Лекция №5. Адресное пространство И СПОСОБЫ АДРЕСАЦИИ В РЕАЛЬНОМ И защищенном режиме.

Программирование защищенного режима. Особенности 32-разрядных процессоров

Процессор содержит около 40 программно-адресуемых регистров (не считая регистров сопроцессоров), из которых 6 являются 16-разрядными, а прочие - 32-разрядными.

Регистры данных:

eax - аккумулятор;

ebx - базовый регистр;

ecx - счетчик;

edx - регистр данных;

Регистры-указатели:

esi - индекс источника;

edi - индекс приемника;

ebp - указатель базы;

esp - указатель стека.

Сегментные регистры - 16-разрядные.

- CS, DS, ES, FS, GS, SS.

Регистры общего назначения и регистры-указатели отличаются от аналогичных регистров процессора 8086 тем, что они являются 32-разрядными.

Для сохранения совместимости с ранними моделями процессоров допускается обращения к младшим половин всех регистров, имеющих те же мнемонические обозначения, что и в МП8086 (AX, BX, CX, DX, SI, DI, BP и SP).

Сохранена возможность работы с младшими и старшими половинками регистров МП8086. Однако старшие половины 32-разрядных регистров процессора не имеют мнемонических обозначений и непосредственно недоступны.

Для того, чтобы получить содержание старшей половины, например, регистра EAX (Биты 31 ... 16) придется сдвинуть содержание EAX на 16 бит справа (в регистр AX) и прочитать потом содержимое регистра AX.

В состав сегментных регистров включены еще 2 регистра: FS и GS, которые могут использоваться для сохранения сегментных адресов двух дополнительных сегментов

данных. То есть при работе в реальном режиме по программе можно обеспечить доступ до четырех сегментов данных, а не к двум, как при использовании МП8086.

Регистр флагов процессору 486 принято называть EFLAGS. дополнительно к шести флагов состояния (CF, PF, AF, ZF, SF и OF) и трех флагов управления состоянием процессору (TF, IF, DF), он включает 3 новых флаги:

NT, RF и VM и двухразрядное поле IOPL.

Новые флаги NT, RF, VM и поле IOPL используются процессором только в защищенном режиме.

Двухразрядное поле привилегий ввода-вывода IOPL (Input / Output Privilege Level) указывает на максимальное значение уровня текущего приоритета (от 0 до 3), при котором команды в / в выполняются без генерации исключительной ситуации.

Флаг вложенной задачи NT (Nested Task) показывает, текущая задача вложенной в исполнение другой задачи. В этом случае NT = 1. флаг устанавливается автоматически при переключении задач. Значение NT проверяется командой iret для определения способа возврата в процедуру, которая вызвала.

Управляющий флаг рестарта RF (Restart Flag) используется вместе с налаживая регистрами. Если RF = 1, то ошибки, возникающие при налаживание при использовании команды, игнорируются к выполнению следующей команды.

Управляющий флаг виртуального режима VM (Virtual Mode) используется для перевода процессора с защищенного режима в режим виртуального процессора 8086. В этом случае процессор функционирует как быстродействующий МП 8086, но реализует механизмы защиты памяти, страничной адресации и ряд других возможностей.

При работе с процессором программист имеет доступ к четырем управляющих регистрів CR0 ... CR3, в которых содержится информация о состоянии компьютера. эти регистры доступны только в защищенном режиме для программ, имеющих уровень привилегий 0.

Регистр CR0 - это слово состояния системы.

Для управления режимом работы процессора и определения его состояния используются следующие шесть битов регистра CR0.

бит 6 - бит страничного преобразования PG (Paging Enable). Если этот бит установлен, то страничное преобразование разрешено; если сброшен, то запрещено.

бит 4 - бит типа сопроцессора ET (Extension Type) в ПМ 80286 и 80386 указывал на тип подключенного сопроцессора. Если ET = 1, то 80387, если ET = 0, то 80287. В более новых процессорах бит ET всегда установлен.

