часть кодов зарезервирована для использования в будущем.

Для обработки строковых данных ассемблер имеет пять команд обработки строк:

|  |  |
| --- | --- |
| MOVS [b,w,d] | переслать один байт, одно слово или двойное слово из одной области памяти в другую; |
| LODS [b,w,d] | загрузить из памяти один байт, одно слово или двойное слово в регистр AL или в регистр AX; |
| STOS [b,w,d] | записать содержимое регистра AL или AX в память; |
| CMPS [b,w,d] | сравнить содержимое двух областей памяти, размером один байт, одно слово или двойное слово; |
| SCAS [b,w,d] | сравнить содержимое регистра AL или AX с содержимым памяти. |

Несмотря на то, что цепочечные команды имеют отношение к одиному байту, одному слову или одному двойному слову, префикс REP обеспечивает повторение команды несколько раз. Префикс кодируется непосредственно перед цепочечной командой, например, REP MOVSB. Для использования префикса REP необходимо установить начальное значение в регистре CX. При выполнении цепочечной команды с префиксом REP происходит уменьшение на 1 значения в регистре CX до нуля. Таким образом, можно обрабатывать строки любой длины.

Флаг направления определяет направление повторяющейся операции:

* для направления слева направо неоходимо с помощью команды CLD установить флаг DF в 0;
* для направления справа налево необходимо с помощью команды STD установить флаг DF в 1.

При выполнении команд CMPS и SCAS возможна установка флагов состояния, так чтобы операция могла прекратиться сразу после обнаружения необходимого условия. Ниже приведены модификации префикса REP для этих целей.

|  |  |
| --- | --- |
| REP | Повторять операцию, пока CX не равно 0; |
| REPZ или REPE | повторять операцию, пока флаг ZF показывает "равно или ноль". Прекратить операцию при флаге ZF, указывающему на не равно или не ноль или при CX равном 0; |
| REPNE или REPNZ | Повторять операцию, пока флаг ZF показывает "не равно или не ноль". Прекратить операцию при флаге ZF, указывающему на "равно или нуль" или при CX равным 0. |

1. **Системы счисления. Двоичная, двоично-десятичная и шестнадцатеричная системы счисления.**

Система счисления — символический метод записи чисел, представление чисел с помощью письменных знаков. Системы счисления подразделяются на позиционные, непозиционные и смешанные.

Свойства:

* даёт представления множества чисел (целых и/или вещественных);
* даёт каждому числу уникальное представление (или, по крайней мере, стандартное представление);
* отражает алгебраическую и арифметическую структуру чисел.

Двоичная запись чисел: числа записываются с помощью двух символов (0 и 1). Чтобы не путать, в какой системе счисления записано число, его снабжают указателем справа внизу. Например, число 5 в двоичной 1012. Иногда двоичное число обозначают префиксом 0b, например 0b101.

Преобразование двоичных чисел в десятичные: допустим, дано двоичное число 1100012. Для перевода в десятичное запишите его как сумму по разрядам следующим образом: 1 \* 25 + 1 \* 24 + 0 \* 23 + 0 \* 22 + 0 \* 21 + 1 \* 20 = 49.

Двоично-десятичный код (англ. binary-coded decimal), BCD — форма записи целых чисел, когда каждый десятичный разряд числа записывается в виде его четырёхбитного двоичного кода. Например, десятичное число 31110 будет записано в двоичной системе счисления в двоичном коде как 1 0011 01112, а в двоично-десятичном коде как 0011 0001 0001BCD.

Шестнадцатеричная система счисления (шестнадцатеричные числа) — позиционная система счисления по целочисленному основанию 16. Обычно в качестве шестнадцатеричных цифр используются десятичные цифры от 0 до 9 и латинские буквы от A до F для обозначения цифр от 1010 до 1510, то есть (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

Перевод чисел из шестнадцатеричной системы в десятичную: необходимо это число представить в виде суммы произведений степеней основания шестнадцатеричной системы счисления на соответствующие цифры в разрядах шестнадцатеричного числа. Например, требуется перевести шестнадцатеричное число 5A3 в десятичное. В этом числе 3 цифры. В соответствии с вышеуказанным правилом представим его в виде суммы степеней с основанием 16:

5A316 = 3·160+10·161+5·162= 3·1+10·16+5·256 = 3+160+1280 = 144310.

1. **Формы представления и способы кодирования двоичных чисел.**

Система счисления — символический метод записи чисел, представление чисел с помощью письменных знаков. Системы счисления подразделяются на позиционные, непозиционные и смешанные.

Для представления числа в ЭВМ используется стандартная разрядная сетка, состоящая из определённого количества разрядов. Знак кодируется старшим разрядом. В зависимости от способа использования разрядной сетки, различают 2 формы: естественная (последовательность цифр: 1,2… положение точки фиксировано) и нормальная (числа с помощью мантиссы, порядка).

Числа с плавающей точкой описываются следующей формулой:

N = ±M \* n^(±p), где

N — записываемое число;

M — мантисса;

n — основание системы счисления, целое число;

p — порядок числа, показывающий истинное положение точки в разрядах мантиссы, целое;

n^p — характеристика числа.

Способы кодирования двоичных чисел:

Прямой код является простейшим машинным кодом и получается при кодировании в числе только знаковой информации. Прямой код числа совпадает с его изображением в естественной форме, за исключением отрицательных чисел, у которых в знаковом разряде ставится единица. Недостатком кодирования чисел в прямом коде является то, что правила счета, оказываются разными для положительных и отрицательных чисел.

Дополнительный код находится как дополнение модуля отрицательного числа до некоторого граничного числа, представимого в данной ЭВМ. Изображение положительных чисел в дополнительном коде совпадает с изображением их в прямом коде. Дополнительный код для отрицательных чисел больших единицы образуется в соответствии с формулой Aдоп=2n+1+A, а для чисел меньших единицы A=2+A, где n – количество разрядов целой части числа. Таким образом, для представления отрицательного числа в дополнительном коде необходимо в знаковом разряде поставить единицу, произвести обращение всех разрядов числа и к полученному коду прибавить единицу младшего разряда.

Обратный код является инверсией числа и образуется путем замены его

цифр взаимно обратными.

Наибольшее распространение получило кодирование десятичных цифр кодом 8–4–2–1 (BCD), при котором четырьмя разрядами двоичного кода отдельно кодируется каждая цифра в десятичном числе. При этом один байт может хранить либо две цифры десятичного числа, либо одну.

