Лекция:

**Архитектура 32-битных микропроцессоров семейства Intel IA-32. Часть 1**

**Регистры**

Программная модель включает восемь регистров общего назначения, шесть регистров сегментов, указатель команд, регистр системных флагов, регистры системных адресов, четыре регистра управления и шесть регистров отладки.

Восемь регистров общего назначения имеют длину в 32 бит и содержат адреса или данные. Они поддерживают операнды-данные длиной 1, 8, 16, 32 и (при использовании двух регистров) 64 бит; битовые поля от 1 до 32 бит; операнды-адреса длиной 16 и 32 бит. Эти регистры называются EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP. Доступ к младшим 16 бит этих регистров выполняется независимо при использовании соответствующих имен 16-битных регистров: AX, BX, CX, DX, SI, DI, BP и SP. Также могут использоваться индивидуально младший (биты 0-7) и старший (биты 8-15) байты регистров AX, BX, CX, DX. Им соответствуют обозначения AH, DH, CH, BH и AL, DL, CL, BL.

Хотя регистр ESP тоже относится к регистрам общего назначения, он содержит указатель на вершину стека и не используется для других целей.

Следует отметить, что регистры могут быть неравнозначны и при использовании определенных инструкций могут иметь специальное значение:

* EAX - аккумулятор, операнд-источник или приемник результата (некоторые инструкции могут быть короче на один байт при использовании EAX);
* EBX - указатель на данные в сегменте DS;
* ECX - счетчик для цепочечных (например, MOVS) и циклических (с префиксом REP) инструкций;
* EDX - адрес порта ввода-вывода для инструкций IN/INS, OUT/OUTS;
* ESI - указатель на операнд-источник в сегменте DS для цепочечных инструкций;
* EDI - указатель на операнд-приемник в сегменте ES для цепочечных инструкций;
* EBP - указатель на данные в сегменте SS.

МП включает шесть непосредственно доступных 16-битных регистров сегментов. С каждым сегментным регистром ассоциирован программно-недоступный кэш дескриптора соответствующего сегмента, содержащий базовый адрес сегмента в линейном адресном пространстве, предел сегмента и атрибуты сегмента. Этот кэш заполняется при загрузке значения в сегментный регистр. В реальном режиме предел сегмента всегда 0FFFFh, атрибуты игнорируются, а базовый адрес вычисляется сдвигом значения селектора на 4 бита влево. В защищенном режиме кэш заполняется соответствующими значениями из дескрипторной таблицы.

Не все сегментные регистры равнозначны. Регистр CS хранит селектор сегмента кода. МП извлекает очередную инструкцию для исполнения, формируя логический адрес из селектора в CS и смещения в регистре EIP. Значение этого регистра нельзя изменить непосредственно, оно меняется в командах межсегментного перехода (FAR JMP), межсегментного вызова (FAR CALL), при вызове обработчика прерывания (INT) и при возврате из далекой процедуры (RETF) или обработчика прерывания (IRET).

Регистр SS хранит селектор сегмента стека. Стек используется для передачи параметров подпрограммам и для сохранения адреса возврата при вызове подпрограммы или обработчика прерывания. Вершиной стека считается байт, логический адрес которого образуется из селектора в регистре SS и смещения в регистре ESP. Программа может непосредственно изменить значение SS, что дает ей возможность переключаться между несколькими стеками. Причем на время выполнения команды MOV SS,xxxx и одной команды, следующей за ней (обычно это MOV ESP,xxxx), запрещаются маскируемые и блокируются немаскируемые прерывания.

Регистры DS, ES, FS и GS хранят селекторы сегментов данных. Если инструкция обращается к памяти, но содержит только смещение, то считается, что она обращается к данным в сегменте DS. Сегмент ES может использоваться без явного указания в цепочечных командах. Сегменты FS и GS используются при обращении к памяти только при явном использовании в инструкции префиксов этих сегментов.

Указатель команд (EIP) является 32-разрядным регистром. Он содержит смещение следующей команды, подлежащей выполнению. Относительный адрес отсчитывается от базового адреса сегмента исполняемой задачи. Указатель команд непосредственно недоступен программисту, но он изменяется явно командами управления потоком, прерываниями и исключениями (JMP, CALL, RET, IRET, команды условного перехода). Получить текущее значение EIP можно, если выполнить команду CALL, а затем прочитать слово на вершине стека.