бит 3 - бит переключения задачи TS (Task Switched). Этот бит автоматически устанавливается процессором при каждом переключении задачи. Бит может быть очищенный командой *clts,* которую можно использовать только на нулевом уровне привилегий.

бит 2 - бит эмуляции сопроцессору EM (Emulate). Если EM = 1, то обработка команд сопроцессору выполняется программно.

бит 1 - бит присутствия арифметического сопроцессору MP (Math Present). Операционная система устанавливает MP = 1, если сопроцессор присутствует. Этот бит управляет работой команды wait, которая используется для синхронизации работы программы и сопроцессору.

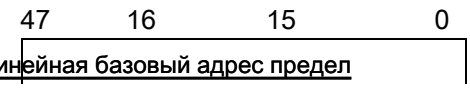
бит 0 - бит разрешения защиты PE (Protection Enable). При PE = 1 процессор работает в защищенном режиме; при PE = 0 - в реальном. PE может быть установлен при загрузке регистра CR0, командами *lmsw* или *mov cr0* , А сброшенный только командой *mov cr0* .

Регистр CR1 зарезервирован фирмой INTEL для следующих моделей процессоров. Регистры CR2 и CR3 служат для поддержания страничного преобразования адреса. эти два регистра используются вместе. CR2 содержит полную линейную адрес команды, вызвавшей последнюю исключительную ситуацию на странице, а CR3 - адрес, указывающий базу каталога страницы.

Регистры системных адресов.

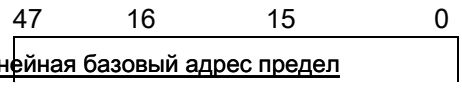
Регистры системных адресов входящих в состав процессору и используются в защищенном режиме работы процессора. Они задают расположение системных таблиц, служащие для организации системной адресации в защищенном режиме.

Регистр таблицы глобальных дескрипторов.



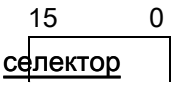
GDTR (Global Descriptor Table Register) - служит для сохранения линейной базового адреса и границы таблицы глобальных дескрипторов.

Регистр таблицы дескрипторов прерываний.



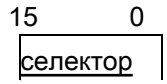
IDTR (Interrupt Descriptor Table Register) - для сохранения линейной базовой адреса и границы таблицы дескрипторов прерываний.

Регистр таблицы локальных дескрипторов.



LDTR (Local Descriptor Table Register) - для сохранения селектора сегмента таблицы дескрипторов.

Регистр состояния задачи.



TR (Task Register) - для сохранения селектора сегмента состояния задачи.

В систему команд современных процессоров включен ряд новых команд. команды общего назначения.

bound - проверка индекса массива по границ массива.

bsf / bsr - команды сканирования битов.

bt / btc / btr / bts - команды выполнения битовых операций.

bswap - изменение порядка байтов операнда.

cdq - превращение двойного слова в четверное.

cmpsd - сравнение строк по двойным словам.

cmpxchg - сравнение и обмен операндов.

cmpxchg86 - сравнение и обмен 8-битных операндов.

cpuid - идентификация процессору.

cwde - превращение слова в двойное слово с расширением

enter - создание кадра стека для параметров процедур.

imul reg, imm - умножение операнда со знаком на непосредственное значение.

ins / outs - ввода / вывода из порта в строку.

iretd - возвращение из прерывания в 32-разрядном режиме.

j (cc) - команды условного перехода, допускающих 32-битный смещение.

leave - выход из процедуры с удалением кадра стека, созданного командой enter.

lss / lfs / lgs - команды загрузки сегментных регистров.

mov DRx, reg; reg, DRx

mov CRx, reg; reg, CRx

mov TRx, reg; reg, TRx - команды обмена данными со специальными регистрами. Как источник приемника могут быть использованы регистры CR0 ... CR3, DR0 - DR7, TR3 - TR5.