1. **Неупакованные BCD-числа. Корректировки арифметических операций для неупакованных BCD-чисел.**

Неупакованный двоично-десятичный тип – двоичное представление десятичных цифр от 0 до 9. Числа хранятся как байтовые значения без знака по одной цифре в каждом байте (в младшей тетраде);

|  |  |  |
| --- | --- | --- |
| AAA | ASCII Adjust for Addition - коррекция для сложения ASCII-кода | Если младший полубайт регистра AL>9 или AF=1, то: (AL)=(AL)+6; (АН)=(АН)+1; AF= CF=1.  Иначе: AF=CF=0.  В обоих случаях значение старшего полубайта регистра AL обнуляется. |
| AAD | ASCII Adjust for Division - коррекция для деления ASCII-кода | Подготовка двухзначного числа в регистре АХ для операции деления:  (AL)=(АНх10)+AL; (АН)=0. |
| AAM | ASCII Adjust for Multiplication - коррекция для умножения ASCII-кода | Разделить содержимое регистра AL на 10; частное записать в регистр АН, остаток — в регистр AL. |
| AAS | ASCII Adjust for Subtraction - коррекция для вычитания ASCII-кода | Если младший полубайт регистра AL>9 или AF=1, то: (AL)=(AL)-6; (АН) = (АН)-1; AF= CF=1.  Иначе: AF=CF=0.  В обоих случаях значение старшего полубайта регистра AL обнуляется. |

Эти команды кодируются без операндов и выполняют автоматическую коррекцию в регистре AX. Коррекция необходима, так как ASCII код представляет так называемый распакованный десятичный формат, в то время, как компьютер выполняет арифметические операции в двоичном формате.

1. **Упакованные BCD-числа. Корректировки арифметических операций для упакованных BCD-чисел.**

Упакованный двоично-десятичный тип – двоичное представление двух десятичных цифр от 0 до 9, упакованных в один байт. Каждая цифра хранится в своей тетраде (старшая – в старшей, младшая – в младшей);

|  |  |  |
| --- | --- | --- |
| DAA | Decimal Adjustment for Addition - десятичная коррекция для сложения | 1. Если AF=1 или значение младшей тетрады AL>9, то: (AL)=(AL)+6; AF=1; при возникновении  переноса при сложении установить флаг CF=1, иначе — AF=0.  2. Если CF=1 или значение старшей тетрады AL>9, то: (AL)=(AL)+60h; CF=1. Иначе — CF=0. |
| DAS | Decimal Adjustment for Subtraction - десятичная коррекция для вычитания | 1. Если AF=1 или значение младшей тетрады AL>9, то: (AL)=(AL)-6; AF=1; в случае заема при  вычитании установить флаг CF=1. Иначе — AF=0.  2. Если CF=1 или значение старшей тетрады AL>9, то: (AL)=(AL)-60h; CF=1. Иначе — CF=0. |

1. **Сегментная организация памяти в реальном режиме. Модели памяти.**

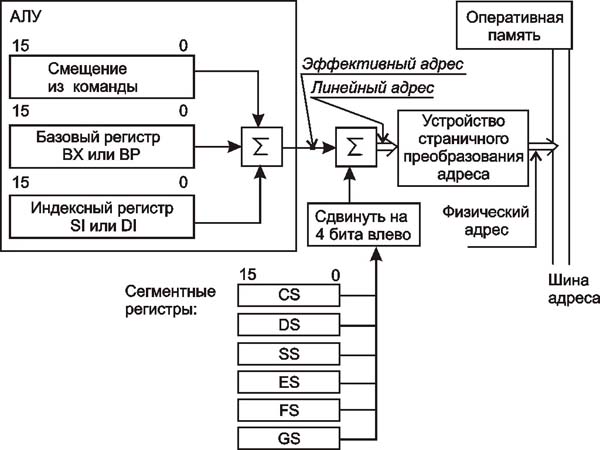
Рассмотрим конкретную реализацию сегментированной памяти на примере процессора 8086, которая очень проста и доступна для понимания.

Под сегментом понимается блок смежных ячеек памяти в адресном пространстве 1 Мбайт. Максимальный размер блока 64 Кбайт, начальный или базовый адрес находится на 16-байтной границе (такая граница называется параграфом). Для обращения к нужной ячейке памяти надо задать базу сегмента и 16-битное расстояние от базы называемое смещением или относительным адресом.

Модели памяти

|  |  |
| --- | --- |
| TINY | Код, данные и стек размещаются в одном и том же сегменте размером 64Кбайт |
| SMALL | Код – в одном сегменте, а данные и стек – в другом |
| COMPACT | Код – в одном сегменте, а данные могут располагаться в нескольких сегментах. Используются дальние указатели |
| MEDIUM | Код – в нескольких сегментах, а все данные – в одном. Для доступа к данным используется только смещение, а для вызова подпрограмм – команды дальнего вызова процедур |
| LARGE, HUGE | И код, и данные могут занимать несколько сегментов |
| FLAT | То же, что и TINY, но используются 32-разрядные адреса. Максимальный размер сегмента – 4 Гбайт |

1. **Формирование физического адреса в реальном режиме. Недостатки реального режима.**



Характеристики механизма адресации физической памяти в реальном режиме.

* Диапазон изменения физического адреса — от 0 до 1 Мбайт. Эта величина определяется тем, что шина адреса i8086 имела 20 линий.
* Максимальный размер сегмента— 64 Кбайт. Это объясняется 16-разрядной архитектурой 18086. Нетрудно подсчитать, что максимальное значение, которое могут содержать 16-разрядные регистры, составляет 216 - 1, что применительно к памяти и определяет величину 64 Кбайт.
* Для обращения к конкретному физическому адресу оперативной памяти необходимо определить адрес начала сегмента (сегментную составляющую) и смещение внутри сегмента.

Недостатки сегментной организации памяти в реальном режиме:

* Сегменты могут перекрываться друг другом.
* Пользователь может обратиться по любому адресу, в том числе и по несуществующему.
* Ограниченный размер сегмента.

1. **Микропроцессор IA-32. Регистры общего назначения**

Регистры общего назначения (РОН) без каких-либо ограничений могут использоваться для хранения операндов логических и арифметических операций, компонентов адреса, указателей на ячейки памяти, однако, каждый из них имеет определенное функциональное назначение:

* EAX/AX/AH/AL (Accumulator register) – аккумулятор – применяется для хранения промежуточных данных, а также адресов;
* EBX/BX/BH/BL (Base register) – базовый регистр – применяется для хранения промежуточных данных, а также для хранения базового адреса объектов в памяти;
* ECX/CX/CH/CL (Count register) – регистр-счетчик – применяется для хранения промежуточных данных, а также в циклических командах, производящих некоторые повторяющиеся действия, в качестве счетчика итераций;
* EDX/DX/DH/DL (Data register) – регистр данных – так же, как и регистр EAX/AX/AH/AL, он используется для хранения промежуточных данных.