Регистр системных флагов EFLAGS содержит группу флагов состояния, управления и системных флагов. Младшие 16 бит регистра представляют собой 16-разрядный регистр флагов и состояния МП 8086, называемый FLAGS, который наиболее полезен при исполнении программ для МП 8086 и Intel-286. Некоторые из флагов могут быть изменены специально предназначенными для этой цели инструкциями. Для изменения или проверки группы флагов можно воспользоваться следующими командами:

* LAHF/SAHF - загрузка/сохранение младших 8 битов регистра флагов в регистре AH;
* PUSHF/POPF - помещение/извлечение из стека младших 16 битов регистра флагов;
* PUSHFD/POPFD - помещение/извлечение из стека 32-битного регистра EFLAGS.

Регистры управления сегментированной памятью, известные также как регистры системных адресов, указывают на структуры данных, которые управляют механизмом сегментированной памяти. Они определены для ссылок на таблицы или сегменты, поддерживаемые моделью защиты МП.

* Регистр глобальной дескрипторной таблицы (GDTR). Содержит 32-битный линейный адрес и 16-битную границу глобальной дескрипторной таблицы. Значение этого регистра можно загрузить/сохранить при помощи привилегированных инструкций LGDT/SGDT. В реальном режиме этот регистр не используется. Перед переходом в защищенный режим в этот регистр следует загрузить корректные значения.
* Регистр локальной дескрипторной таблицы (LDTR). Содержит 16-битный селектор локальной дескрипторной таблицы. С регистром связан программно-недоступный кэш дескриптора для хранения базового адреса, предела и атрибутов соответствующей дескрипторной таблицы. Значение этого регистра можно загрузить/сохранить при помощи привилегированных инструкций LLDT/SLDT. В реальном режиме этот регистр не используется, и попытка обращения к нему генерирует особый случай "недействительный код операции" (исключение #6). С каждой задачей в защищенном режиме может быть связана своя локальная дескрипторная таблица, поэтому селектор LDT хранится в TSS и автоматически загружается при переключении задач.
* Регистр таблицы дескрипторов прерываний (IDTR). Указывает на таблицу точек входа в программы обработки прерываний. Регистр содержит 32-битный линейный базовый адрес и 16-битный предел таблицы. Значение этого регистра можно загрузить/сохранить при помощи привилегированных инструкций LIDT/SIDT. При инициализации МП базовый адрес IDT устанавливается в 0, а предел - 0FFFFh. В реальном режиме эта таблица хранит 4-байтные векторы прерываний, а в защищенном - 8-байтные дескрипторы шлюзов обработчиков прерываний и исключений. Это единственный регистр среди перечисленных, который используется в реальном режиме.
* Регистр задачи (TR). Указывает на информацию, необходимую МП для определения текущей задачи. Регистр содержит 16-битный селектор дескриптора сегмента состояния задачи. С регистром связан программно-недоступный кэш дескриптора TSS для хранения базового адреса, предела и атрибутов соответствующего сегмента состояния задачи. Значение этого регистра можно загрузить/сохранить при помощи привилегированных инструкций LTR/STR. В реальном режиме этот регистр не используется, и попытка обращения к нему генерирует особый случай "недействительный код операции" (исключение #6).

МП имеет четыре 32-разрядных регистра управления CR0-CR4, в которых хранятся флаги состояния МП или глобальные флаги. Вместе с регистрами системных адресов эти регистры хранят информацию о состоянии МП, которая влияет на все задачи в системе. Системным программистам регистры управления доступны только через варианты команды MOV, которые позволяют их загружать или сохранять в регистрах общего назначения.

Шесть доступных регистров отладки (DR0-DR3, DR6, DR7, регистры DR4 и DR5 зарезервированы) расширяют возможности отладки. Они устанавливают точки останова по данным и позволяют устанавливатьзадавать точки останова по командам без модификации сегментов программ. Регистры DR0-DR3 предназначены для хранения четырех линейных адресов точек останова. Регистр DR6 отражает текущее состояние точек останова. Регистр DR7 задает условие для точек останова.

В МП Intel-386 и Intel-486 использовались также 2 регистра страничных проверок (TR6 и TR7), которые позднее были исключены из архитектуры IA-32.

**Формат команды микропроцессора IA-32**

Инструкция микропроцессора может содержать следующие поля:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **префикс** | **КОП** | **Mod R/M** | **SIB** | **смещение** | **непосредственный операнд** |
| 0/1 байт | 1/2 байта | 0/1 байт | 0/1 байт | 0/1/2/4 байта | 0/1/2/4 байта |

Префикс - необязательная часть инструкции, которая позволяет изменить некоторые особенности ее выполнения. В команде может быть использовано сразу несколько префиксов разного типа. Типы префиксов: командные префиксы (префиксы повторения) REP, REPE/REPZ, REPNE/REPNZ; префикс блокировки шины LOCK; префиксы размера; префиксы замены сегмента.