movsx / movzx - знаковое / беззнаковое расширения до размера приемника и передатчика.

popa - извлечение из стека всех 16-разрядных регистров общего назначения (ax, bx, cx, dx, sp, bp, si, di).

popad - извлечение из стека всех 32-разрядных регистров общего назначения (eax, ebx, ecx, edx, ESP, ebp, esi, EDI).

push imm - запись в стек непосредственного операнда размером байт, слово, двойное слово (например, push 0FFFFFFFFh).

pusha - запись в стек всех 16-разрядных регистров общего назначения.

pushad - запись в стек всех 32-разрядных регистров общего назначения.

rcr / rcl / ror / rol reg / mem, imm - циклический сдвиг на непосредственное значение.

sar / sal / shr / shl reg / mem, imm - арифметический сдвиг в непосредственное значение.

scads - сканирование строки двойных слов с целью сравнения.

set (cc) - установление биту по условию.

shrd / shld - логический сдвиг с двойной точностью.

stosd - запись двойного слова в строку.

xadd - обмен и добавления.

xlatb - табличная трансляция.

Команды защищенного режима.

arpl - корректировка поля RP2 селектору.

clts - сброс флага переключения задач в регистре CR0.

lar - загрузка байта разрешения доступа.

lgdt - загрузка регистра таблицы глобальных дескрипторов.

lidt - загрузка регистра таблицы прерываний.

lldt - загрузка регистра таблицы локальных дескрипторов.

lmsw - загрузка слова состояния машины.

lsl - загрузка пределы сегмента.

ltr - загрузка регистра задачи.

rdmsr - чтение особого регистра модели.

sgdt - сохранение регистра таблицы глобальных дескрипторов.

sidt - сохранение регистра таблицы прерываний.

sedt - сохранение регистра таблицы локальных дескрипторов.

smsw - сохранение слова состояния.

ssl - сохранение границы сегмента.

str - сохранение регистра задачи.

verr - проверка доступности сегмента для чтения.

verm - проверка доступности сегмента для записи.

Знакомство с защищенным режимом.

Переводом микропроцессора в защищенный режим можно полностью реализовать все возможности, заложенные в его архитектуру и недоступны в реальном режиме. сюда можно отнести:

- увеличение адресного пространства до 4 Гбайт;
- возможность работать в виртуальном адресном пространстве, что превышает максимально возможный объем физической памяти и достигает величины 64 Тбайт.

Для реализации такого режима необходима соответствующая операционная система, сохраняет все сегменты выполняемых программ в большом дисковом пространстве, автоматически загружая в оперативную память те или иные сегменты по необходимости.

Организация многозадачного режима с параллельным выполнением нескольких программ (процессов). Такой режим организует многозадачная операционная система, но

микропроцессор предоставляет для этого режима механизм защиты задач друг от друга по помощью четырехуровневой системы привилегий.

Страничная организация памяти, повышает уровень защиты задач друг от друга, и эффективность их выполнения.

Программы, написанные для защищенного режима очень сложные. Но можно упростить, использовав то, что отдельные архитектурные особенности защищенного режима оказываются достаточно замкнутыми и независимыми. Так, при работе в однозадачных режиме отпадает необходимость в изучении различных методов защиты задач друг от друга. Во многих случаях можно не включать механизм страничной организации памяти. Часто нет необходимости использовать уровне привилегий.

В отличие от реального режима, в котором сегменты определяются их базовыми адресами, задаются программистом в явной форме, в защищенном режиме для каждого сегмента программы должен быть определен дескриптор - 8-байтовое поле, в котором записывается базовый адрес сегмента, его длина и другие характеристики.

Для обращения к необходимому сегмента программист заносит в сегментный регистр не сегментный адрес, а селектор, в состав которого входит номер соответствующего сегмента дескриптору.