Следующие два регистра используются для поддержки цепочечных команд:

* ESI/SI (Source Index register) – индекс источника – содержит адрес текущего элемента в цепочке-источнике;
* EDI/DI (Destination Index register) – индекс приемника – содержит адрес текущего элемента в цепочке-приемнике.

Следующие регистры предназначены для работы со стеком:

* ESP/SP (Stack Pointer register) – регистр указателя стека – содержит указатель (адрес) на вершину стека в сегменте стека. Содержимое регистра неявно изменяется с помощью команд работы со стеком. Использование этого регистра для хранения каких-либо операндов программы недопустимо;
* EBP/BP (Base Pointer register) – регистр указателя базы кадра стека – предназначен для организации произвольного доступа к данным внутри стека. Например, при организации доступа к параметрам, передаваемым в процедуры через стек..

1. **Микропроцессор IA-32. Сегментные регистры**

Каждый сегментный регистр обеспечивает адресацию 64КБайт памяти, которая называется текущим сегментом. Сегмент выравнен на границу параграфа и его адрес в сегментном pегистре предполагает наличие справа четырех нулевых битов. Сегментные регистры предназначены для аппаратной поддержки структурной организации программы в виде отдельных частей, называемых сегментами. Операционная система размещает сегменты программы в оперативной памяти, после чего помещает их адреса в соответствующие сегментные регистры.

* сегмент кода – содержит машинные команды, для доступа к нему служит регистр CS (Code Segment register) – сегментный регистр кода;
* сегмент стека – для доступа к нему служит регистр SS (Stack Segment register) – сегментный регистр стека;
* сегмент данных – содержит обрабатываемые программой данные, для доступа к нему служит регистр DS (Data Segment register) – сегментный регистр данных;
* дополнительные сегменты данных – их адреса должны содержаться в регистрах ES, GS, FS.

1. **Микропроцессор IA-32. Указатель команд, регистр флагов и управляющие регистры**

Регистры состояния и управления содержат информацию о состоянии МП и программы, команды которой выполняются в настоящий момент.

С помощью регистра флагов EFLAGS/FLAGS можно получать информацию о результатах выполнения команд и влиять на состояние МП. Флаг, как правило, представляет собой один бит информации. Говорят, что флаг «установлен», если он равен 1, и «сброшен» если равен 0. С помощью регистра флагов EFLAGS/FLAGS можно получать информацию о результатах выполнения команд и влиять на состояние МП. Флаг, как правило, представляет собой один бит информации. Говорят, что флаг «установлен», если он равен 1, и «сброшен» если равен 0.

Регистр указатель команд EIP/IP содержит адрес (смещение) следующей исполняемой команды относительно начала сегмента кода, адрес которого содержится в регистре CS. Регистр EIP/IP непосредственно недоступен программисту, но загрузка и изменение его значения может производиться командами безусловного и условных переходов, вызова процедур и прерываний, а также возврата из них..

1. **Микропроцессор IA-32. Организация памяти.**

Сегментная адресация памяти — схема логической адресации памяти компьютера в архитектуре x86. Линейный адрес конкретной ячейки памяти, который в некоторых режимах работы процессора будет совпадать с физическим адресом, делится на две части: сегмент и смещение. Сегментом называется условно выделенная область адресного пространства определённого размера, а смещением — адрес ячейки памяти относительно начала сегмента. Базой сегмента называется линейный адрес (адрес относительно всего объёма памяти), который указывает на начало сегмента в адресном пространстве. В результате получается сегментный (логический) адрес, который соответствует линейному адресу база сегмента+смещение и который выставляется процессором на шину адреса.

Селектором называется число (в x86 — 16-битное), однозначно определяющее сегмент. Селектор загружается в сегментные регистры.

В реальном и защищённом режимах x86-процессора функционирование сегментной адресации отличается.

1. **Микропроцессор IA-32. Модель памяти в реальном режиме.**

Реальный режим (real mode): Классический режим адресации, использованный в первых моделях семейства. Использует сегментированную модель памяти, организованную следующим образом: адресное пространство в 1 MiB разбивается на 16-байтовые блоки, называемые параграфами. Всего параграфов в 1 MiB — 65536 (216), что позволяет пронумеровать их 16-разрядными числами. Сегменты памяти имеют размер 65536 байт (64Кбайт), и всегда начинаются на границе параграфа. Адрес ячейки памяти состоит из двух частей: номера параграфа, с которого начинается сегмент и смещения внутри сегмента и обычно записывается как SSSS:OOOO (Segment и Offset), где S и O — шестнадцатеричные цифры. SSSS называется сегментной компонентой адреса, а OOOO — смещением. Адрес ячейки, выдаваемый на шину, представляет собой сегментную компоненту умноженную на 16 плюс смещение. Сегментная компонента помещается в специальный регистр, называемый сегментным, а смещение в регистр IP(регистр инструкций).

Несмотря на то, что сегментные регистры имеют специальные назначения, архитектура допускает при обращении к данным заменить один сегмент на любой другой. Сегменты кода, стека и получателя строк всегда используют регистры CS, SS и ES и не могут быть изменены.

Общий объём памяти, адресуемый в реальном режиме составляет 1048576 байт (220 = 216 \* 24 – двоичная система; 0000:0000 – FFFFF:FFFFF - логический адрес (физический адрес) в шестнадцатеричной системе счисления). Сегментный подход позволяет разделить всю память на 16 сегментов, начинающихся с адресов, кратных 64 Кбайт. Эти 16 сегментов называют страницами памяти. Обычно деление на страницы используется для совместного функционирования устройств, интерфейсы которых отображены на адресное пространство памяти; тогда каждое такое устройство использует одну страницу памяти, и адрес ячейки в адресном пространстве устройства будет совпадать со смещением в сегменте памяти компьютера. Реально доступная пользователю память составляет 640 Кбайт (первые 10 страниц). Также в реальном режиме отсутствует защита памяти и разграничение прав доступа.

1. **Микропроцессор IA-32. Режимы адресации.**

Прямая адресация – простейший вид адресации операнда в памяти, может быть двух типов:

* относительная – используется в командах условного и безусловного перехода.
* абсолютная – использует несколько форм такой адресации.