КОП - код операции.

Байт "Mod R/M" определяет режим адресации, а также иногда дополнительный код операции. Необходимость байта "Mod R/M" зависит от типа инструкции.

Байт SIB (Scale-Index-Base) определяет способ адресации при обращении к памяти в 32-битном режиме. Необходимость байта SIB зависит от режима адресации, задаваемого полем "Mod R/M".

Кроме того, инструкция может содержать непосредственный операнд и/или смещение операнда в сегменте данных.

На размер инструкции накладывается ограничение в 15 байт. Инструкция большего размера может получиться при некорректном использовании большого количества префиксов. В IA-32 в таком случае генерируют исключение #13.

Если инструкция микропроцессора требует операнды, то они могут задаваться следующими способами: непосредственно в коде инструкции (только операнд-источник); в одном из регистров; через порт ввода-вывода; в памяти.

Для совместимости с 16-битными процессорами архитектура IA-32 использует одинаковые коды для инструкций, оперирующих как с 16-битными, так и 32-битными операндами. Новая архитектура предусматривает также новые возможности при указании адреса для операнда в памяти. Как процессор будет считать операнд или его адрес, зависит от эффективного размера операнда и эффективного размера адреса для данной команды. Эти значения определяются на основе режима работы, бита D дескриптора используемого сегмента и наличия в инструкции определенных префиксов.

Непосредственный режим адресации подразумевает включение операнда-источника в код инструкции. Операнд может быть 8-битовым или 16-битовым, если значение эффективного размера операнда - 16. Операнд может быть 8-битовым или 32-битовым, если значение эффективного размера операнда - 32. Обычно непосредственные операнды используются в арифметических инструкциях.

Регистровый режим адресации определят операнд-источник или операнд-приемник в одном из регистров процессора или сопроцессора.

В некоторых случаях, (например, в инструкциях DIV и MUL) могут использоваться пары 32-битных регистров (например, EDX:EAX), образуя 64-битный операнд.

Адресация через порт ввода-вывода подразумевает получение операнда или сохранение операнда через пространство портов ввода-вывода. Адрес порта ввода-вывода либо непосредственно включается в код инструкции, либо берется из регистра DX.

Очень распространенный способ адресации операнда - адресация через память. Таким образом, может быть указан операнд-источник или операнд-приемник. Следует отметить, что процессор не позволяет одновременно задавать оба операнда через память (за исключением некоторых цепочечных команд).

Для получения операнда из памяти процессору необходимо знать селектор сегмента и смещение в сегменте. В некоторых командах селектор может быть указан непосредственно в коде инструкции. В других случаях процессор может явно или неявно использовать значение одного из сегментных регистров. Под неявным использованием сегментных регистров подразумевается то, что в зависимости от предназначения операнда процессор использует определенный сегментный регистр для обращения к памяти: CS -для выборки инструкций; SS - для работы со стеком или обращения к памяти через регистры ESP или EBP; ES - для получения адреса операнда-приемника в цепочечных командах; DS - при всех остальных обращениях к памяти. Явное использование сегментных регистров возможно, если в код инструкции включается префикс смены сегмента. Указание префикса смены сегмента допустимо не для всех команд: нельзя менять сегмент для команд работы со стеком (всегда используется SS); для цепочечных команд можно менять сегмент только операнда-источника (операнд-приемник всегда адресуется через ES).

Смещение в сегменте (эффективный или исполнительный адрес - EA) может быть вычислено на основе значений регистров общего назначения и/или указанного в коде инструкции относительного смещения, при этом любой или даже несколько из указанных компонентов могут отсутствовать:

EA = BASE + (INDEX\*SCALE) + DISPLACEMENT

Такая схема позволяет в языках высокого уровня и на языке Ассемблера легко реализовать работу с массивами.

**Вопросы для самоконтроля**

* Назовите основные блоки и их функциональное назначение в структуре IA-32.
* Какие регистры составляют программную модель IA-32?
* Перечислите регистры управления сегментированной памятью.
* В каких режимах может работать IA-32?
* Какие поля составляют команду IA-32?
* Перечислите типы префиксов.