биты	15 3	2	1	0
Название	индекс дескриптору (Номер)	TTI	RPL	

Рис. 1 Селектор дескриптору:

Где: RPL - уровень привилегий; TI = 0 дескриптор глобальный; TI = 1дескриптор локальный.

Процессор по этому номеру находит нужный дескриптор, извлекает из него базовый адрес сегмента, и добавляя к ней указан в конкретной команде сдвиг, формирует ячейки памяти. Индекс дескриптору записывается в селектор начиная с биту 3, что эквивалентно умножению его на 8 (2 ^ 3). Таким образом, можно считать, что селекторы последовательных дескрипторов представляют собой числа 0, 8, 16, 26 ....

Дескрипторы могут быть 3-х типов: памяти, системные и шлюзы. форматы дескрипторов памяти и системных совпадают. Формат дескриптору сегмента памяти показано на рисунке 3.2.

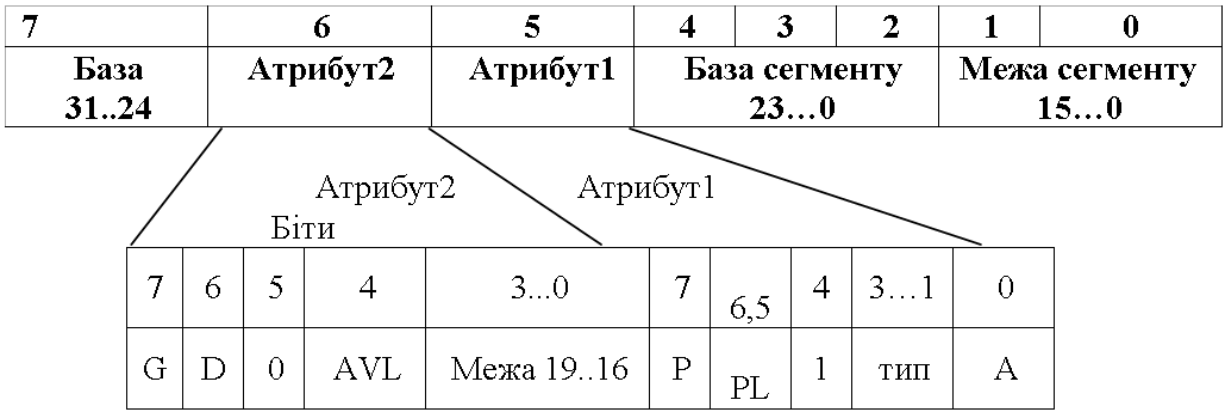


Рис. 2 Формат дескриптору сегмента памяти

Дескриптор занимает 8 байтов и предназначен для использования прикладными программами. В байтах 2-4 и 7 записывается линейная сегментный адрес базы сегмента. Так как базовый адрес имеет 32 бита, она может быть назначена в любой точке 4 гигабайтного адресного пространства.

В байтах 0,1 записываются младшие 16 бит границы сегмента, а в младшие 4 бита байта атрибутов 2 - биты 16 ... 19 оставшихся. Предел описывается 20-ю битами и ее числовое значение не может превышать 1М (2 ^ 20).

Но единицы, в которых задается предел, можно изменять, что осуществляется помощью бита дробовости G (бит 7 байта атрибутив2).

Если G = 0, предел указывается в байтах, если G = первую блоках по 4К. база сегмента в обоих случаях задается с точностью до байта.

Если G = 1, действительный предел сегмента определяется следующим образом «Граница сегмента = Граница в дескрипторе * 4К + 4095».

Таким образом, сегмент всегда простирается до конца последнего 4К-байтного блока.

Пусть, например, базовый адрес сегмента равна 0, предел тоже равна 0 и бит дробовости установлен. тогда действительно предел сегмента равна: 0 * 4К + 4095 = 4095.