Косвенная базовая (регистровая) адресация – адрес операнда находится в любом из регистров общего назначения, кроме SP/ESP и BP/EBP. Синтаксически в команде этот режим адресации выражается заключением имени регистра в квадратные скобки [].

Косвенная базовая (регистровая) адресация со смещением: предназначена для доступа к данным с известным смещением относительно некоторого базового адреса.

Косвенная индексная адресация со смещением позволяет масштабировать содержимое индексного регистра.

Косвенная базово-индексная: эффективный адрес формируется как сумма содержимого двух любых регистров общего назначения: базового и индексного.

Косвенная базово-индексная адресация со смещением: эффективный адрес формируется как сумма трех составляющих: содержимого базового регистра, содержимого индексного регистра и значения смещения.

Косвенная базово-индексная адресация со смещением и масштабированием объединяет в себе все предыдущие режимы косвенной адресации.

1. **Типы данных микропроцессора.**

С точки зрения размерности, МП аппаратно поддерживает следующие основные типы данных:

* байт – восемь последовательно расположенных битов;
* слово – два байта, имеющих последовательные адреса. Слово делится на младший байт и старший. Младший байт всегда хранится по меньшему адресу, который является адресом слова;
* двойное слово – четыре байта, расположенных по последовательным адресам. Двойное слово состоит из младшего слова и старшего. Младшее слово хранится по меньшему адресу, который является адресом двойного слова;
* учетверенное слово – восемь байт, расположенных по последовательным адресам. Учетверенное слово делится на младшее двойное слово и старшее двойное слово. Младшее двойное слово хранится по меньшему адресу, который является адресом учетверенного слова;
* 128-разрядный упакованный тип данных – появился в МП Pentium III. Для работы с ним были введены специальные команды.

1. **Система команд микропроцессора.**

Система команд (также набор команд) — соглашение о предоставляемых архитектурой средствах программирования, а именно: определённых типах данных, инструкций, системы регистров, методов адресации, моделей памяти, способов обработки прерываний и исключений, методов ввода и вывода.

Основными классы:

* Инструкции пересылки данных: позволяют передавать константы или переменные между регистрами и памятью, а также портами ввода/вывода в различных комбинациях, но в памяти может находиться не более одного операнда.
* Инструкции двоичной арифметики: выполняют все арифметические действия с байтами, словами и двойными словами, кодирующими знаковые или беззнаковые целые числа.
* Инструкции десятичной арифметики: позволяют оперировать с неупакованными или упакованными двоично–десятичными числами. Арифметические операции над этими числами требуют применения инструкций коррекции форматов.
* Инструкции логических операций: выполняют все функции булевой алгебры над байтами, словами или двойными словами.
* Инструкции сдвигов: сдвиги и вращения (циклические сдвиги) выполняются над регистром или операндом в памяти.
* Инструкции обработки бит и байт: позволяют проверять (копировать в CF) и устанавливать значение указанного операнда, а также искать установленный бит.
* Инструкции передачи управления: передача управления осуществляется с помощью инструкций безусловных и условных переходов, вызовов процедур и прерываний (исключений).
* Инструкции строковых операций: строковые операции выполняются с операндами в памяти, адресуемыми регистрами DS:SI (DS:ESI) для источника и ES:DI (ES:EDI) для приемника. Операции могут использоваться с префиксами условного или безусловного повтора.
* Инструкции работы с флагами: операции с флагами позволяют изменять значения отдельных флагов, а также сохранять их значения в стеке (или регистре AH) и восстанавливать сохраненные значения.
* Инструкции загрузки указателей: позволяют загружать дальние указатели из памяти в регистр общего назначения и соответствующий сегментный регистр.
* Разные инструкции: инструкции не вошедшие ни в один из перечисленных классов.

1. **Программная модель математического сопроцессора. Регистры**

Арифметический сопроцессор (FPU) предназначен для выполнения операций над числами в формате с плавающей точкой (вещественные числа) и длинными целыми числами. Он значительно (в десятки раз) ускоряет вычисления, связанные с вещественными числами. Сопроцессор может выполнять как простые арифметические операции (сложение, вычитание, умножение, деление и т.п.), так и вычислять значения различных функций (синус, косинус, тангенс, логарифмы и т. п.).

В программной модели сопроцессора можно выделить три группы регистров:

* восемь регистров R0 – R7, составляющих основу программной модели сопроцессора – стек сопроцессора;
* три служебных регистра: регистр состояния сопроцессора SWR, управляющий регистр сопроцессора CWR, регистр слова тегов TWR;
* два регистра указателей: данных – DPR и команд – IPR.



Регистры стека сопроцессора: размерность каждого из регистров R0-R7 80 битов. Реализация численных алгоритмов на основе регистрового стека позволяет получить существенный выигрыш в скорости вычислений.

Регистровый стек сопроцессора организован по принципу кольца. Это означает, что все регистры стека с функциональной точки зрения абсолютно одинаковы и равноправны. Как и в любом стеке, в стеке сопроцессора есть вершина, которая является «плавающей». Контроль текущей вершины осуществляется аппаратно с помощью трехбитового поля TOP регистра SWR. В этом поле фиксируется физический номер (0...7) регистра стека, который в данный момент является текущей вершиной стека. Команды сопроцессора не оперируют физическими номерами регистров стека R0...R7. Вместо этого они используют логические номера этих регистров ST(0)...ST(7). C помощью логических номеров реализуется относительная адресация регистров стека сопроцессора. По мере записи в стек, указатель его вершины движется по направлению к младшим номерам физических регистров (top = top + 1). Если текущей вершиной является R0, то после записи очередного значения в стек сопроцессора его текущей вершиной, например, станет физический регистр R2. Что касается логических номеров регистров стека ST(0)...ST(7), то, они «плавают» вместе с изменением текущей вершины стека. Таким образом, реализуется принцип кольца.

Регистр управления CWR: регистр управления работой сопроцессора определяет особенности обработки численных данных и состоит из:

* шести масок исключений – предназначены для маскирования исключительных ситуаций, возникновение которых фиксируется с помощью шести бит регистра SWR. Если какие-то биты исключений в регистре CWR установлены в единицу, то это означает, что соответствующие исключения будут обрабатываться самим сопроцессором. Если для какого-либо исключения в соответствующем бите масок исключений регистра CWR содержится нулевое значение, то при возникновении исключения этого типа будет возбуждено прерывание 16 (10h));
* поле управления точностью РС (Precision Control) – предназначено для выбора длины мантиссы. Возможные значения в этом поле означают следующую длину мантиссы: 00 – 24 бита; 10 – 53 бита; 11 – 64 бита (значение по умолчанию);
* поле управления округлением RC (Rounding Control) – позволяет управлять процессом округления чисел в процессе работы сопроцессора. Возможные значения поля RС: 00 – значение числа округляется к ближайшему числу в большую или меньшую сторону; 01 – значение числа округляется в меньшую сторону; 10 – значение числа округляется в большую сторону; 11 – производится отбрасывание дробной части.