Лекция:

**Архитектура 32-битных микропроцессоров семейства Intel IA-32. Часть 2**

**Режимы работы**

**Реальный режим (Real Mode)**

После инициализации (системного сброса) МП находится в реальном режиме. В реальном режиме МП работает как очень быстрый 8086 с возможностью использования 32-битных расширений. Механизм адресации, размеры памяти и обработка прерываний (с их последовательными ограничениями) МП 8086 полностью совпадают с аналогичными функциями других МП IA-32 в реальном режиме. В отличие от 8086, остальные члены семейства IA-32 в определенных ситуациях генерируют исключения, например, при превышении предела сегмента, который для всех сегментов в реальном режиме равен 0FFFFh.

Имеется две фиксированные области в памяти, которые резервируются в режиме реальной адресации: область инициализации системы и область таблицы прерываний.

Ячейки от 00000h до 003FFH резервируются для векторов прерываний. Каждое из 256 возможных прерываний имеет зарезервированный 4-байтовый адрес перехода. Ячейки от FFFFFFF0H до FFFFFFFFH резервируются для инициализации системы.

**Режим системного управления (System Management Mode)**

В новых поколениях МП Intel появился еще один режим работы - режим системного управления. Впервые он был реализован в МП 80386SL и i486SL. Начиная с расширенных моделей Intel-486, этот режим стал обязательным элементом архитектуры IA-32. С его помощью прозрачно даже для операционной системы на уровне BIOS реализуются функции энергосбережения.

Режим системного управления предназначен для выполнения некоторых действий с возможностью их полной изоляции от прикладного программного обеспечения и даже от операционной системы. МП переходит в этот режим только аппаратно: по низкому уровню на контакте SMI# или по команде с шины APIC (Pentium+). Никакой программный способ не предусмотрен для перехода в этот режим. МП возвращается из режима системного управления в тот режим, при работе в котором был получен сигнал SMI#. Возврат происходит по команде RSM. Эта команда работает только в режиме системного управления и в других режимах не распознается, генерируя исключение #6 (недействительный код операции).

Когда МП находится в режиме SMM, он выставляет сигнал SMIACT#. Этот сигнал может служить для включения выделенной области физической памяти (System Management RAM), так что память SMRAM можно сделать доступной только для этого режима. При входе в режим SMM МП сохраняет свой контекст в SMRAM (контекст сопроцессора не сохраняется) и передает управление процедуре, называемой обработчиком System Management Interrupt. Состояние МП в этот момент точно определено: регистр EFLAGS обнулен (кроме зарезервированных битов), сегментные регистры содержат селектор 0000, базы сегментов установлены в 00000000, пределы 0FFFFFFFFh.

Следует отметить, что в режиме SMM не предусмотрена работа с прерываниями и особыми случаями: прерывания по IRQ и SMI# замаскированы, пошаговые ловушки и точки останова отключены, обработка прерывания по NMI откладывается до выхода из режима SMM. Если необходимо обеспечить работу с прерываниями или особыми случаями, то надо инициализировать IDT и разрешить прерывания, выставив флаг IF в регистре EFLAGS. Прерывания по NMI будут разблокированы автоматически после первой же команды IRET.

При возврате из SMM (по инструкции RSM) МП восстанавливает свой контекст из SMRAM. Обработчик может программно внести изменения в образ контекста МП, тогда МП перейдет не в то состояние, в котором произошло SMI.

Эти особенности режима системного управления позволяют использовать его для реализации системы управления энергосбережением компьютера или функций безопасности и контроля доступа.

**Защищенный режим (Protected Mode)**

Основным режимом работы МП является защищенный режим. Ключевые особенности защищенного режима: виртуальное адресное пространство, защита и многозадачность. МП может быть переведен в защищенный режим установкой бита 0 (Protect Enable) в регистре CR0. Вернуться в режим реального адреса МП может по сигналу RESET или сбросом бита PE (в Intel-286 недоступно).

В защищенном режиме программа оперирует с адресами, которые могут относиться к физически отсутствующим ячейкам памяти, поэтому такое адресное пространство называется **виртуальным**. Размер виртуального адресного пространства программы может превышать емкость физической памяти и достигать 64Тбайт.

Преобразование логического адреса в физический происходит в два этапа: сначала блок управления сегментами выполняет трансляцию адреса в соответствии с сегментированной моделью памяти, получая 32-битный линейный адрес, а затем блок страничного преобразования выполняет разбиение на страницы, преобразуя 32-битный линейный адрес в 32-битный или 36-битный (P6) физический. МП не предусматривает механизмов запрещения сегментации ; с другой стороны, страничная трансляция есть опциональный механизм и может использоваться либо не использоваться в зависимости от особенностей операционной системы.