То есть, сегмент будет занимать адреса 0 ... 4095 и иметь размер 4-Кбайта. если установить значение предела = 1, размер сегмента будет 8 Кбайт и т.д.

Рассмотрим атрибуты сегмента :

бит A - младший бит типа (Accessed, было обращение) устанавливается процессором в тот момент, когда в какой-нибудь сегментный регистр загружается селектор данного сегмента. Анализируя биты обращения различных сегментов, программа может судить о том, было ли обращение к данному сегменту после того, как она сбросила бит A.

Тип сегмента - занимает 3 бита и может иметь 8 значений:

0 - разрешено только чтение (сегмент данных);

1 - разрешено чтение и запись (сегмент данных);

2 - расширение вниз, разрешено только чтение (сегмент стека)

3 - расширение вниз, разрешены чтение и запись (сегмент стека)

4 - разрешено только выполнение (сегмент команд);

5 - разрешено выполнение и чтение (сегмент команд);

6 - разрешено только выполнение (подчиненный сегмент);

7 - разрешено выполнение и чтение (подчиненный сегмент).

Тип определяет правила доступа к сегменту. Таким образом, в защищенном режиме не предусмотрено составление программ, которые сами модифицируются.

Подчиненные согласованы сегменты (conforming) обычно используются для хранения подпрограмм общего пользования; для них не действуют общие правила защиты программ друг от друга.

Сегменты с расширением вниз, то есть в сторону меньших адресов, используемых для организации стеков, которые по ходу выполнения программы должны расширяться.

бит 4 байта атрибутив1 является идентификатором сегмента. Если он равен 1, дескриптор описывает сегмент памяти, 0 - это дескриптор системного сегмента.

поле DPL (Descriptor Privilege Level) служит для защиты программ друг от другой. Программам операционной системы обычно назначается уровень 0, прикладным программам - уровень 3.

бит P - присутствие сегмента в памяти.

бит AVL (От Available - доступный) не используется и не реализуется процессором.

бит D (Default, замалчивание) определяет действующий по умолчанию размер для операндов и адресов. Если D = 0, то используются 16-бітові, D = 1 - 32-бітові. он изменяет характеристики сегментов 2-х типов: выполняемых и стека. Атрибут сегмента, действующий по умолчанию, можно изменить на противоположный с помощью префиксов изменения размера операнда (66h) и изменения размера адреса (67h). Таким образом, после сегмента с D = 0 префикс 66h перед некоторой командой заставляет ее рассматривать свои операнды как 32-битные, а для сегмента с D = 1 тот же префикс 66h, наоборот, сделает операнды 16-битными.

Иногда транслятор сам включает в объектный модуль необходимые префиксы. При использовании команды iret в защищенном режиме префикс 66h необходимо включать в программу в явном виде, чтобы процессор снял из стека не три слова, как конечно, а три двойных слова. Без префикса команда iret будет выполняться неверно.

Приставка изменения адреса 67h необходимо использовать, например, перед командой loop, если в качестве счетчика цикла используется не CX, а ECX.

Сегменты стеке, адресуемых через регистр SS, с помощью бита D отмечают, регистр использовать как указатель стека в командах push и pop. Если D = 0 используется регистр SP, если D = 1, то ESP.

В байте атрибутив1 задаются характеристики сегмента. Например, для сегмента команд этот байт может иметь значение 98h (сегмент выполняется), для сегмента данных этот байт может иметь значение 92h (разрешены запись и чтение).

Дополнительные характеристики сегмента указываются в старшем полубайтами байта атрибутов 2 (в частности, тип дробовости).

Сегмент данных в программе должен начинаться с описания таблицы глобальных дескрипторов. В таблице дескрипторов должно быть описано столько дескрипторов, сколько сегментов использует программа. Порядок дескрипторов в таблице не имеет значение, кроме нулевого, который всегда занимает первое место.

Сегменты команд в защищенном режиме нельзя модифицировать по ходу выполнения программы.