Регистр SWR отражает текущее состояние сопроцессора после выполнения последней команды и состоит из:

* 6 флагов исключительных ситуаций: IE (Invalide operation Error) – недействительная операция; DE (Denormalized operand Error) – денормализованный операнд; ZE (divide by Zero Error) – ошибка деления на ноль; ОЕ (Overflow Error) – ошибка переполнения. Возникает в случае выхода порядка числа за максимально допустимый диапазон; UE (Underflow Error) – ошибка антипереполнения. Возникает, когда результат слишком мал; РЕ (Precision Error) – ошибка точности;
* бит SF (Stack Fault) – ошибка работы стека сопроцессора. Бит устанавливается в единицу, если возникает одна из трех исключительных ситуаций РЕ, IE или UE;
* бит ES (Error Summary) – суммарная ошибка работы сопроцессора. Бит устанавливается в единицу, если возникает любая из шести исключительных ситуаций;
* четырех битов С0-C3(Condition Code) – кода условия. Отражают результат выполнения последней команды сопроцессора.

Регистр TWR: регистр тегов представляет собой совокупность двухбитовых полей. Каждое поле соответствует определенному физическому регистру стека и характеризует его текущее состояние. Изменение состояния любого из регистров стека сопроцессора отражается на содержимом соответствующего этому регистру поля регистра тега. Возможны следующие значения в полях регистра тега: 00 – регистр стека сопроцессора занят допустимым ненулевым значением; 01 – регистр стека сопроцессора содержит нулевое значение; 10 – регистр стека сопроцессора содержит одно из специальных численных значений, за исключением нуля; 11 – регистр пуст и в него можно производить запись.

Регистры указатели DPR и IPR: Два регистра указателей – данных DPR (Data Point Register) и команд IPR (Instruction Point Register) предназначены для запоминания информации об адресе команды, вызвавшей исключительную ситуацию и адресе ее операнда. Эти указатели используются при обработке исключительных ситуаций (но не для всех команд).

1. **Типы данных математического сопроцессора.**

Сопроцессор работает с обычными типами данных и со специальными численными значениями.

Обычные типы данных и их размерность:

|  |  |
| --- | --- |
| **Целые числа:** |  |
| Целое слово | 16 |
| Короткое слово | 32 |
| Длинное целое | 64 |
| Целые двоично-десятичные числа | 80 |
| **Вещественные числа:** |  |
| Короткий формат | 32 |
| Длинный формат | 64 |
| Расширенный формат | 80 |

Вещественные числа формируются с помощью формулы:

N = ±M \* n^(±p), где

N — записываемое число;

M — мантисса (должна удовлетворять условию: |M|<1);

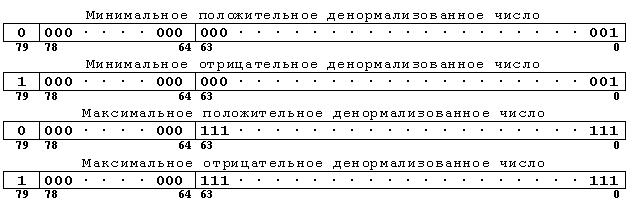
n — основание системы счисления, целое число;

p — порядок числа, показывающий истинное положение точки в разрядах мантиссы, целое;

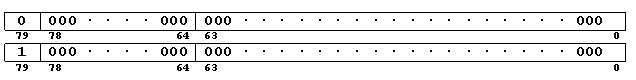
n^p — характеристика числа.

Специальные численные значения:

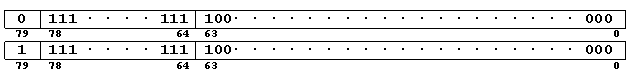
* Денормализованные вещественные числа – это числа, которые меньше минимального положительного нормализованного числа для каждого вещественного формата:



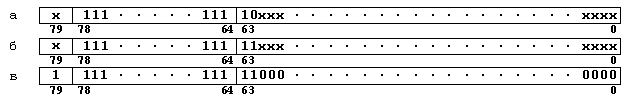
* ±Ноль



* ±Бесконечность



* Нечисла:
* SNAN (Signaling Non a Number) – сигнальные нечисла (а) ;
* QNAN (Quiet Non A Number) – спокойные (тихие) нечисла (б);
* Вещественная неопределенность (в).



1. **Система команд математического сопроцессора.**

Команды передачи данных: осуществляют обмен между регистрами стека, вершиной стека сопроцессора и ячейками оперативной памяти:

* fld/fild источник – загрузка вещественного/целого числа из памяти в вершину стека сопроцессора;
* fst/fist приемник – сохранение вещественного/целого числа из вершины стека сопроцессора в память(поле top не изменяется);
* fstp/fistp приемник – сохранение вещественного/целого числа из вершины стека сопроцессора в память (top = top+1);
* fxch st(i) – обмен значений между текущей вершиной стека и регистром стека сопроцессора ST(i).

Команды загрузки констант (все константы загружаются в вершину стека сопроцессора):

* fldz – загрузка нуля;
* fld1 – загрузка единицы;
* fldpi – загрузка числа π;
* fldl2t – загрузка двоичного логарифма десяти (log210);
* fldl2e – загрузка двоичного логарифма е (log2e);
* fldlg2 – загрузка десятичного логарифма двух (lg2);
* fldln2 – загрузка натурального логарифма двух (ln2).

Команды сравнения данных: сравнение значений числа, находящегося в вершине стека и операнда, указанного в команде:

* fcom/ficom [операнд] – команда без операндов сравнивает два значения: одно находится в регистре ST(0), другое – в регистре ST(1). Если указан операнд, то сравнивается значение в регистре ST(0) стека сопроцессора со значением в памяти;
* fcomp/ficomp операнд – команда сравнивает значение в вершине стека сопроцессора ST(0) со значением операнда, который находится в регистре или в памяти;
* ftst – команда не имеет операндов и сравнивает значение в ST(0) со значением 0.