В рамках сегментированной модели адресации для программы память представляется группой независимых адресных блоков, называемых сегментами. Для адресации байта памяти программа должна использовать логический адрес, состоящий из селектора сегмента и смещения. Селектор сегмента выбирает определенный сегмент, а смещение указывает на конкретный байт в адресном пространстве выбранного сегмента. Селектор сегмента может находиться либо непосредственно в коде команды, либо в одном из сегментных регистров. Смещение также может либо непосредственно находиться в коде команды, либо вычисляться на основе значений регистров общего назначения.

С каждым сегментом связана особая структура, хранящая информацию о нем: **дескриптор**. Дескриптор - это 8-байтная единица описательной информации, распознаваемая устройством управления памятью в защищенном режиме, хранящаяся в дескрипторной таблице. Дескриптор сегмента содержит базовый адрес описываемого сегмента, предел сегмента и права доступа к сегменту. В защищенном режиме сегменты могут начинаться с любого линейного адреса (который называется базовым адресом сегмента) и иметь любой предел вплоть до 4Гбайт.

**Дескрипторные таблицы** - это массивы памяти переменной длины, содержащие 8-байтные элементы: дескрипторы. Дескрипторная таблица может иметь длину от 8 байт до 64 Кбайт и в каждой таблице может быть до 8192 дескрипторов. Существуют две обязательных дескрипторных таблицы - глобальная дескрипторная таблица (Global Descriptor Table - GDT) и дескрипторная таблица прерывания (Interrupt Descriptor Table - IDT), а также множество (вплоть до 8191) необязательных локальных дескрипторных таблиц (Local Descriptor Table - LDT), из которых в каждый момент времени процессору доступна только одна. Расположение дескрипторных таблиц определяется регистрами процессора GDTR, IDTR, LDTR.

GDT содержит дескрипторы, доступные всем задачам в системе. GDT может содержать дескрипторы любых типов: и дескрипторы сегментов, и системные дескрипторы (кроме шлюзов прерываний и ловушек). Первый элемент GDT (с нулевым индексом) не используется. Ему соответствует нуль-селектор, обозначающий "пустой" указатель.

LDT обеспечивают способ изоляции сегментов программы и данных исполняемой задачи от других задач. LDT связана с конкретной задачей и может содержать только дескрипторы сегментов, шлюзы вызовов и шлюзы задач.

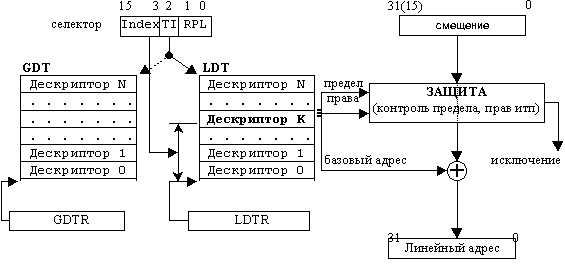
Сегмент не может быть доступен задаче, если его дескриптор не существует ни в текущей таблице LDT, ни в таблице GDT. Использование двух дескрипторных таблиц позволяет, с одной стороны, изолировать и защищать сегменты исполняемой задачи, а с другой - позволяет разделять глобальные данные и код между различными задачами.

IDT может содержать только шлюзы задач, шлюзы прерываний или шлюзы ловушек.

Для вычисления линейного адреса МП выполняет следующие действия (рис. 4.1):

1. МП использует селектор сегмента для нахождения дескриптора сегмента. Селектор содержит индекс дескриптора в дескрипторной таблице (Index), бит TI, определяющий, к какой дескрипторной таблице производится обращение (LDT или GDT), а также запрашиваемые права доступа к сегменту (RPL). Если селектор хранится в сегментном регистре, то обращение к дескрипторным таблицам происходит только при загрузке селектора в сегментный регистр, т. к. каждый сегментный регистр хранит соответствующий дескриптор в программно-недоступном ("теневом") регистре-кэше.
2. МП анализирует дескриптор сегмента, контролируя права доступа (сегмент доступен с текущего уровня привилегий ) и предел сегмента (смещение не превышает предел);
3. МП добавляет смещение к базовому адресу сегмента и получает линейный адрес.

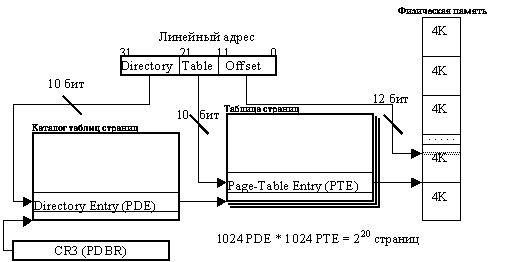
Если страничная трансляция отключена, то сформированный линейный адрес считается физическим и выставляется на шину процессора для выполнения цикла чтения или записи памяти.