Арифметические команды: реализуются четыре основные арифметические операции – сложение, вычитание, умножение и деление. Имеется также несколько дополнительных команд, предназначенных для повышения эффективности использования основных арифметических команд.

Команды трансцендентных функций: предназначенны для вычисления значений тригонометрических функций, таких, как синус, косинус, тангенс, арктангенс, а так-же значений логарифмических и показательных функций.

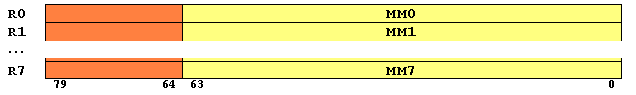
Команды управления сопроцессором:

* wait/fwait – команда ожидания, предназначена для синхронизации работы процессора и сопроцессора;
* finit – команда инициализации сопроцессора (приводит сопроцессор в начальное состояние;
* fstsw приемник – команда сохранения содержимого регистра состояния SWR в ячейке памяти или регистре AX;
* fstcw приемник – команда сохраняет содержимое регистра управления сопроцессора в ячейку памяти, адресуемую операндом приемник;
* fldcw источник – команда выполняет операцию загрузки значения регистра CWR из ячейки памяти, адресуемой операндом источник;
* ffree st(i) – команда освобождения регистра стека ST(i).

1. **Программная модель расширения MMX. Регистры и типы данных**

MMX (Multimedia Extensions — мультимедийные расширения). Технология MMX создана для упрощения и ускорения алгоритмов применяющих однотипные вычисления над несколькими единицами данных. В основном это работа с графикой и звуком.

Технология MMX представлена в виде восьми 64 битных регистров (называемых mm0-mm7) и дополнительном наборе команд для их обработки. Регистры MMX находятся в контексте FPU, но адресуются по номерам, а не относительно вершины стека. Из-за воздействия MMX на контекст FPU эти блоки одновременно не используются.



Основная особенность MMX - это поддержка SIMD (single-instruction, multiple-data) операций, иногда называемых также векторными или упакованными. Большинство команд MMX позволяют одновременно обработать сразу несколько единиц данных. Интерпретация данных регистра MMX зависит от применяемой команды. 64 битный регистр обычно можно воспринимать как:

* одно учетверенное слово (qword);
* два двойных слова (dword);
* четыре слова (word);
* восемь байт (byte).

1. **Система команд MMX.**

Система команд MMX состоит из 57 команд, сгруппированных в следующие категории:

команды пересылки данных:

* movd dst, src – пересылает 32-разрядные данные из памяти в регистры MMX и обратно или из целочисленных регистров процессора в регистры MMX и обратно;
* movq dst,src – пересылает 64-разрядные упакованные данные из памяти в регистры MMX и обратно или между регистрами MMX;

арифметические команды:

* padd[b, w, d, s[b,w], us[b,w]] - сложение элементов данных регистра. возможно насыщение;
* psub[b, w, d, s[b, w], us[b,w]] - вычитание упакованных элементов. Возможно насыщение;
* pmul[h,l]w - знаковое умножение упакованных слов. Возврат старших/младших слов результата;
* pmaddwd - умножение/сложение;

команды сравнения:

* pcmpeq[b, w, d] - сравнить элементы данных, если равны установить биты элемента в единицы, если нет то в нули.
* pcmpgt[b, w, d] сравнить элементы данных, если соответствующий элемент данных первого операнда больше соответствующего второго, то установить его в единицы, если нет то в нули.

логические команды:

* pand[n] - операция AND/ AND и XOR над данными регистра;
* por - операция OR;
* pxor - операция XOR;

команды сдвига:

* psll[w, d, q] - логический сдвиг влево;
* psra[w, d] - арифметический сдвиг вправо;
* psrl[w, d, q] - логический сдвиг вправо;

команды упаковки и распаковки:

* pack[ssdw,sswb,uswb] - преобразование (с насыщением) размеров единиц данных из первого операнда в младшее двойное слово результата, из второго - в старшее.
* punpck[h,l][bw,wd,dq] - распаковка старших/младших двойных слов операндов.

команда инициализации:

* emms - подготовка FPU контекста для выполнения FPU операций после MMX;

дополнительные команды.

1. **Защищенный режим работы микропроцессора. Регистры управления МП.**

Основным режимом работы микропроцессора является защищенный режим. Ключевыми особенностями защищенного режима являются: виртуальное адресное пространство, защита и многозадачность.

Виртуальное адресное пространство – адреса, которые могут относиться к физически отсутствующим ячейкам памяти. Размер виртуального адресного пространства может достигать 64Тбайт. Для адресации виртуального адресного пространства используется сегментированная модель, в которой адрес состоит из двух элементов: селектора сегмента и смещения внутри сегмента. С каждым сегментом связана особая структура, хранящая информацию о нем - дескриптор. Кроме "виртуализации" памяти на уровне сегментов существует возможность "виртуализации" памяти при помощи страниц - страничная трансляция. Страничная трансляция предоставляет удобные средства для реализации в операционной системе функций подкачки.

Многозадачность – встроенные средства переключения задач в защищенном режиме. Среда задачи состоит из содержимого регистров МП и всего кода с данными в пространстве памяти. Микропроцессор способен быстро переключаться из одной среды выполнения в другую, имитируя параллельную работу нескольких задач. Для некоторых задач может эмулироваться управление памятью. При этом задачи виртуального МП изолированы и защищены, как от друг друга, так и от обычных задач защищенного режима.

Защита задач обеспечивается следующими средствами:

* контроль пределов сегментов,
* контроль типов сегментов,
* контроль привилегий,
* привилегированные инструкции,
* защита на уровне страниц.

Контроль пределов и типов сегментов обеспечивает целостность сегментов кода и данных. Программа не имеет права обращаться к виртуальной памяти, выходящей за предел того или иного сегмента. Программа не имеет права обращаться к сегменту данных как к коду и наоборот. Архитектура защиты микропроцессора обеспечивает 4 иерархических уровня привилегий (0: ядро ОС; 1: драйвера ОС; 2: интерфейс ОС; 3: прикладные программы), что позволяет ограничить задаче доступ к отдельным сегментам в зависимости от ее текущих привилегий. Кроме того, текущий уровень привилегий задачи влияет на возможность выполнения тех или иных специфических команд (привилегированных инструкций).

Системные регистры микропроцессора:

Эти регистры обеспечивают работу защищенного режима. Их также можно рассматривать как часть архитектуры микропроцессора, которая намеренно оставлена видимой для того, чтобы квалифицированный системный программист мог выполнить самые низкоуровневые операции.