**Рис. 4.1.** Схема преобразования логического адреса в линейный

Механизм сегментации обеспечивает превосходную защиту, но он не очень удобен для реализации виртуальной памяти (подкачки). В дескрипторе сегмента есть бит присутствия, по нему процессор определяет, находится ли данный сегмент в физической памяти или на внешнем запоминающем устройстве (на винчестере). В последнем случае генерируется исключение #11, обработчик которого может подгрузить сегмент в память. Неудобство заключается в том, что различные сегменты могут иметь различную длину. Этого можно избежать, если механизм подкачки реализовывать на основе страничного преобразования. Особенностью этого преобразования является то, что процессор в этом случае оперирует с блоками физической памяти равной длины (4 Кбайт) - страницами. Страницы не имеют непосредственного отношения к логической структуре программы. Кроме того, в МП подсемейства P6 страничная трансляция обеспечивает 36-битную физическую адресацию памяти (64 Гбайт). Страничное преобразование действует только в защищенном режиме и включается установкой в 1 бита PG в регистре CR0.

В страничном преобразовании участвуют два типа структур: каталоги таблиц (Page Directory) и таблицы страниц (Page Table). Эти структуры состоят из 1024 32-битных элементов. Элементы содержат старшие 20 бит физического адреса адресуемых объектов. Элементы таблицы страниц (Page Table Entry - PTE) адресуют страницы, а элементы каталога таблиц (Page Directory Entry - PDE) адресуют таблицы страниц. Старшие 20 бит физического адреса каталога таблиц хранятся в регистре CR3 (Page Directory Base Register - PDBR) (это единственный регистр процессора, который содержит физический адрес памяти). Все структуры выровнены по границе страницы.



**Рис. 4.2.** Стандартная 2-уровневая схема страничной трансляции

В процессе страничной трансляции адресов полученный линейный адрес разбивается на три части. Старшие десять бит (Directory) линейного адреса являются индексом элемента из каталога таблиц. По этому элементу определяется физический адрес таблицы страниц. Биты 21-12 (Table) линейного адреса выбирают элемент из этой таблицы страниц. Выбранный элемент определяет физический адрес страницы. Младшие 12 бит (Offset) линейного адреса определяют смещение от начала страницы.

Страницы начинаются на границах 4 Кбайт областей памяти, поэтому младшие 12 бит адреса страницы всегда равны нулю. В каталоге таблиц элементы хранят физические адреса таблиц страниц. В таблице страниц элементы хранят физические адреса самих страниц.

В МП Pentium фирма Intel реализовала новую возможность - расширение размера страниц (Page Size Extension - PSE). PSE позволяет использовать страницы размером 4 Мбайт и одноуровневый механизм страничной трансляции.

В подсемействе P6 шина адреса была расширена до 36 бит. Соответственно, процессоры Pentium Pro, Pentium II, Pentium III и более поздние способны адресовать до 64 Гбайт физической памяти. Эта возможность называется расширением физического адреса (Physical Address Extension - PAE) и доступна только при использовании страничной трансляции.

Использование PSE и PAE следующим образом влияет на страничное преобразование:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 4.1. Особенности страничного преобразования в PSE и PAE | | | | | |
| **CR0.PG** | **CR4.PAE** | **CR4.PSE** | **PDE.PS** | **Размер страницы** | **Разрядность физ. адреса** |
| 0 | X | x | x | - | 32 бит |
| 1 | 0 | 0 | x | 4 Кб | 32 бит |
| 1 | 0 | 1 | 0 | 4 Кб | 32 бит |
| 1 | 0 | 1 | 1 | 4 Мб | 32 бит |
| 1 | 1 | x | 0 | 4 Кб | 32 бит |
| 1 | 1 | x | 1 | 2 Мб | 32 бит |

Встроенные средства переключения задач обеспечивают многозадачность в защищенном режиме. **Задача** - это "единица измерения" заданий для процессора, которую процессор может выполнять, приостанавливать и осуществлять над ней диспетчеризацию. В качестве задачи может исполняться прикладная программа, сервис операционной системы, ядро операционной системы, обработчик прерывания или исключения и т.п. В защищенном режиме архитектура IA-32 предоставляет механизм для сохранения состояния задачи и переключения с одной задачи на другую. Все команды процессора выполняются в контексте той или иной задачи. Даже самые простые системы должны определить как минимум одну задачу. Более сложные системы могут использовать средства управления задачами для поддержки многозадачных приложений.

Среда задачи состоит из содержимого регистров МП и всего кода с данными в пространстве памяти. МП способен быстро переключаться из одной среды выполнения в другую, имитируя параллельную работу нескольких задач. Для некоторых задач может эмулироваться управление памятью, как у МП 8086. Такое состояние задачи называется режимом виртуального 8086 (Virtual 8086 Mode). О пребывании задачи в данном состоянии сигнализирует бит VM в регистре флагов. При этом задачи виртуального МП 8086 изолированы и защищены, как от друг друга, так и от обычных задач защищенного режима.

Задачу составляют два компонента: адресное пространство задачи и сегмент состояния задачи (Task State Segment - TSS).

В адресное пространство задачи входят доступные ей сегменты кода, данных и стека. Если используется механизм привилегий, то каждой задаче должны быть предоставлены сегменты стека для всех используемых уровней привилегий.

Сегмент состояния задачи хранит состояние регистров (контекст) процессора:

* состояние сегментных регистров (селекторы сегментов, формирующие адресное пространство задачи);
* состояние регистров общего назначения;
* состояние регистра флагов (EFLAGS);
* указатель очередной команды (EIP);
* значение регистра CR3 (PDBR);
* значение регистра LDTR.

В многозадачных системах TSS предоставляет механизм для связывания (вложения) задач.

Каждая задача идентифицируется селектором соответствующего ей TSS. Этот селектор загружается в регистр задачи (Task Register - TR) при переключении на задачу. В теневую часть регистра загружаются базовый адрес, предел и атрибуты TSS. Операционная система может предусматривать для каждой задачи свое линейное адресное пространство (свой набор страниц для страничного преобразования ), тогда при переключении задачи загружается и регистр CR3 (PDBR), хранящий адрес каталога таблиц для страничного преобразования.

В защищенном режиме процессор обеспечивает определенные механизмы защиты на основе сегментации и на основе страничного преобразования. Механизмы защиты позволяют ограничивать доступ к определенным сегментам или страницам при помощи уровней привилегий (4 для сегментов и 2 для страниц). Например, критические код и данные операционной системы могут быть расположены на более привилегированном уровне, чем прикладные программы. Это позволит ограничить и контролировать доступ прикладных программ к функциям и данным операционной системы.

Механизм защиты обеспечивает соответствие любой ссылки на ячейки памяти определенным условиям. Все проверки выполняются до начала цикла обращения к памяти. Нарушение любого условия приводит к генерации исключения. Проверки выполняются параллельно с формированием адреса и поэтому не ухудшают производительность процессора. Все ссылки должны пройти следующие проверки:

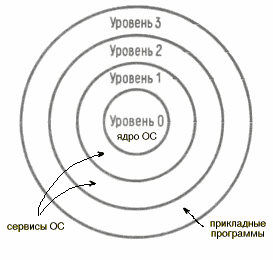
* контроль предела;
* контроль типа;
* контроль уровня привилегий ;
* контроль выравнивания;
* ограничение адресного пространства;
* ограничение точек входа в процедуры (для шлюзов);
* ограничение набора команд (привилегированные инструкции).

В защищенном режиме нет способа отключить механизм защиты. Даже если назначить всем сегментам и задачам нулевой (самый высший) уровень привилегий, все равно будут выполняться проверки контроля предела и типа. В то же время, механизм защиты на уровне страниц можно подавить, если назначить всем страницам уровень привилегий суперпользователя и обеспечить доступ на чтение и запись.

Контроль пределов и типов сегментов обеспечивает целостность сегментов кода и данных. Программа не имеет права обратиться к виртуальной памяти, выходящей за предел того или иного сегмента. Программа не имеет права обратиться к сегменту данных как к коду, и наоборот.

Архитектура защиты МП обеспечивает 4 иерархических уровня привилегий, что позволяет ограничить задаче доступ к отдельным сегментам в зависимости от ее текущих привилегий. Привилегии - это свойство (обычно устанавливаемое при проектировании системы), которое определяет, какие компьютерные операции разрешаются в любой момент времени и какие доступы к памяти законны. Привилегии используются для обеспечения безопасности в компьютерной системе. Привилегии реализуются путем присвоения значения от 0 до 3 ключевым объектам, которые опознаются процессором. Значение 0 соответствует наибольшим привилегиям, тогда как значение 3 - наименьшим.