Системные регистры можно разделить на три группы:

Четыре регистра управления: cr0 – cr3 - эти регистры предназначены для общего управления системой (определяют режимы работы микропроцессора), 32-битовые. Регистры управления доступны только программам с уровнем привилегий 0;

* Регистр cr0 содержит системные флаги, управляющие режимами работы мик­ропроцессора и отражающие его состояние глобально, независимо от конкрет­ных выполняющихся задач.
* Регистр cr1 зарезервирован.
* Регистр cr2 используется при страничной организации оперативной памяти для регистрации ситуации, когда текущая команда обратилась по адресу, со­держащемуся в странице памяти, отсутствующей в данный момент времени в памяти.
* Регистр cr3 также используется при страничной организации памяти.

Четыре регистра системных адресов (регистры управления памятью) предназначены для защиты программ и данных в мультизадачном режиме работы микропроцессора. При работе в защищенном режиме микропроцессора адресное пространство делится: на глобальное (общее для всех задач) и локальное (отдельное для каждой задачи). Этим разделением и объясняется то, что в архитектуре микропроцессора присутствуют следующие системные регистры:

* GDTR - регистр таблицы глобальных дескрипторов (содержит базовый адрес глобальной дескрипторной таблицы и предел таблицы);
* LDTR - регистр таблицы локальных дескрипторов (содержит указатель на локальную таблицу сегментов текущей задачи);
* IDTR - регистр таблицы дескрипторов прерываний (содержит базовый адрес дескрипторной таблицы прерываний и предел GDT - размер (в байтах) таблицы);
* TR - регистр задачи (указывает на дескриптор в таблице GDT, позволяющий получить доступ к дескриптору текущей задачи - этот дескриптор описывает текущий сегмент состояния задачи).

Восемь регистров отладки: dr0 – dr7;

* Регистры dr0, dr1, dr2, dr3 предназначены для задания линейных адресов четырех точек прерывания.
* Регистры dr4, dr5 не используются чаще всего.
* Регистр dr6 называется регистром состояния отладки.
* Регистр dr7 называется регистром управления отладкой.

1. **Модель памяти в защищенном режиме.**

Линейная адресация памяти — схема адресации памяти компьютера в защищенном режиме. Используется большинством современных многозадачных ОС.

Благодаря механизму линейной адресации можно создавать любое количество независимых виртуальных адресных пространств (ограниченное только размерами оперативной памяти). Причём каждая страница линейного адресного пространства может находиться по любому физическому адресу или даже быть выгруженной на диск.

При использовании линейной адресации 32-битный логический адрес делится на три части:

* Номер записи в каталоге страниц (номер таблицы страниц);
* Номер записи в таблице страниц (номер страницы в таблице страниц);
* Смещение в рамках страницы.

Для включения линейной адресации необходимо, находясь в защищенном режиме, установить бит PG=1 в регистре CR0. Предварительно необходимо создать в памяти каталог страниц (англ. PageDirectory, PD) и таблицы страниц (англ. PageTable, PT), после чего в регистр CR3 загрузить физический адрес каталога страниц.

1. **Уровни привилегий в защищенном режиме**.

Основой защищённого режима являются уровни привилегий. Уровень привилегий - это степень использования ресурсов процессора. Всего таких уровней четыре и они имеют номера от 0 до 3. Уровень номер 0 - самый привелигерованный. Когда программа работает на этом уровне привилегий, ей "можно всё". Уровень 1 - менее привелигерованный и запреты, установленные на уровне 0 действуют для уровня 1. Уровень 2 - ещё менее привелигерованный, а 3-й - имеет самый низкий приоритет. Таким образом, оптимальная схема работы программ по уровням привилегий будет следующая:

* уровень 0: ядро ОС,
* уровень 1: драйвера ОС,
* уровень 2: интерфейс ОС,
* уровень 3: прикладные программы

1. **Организация защиты кода и данных в защищенном режиме.**

Защита задач обеспечивается следующими средствами:

* контроль пределов сегментов;
* контроль типов сегментов;
* контроль привилегий;
* привилегированные инструкции
* защита на уровне страниц.

Контроль пределов и типов сегментов обеспечивает целостность сегментов кода и данных. Программа не имеет права обращаться к виртуальной памяти, выходящей за предел того или иного сегмента. Программа не имеет права обращаться к сегменту данных как к коду и наоборот. Архитектура защиты микропроцессора обеспечивает 4 иерархических уровня привилегий, что позволяет ограничить задаче доступ к отдельным сегментам в зависимости от ее текущих привилегий. Кроме того, текущий уровень привилегий задачи влияет на возможность выполнения тех или иных специфических команд (привилегированных инструкций). Функции страничной трансляции, впервые появившиеся в МП Intel386, обеспечивают дополнительные механизмы защиты на уровне страниц.

Выведение информации о базовом адресе сегмента и его размере на уровень микропроцессора позволяет аппаратно контролировать работу программ с памятью и предотвращать обращения по несуществующим адресам либо по адресам, находящимся вне предела, разрешенного полем размера сегмента limit.

1. **Переключение МП из реального режима в защищенный и обратно.**

После инициализации процессор находится в реальном режиме. Процессор может быть переведен в защищенный режим установкой бита PE = 1 (Protect Enable) в регистре CR0. Второй вариант "достался в наследство" от 16-разрядной архитектуры 80286, для совместимости с которой ее регистр MSW (Machine Status Word) отображается на младшее слово регистра CR0.

Вернуться в режим реального адреса процессор может по сигналу RESET или сбросив бит PE = 0.

Алгоритм перехода к защищённому режиму:

1. Создать в памяти таблицы GDT и IDT.

2. Запретить внешние прерывания.

3. Перенастроить контроллеры прерываний на новые типы (номера) прерываний.

4. Открыть линию А20 (если она была закрыта).

5. Во внутренние регистры процессора gdtr и idtr загрузить начальные адреса и размеры таблиц GDT и IDT.

6. Установить бит РЕ. С этого момента мы в защищенном режиме.

7. Очистка очереди команд.

8. Разрешить внешние прерывания.

Пункты 3,4 и 7 являются необязательными.

1. **Обработка прерываний в защищенном режиме.**

В защищённом режиме все прерывания разделяются на два типа - обычные прерывания и исключения (exception - исключение, особый случай).

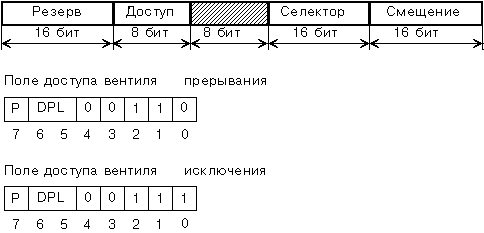
Обычное прерывание инициируется командой INT (программное прерывание) или внешним событием (аппаратное прерывание). Перед передачей управления процедуре обработки обычного прерывания флаг разрешения прерываний IF=0 сбрасывается и прерывания запрещаются.

Исключение происходит в результате ошибки, возникающей при выполнении какой-либо команды, например, если команда пытается выполнить запись данных за пределами сегмента данных, который не определён в таблице дескрипторов. По своим функциям исключения соответствуют зарезервированным для процессора внутренним прерываниям реального режима. Когда процедура обработки исключения получает управление, флаг IF не изменяется. Поэтому в мультизадачной среде особые случаи, возникающие в отдельных задачах, не оказывают влияния на выполнение остальных задач.

Обработка прерываний и исключений в защищённом режиме по аналогии с реальным режимом базируется на таблице прерываний. Но таблица прерываний защищённого режима является таблицей дескрипторов, которая содержит так называемые вентили прерываний, вентили исключений и вентили задач.

Таблица прерываний защищённого режима называется дескрипторной таблицей прерываний IDT (InterruptDescriptorTable). Также как и таблицы GDT и LDT, таблица IDT содержит 8-байтовые дескрипторы. Причём это системные дескрипторы - вентили прерываний, исключений и задач. Поле TYPE вентиля прерывания содержит значение 6, а вентиля исключения - значение 7.

Формат элементов дескрипторной таблицы прерываний IDT:



1. **Управление задачами в защищенном режиме. Переключение задач в защищенном режиме.**

Защищенный режим предоставляет средства переключения задач. Состояние каждой задачи (значение всех связанных с ней регистров процессора) может быть сохранено в специальном сегменте состояния задачи (TSS), на который указывает селектор в регистре задачи. При переключении задач достаточно загрузить новый селектор в регистр задачи, и состояние предыдущей задачи автоматически сохранится в ее TSS, а в процессор загрузится состояние новой (возможно, и ранее прерванной) задачи и начнется (продолжится) ее выполнение. Следующий очень важный механизм, действующий только в защищенном режиме, — многозадачность. Задача — это элемент работы, который процессор может исполнять, запустить или отложить. Задачи используют для выполнения программ, процессов, обработчиков прерываний и исключений, ядра операционной системы и пр. Любая программа, выполняющаяся в защищенном режиме, должна осуществляться как задача (хотя мы пока игнорировали это требование). Процессор предоставляет средства для сохранения состояния задачи, запуска задачи и передачи управления из одной задачи в другую.

Задача состоит из сегмента состояния задачи (TSS), сегмента кода, одного или нескольких (для разных уровней привилегий) сегментов стека и одного или нескольких сегментов данных. Задача определяется селектором своего сегмента TSS. Когда задача выполняется, ее селектор TSS (вместе с дескриптором в скрытой части) загружен в регистр TR процессора.

Запуск задачи осуществляется при помощи команды CALL или JMP на сегмент TSS или на шлюз задачи, а также при запуске обработчика прерывания или исключения, который описан как шлюз задачи. При этом автоматически осуществляется переключение задач. Состояние текущей задачи записывается в ее TSS, состояние вызываемой задачи считывается изее TSS, и управление передается на новые CS:EIP. Если задача не была запущена командой JMP, селектор сегмента TSS старой задачи сохраняется в TSS новой и устанавливается флаг NT, так что следующая команда IRET выполнит обратное переключение задач.

Задачи не могут вызываться рекурсивно. В дескрипторе TSS-задачи, которая была запущена, но не была завершена, тип изменяется на «занятый TSS» и переход на такой TSS невозможен. Задача может иметь собственную таблицу дескрипторов (LDT) и полный комплект собственных таблиц страниц, так как регистры LDTR и CR3 входят в состояние задачи.

1. **Суперскалярная архитектура МП.**

Суперскалярность — архитектура вычислительного ядра, использующая несколько декодеров команд, которые могут загружать работой множество исполнительных блоков. Планирование исполнения потока команд является динамическим и осуществляется самим вычислительным ядром.

Если в процессе работы команды, обрабатываемые конвейером, не противоречат друг другу, и одна не зависит от результата другой, то такое устройство (ядро) может осуществить параллельное выполнение команд. В суперскалярных системах решение о запуске инструкции на исполнение принимает сам вычислительный модуль, что требует много ресурсов. В более поздних системах используется статпланирование, то есть параллельные инструкции объединяются компилятором в длинную команду, в которой все инструкции заведомо параллельные.

Микропроцессор считается скалярным, если имеет конвеер. При наличии двух и более конвееров, такой МП считатется суперскалярным. С каждым новым конвеером скорость увеличивается на 30-40%.

Некоторые способы увеличения скорости:

Кэширование – способ увеличения быстродействия системы за счёт хранения часто используемых команд и данных в памяти.

Предсказание правил адреса перехода – использование часто используемых команд. Под переходом понимается алгоритм запланированного изменения перехода программ. Типичная программа на каждые 6-8 команд содержит 1 команду перехода. При наличии в микропроцессоре конвеера, через каждые 6-8 команд конвеер очищается и заполняется в соответствии с адресом перехода, таким образом, теряется преимущество конвеера, поэтому и был введён блок предсказания переходов. В Pentium имеется буфер переходов, хранящий последние 256 адресов перехода.

1. **Конвейерная обработка команд. Динамическое исполнение команд.**

Конвеер имеет 5 ступеней:

1. Выработка команды из кэша памяти и оперативной памяти;
2. Декодирование команд;
3. Генерация адреса для опеределения местоположения операндов в памяти;
4. Выполнение операций с помощью АЛУ;
5. Запись результата в память.

На стадии выполнения каждого этапа, каждая машинная команда разбивается на более мелкие операции, из-за этого скорость вычислений возрастает.

Динамическое выполнение: Благодаря динамическому выполнению процессор может параллельно обрабатывать большое количество команд. Основные особенности динамического выполнения следующие:

* множественное предсказание ветвлений: это средство предназначено для прогнозирования значения счетчика команд при выполнении команд ветвления;
* анализ потока данных: благодаря ему можно получить информацию, необходимую для планирования выполнения команд, независимо от их первоначального порядка в программе;
* упреждающее выполнение: "предугадывает" изменения счетчика команд и выполняет команды, результаты которых, вероятно, вскоре понадобятся.