Четыре уровня привилегий можно интерпретировать в виде колец защиты (рис. 4.3). Центр (уровень 0) предназначен для сегментов, содержащих наиболее критичные программы (обычно ядро операционной системы). Внешние кольца предназначены для сегментов с менее критичными программами или данными. Использование всех четырех уровней привилегий не является необходимым. Существующие системы, спроектированные с меньшим количеством уровней, могут просто игнорировать другие допустимые уровни. UNIX и Windows, например, используют только два уровня привилегий: 0 (для ядра системы) и 3 (для всего остального), а OS/2 использует уровни 0 (для ядра системы), 2 (для процедур ввода-вывода) и 3 (для прикладных программ).



**Рис. 4.3.** Интерпретация уровней привилегий в виде колец защиты

Механизм контроля уровня привилегий микропроцессора оперирует следующими значениями:

**CPL** - текущий уровень привилегий (Current Privilege Level): уровень привилегий, на котором в данный момент исполняется задача. Значение CPL хранится в поле RPL селектора сегмента кода, который помещен в регистр CS. Обычно это значение соответствует уровню привилегий дескриптора исполняемого сегмента кода. Уровень привилегий меняется, когда управление передается сегменту кода с другим значением DPL (за исключением подчиняемых сегментов кода).

**DPL** - уровень привилегий дескриптора (Descriptor Privilege Level): наименее привилегированный уровень, на котором задача может получить доступ к сегменту или шлюзу, связанному с этим дескриптором. Уровень DPL определяется битами 46 и 45 дескриптора.

**RPL** - запрашиваемый уровень привилегий (Requested Privilege Level) используется для временного понижения своего уровня привилегий при обращении к памяти. RPL заносится в младшие биты селектора.

Механизм контроля уровня привилегий обычно сравнивает уровень привилегий дескриптора (DPL) с максимальным из двух чисел CPL и RPL. Наименее привилегированный из текущего уровня привилегий и запрашиваемого считается эффективным уровнем привилегий: EPL = max (CPL, RPL).

Контроль привилегий при доступе к данным осуществляется при загрузке селектора в сегментный регистр DS, ES, FS, GS (либо при обращении к памяти, если селектор содержится в коде инструкции). Программа может обратиться к сегменту данных, который находится на том же или более низком уровне привилегий (с учетом RPL), т.е. доступ к данным разрешен, если max(CPL, RPL)<= DPL ; в противном случае генерируется нарушение общей защиты.

Контроль привилегий при доступе к стеку осуществляется при загрузке селектора в регистр SS. Программа должна использовать сегмент стека, находящийся на том же уровне привилегий, т.е. CPL = RPL = DPL.

Для передачи управления на обычный сегмент кода его уровень привилегий должен совпадать с текущим уровнем привилегий. Значение RPL должно быть не больше CPL, чтобы не вызывать исключения, но вне зависимости от значения RPL уровень привилегий не изменится: CPL = DPL. Текущий уровень привилегий может измениться при передаче управления через селектор шлюза. Шлюзы бывают четырех типов: шлюз вызова, шлюз задачи, шлюз ловушки и шлюз прерывания (последние два типа не используются в командах перехода и вызова). При передаче управления через селектор шлюза задачи или селектор TSS происходит переключение задач. Одна задача может передать управление другой задаче, при этом контролируется, чтобы объект, через который передается управление, был на том же или более низком уровне привилегий, чем CPL (правило контроля как для сегментов данных).

Кроме того, текущий уровень привилегий задачи влияет на возможность выполнения тех или иных специфических команд (привилегированных инструкций). Кроме привилегированных, существуют инструкции, результат выполнения которых зависит от поля IOPL в регистре флагов (I/O Privilege Level - уровень привилегий ввода-вывода): IN, INS, OUT, OUTS, CLI, STI. Механизм защиты процессора позволяет выполнять эти инструкции, только если задача обладает достаточными привилегиями, т.е. CPL <= IOPL.

Функции страничной трансляции, впервые появившиеся в Intel386, обеспечивают дополнительные механизмы защиты на уровне страниц. Особенно это удобно при использовании сплошной модели памяти, когда и операционная система, и прикладные программы работают в едином пространстве логических адресов, т.к. защита на уровне страниц может обеспечить разграничение привилегий для страниц операционной системы и прикладных программ. Защита на уровне страниц предоставляет два типа контроля: ограничение адресного пространства (страницы супервизора и страницы пользователя) и ограничение типа доступа (доступ только для чтения и доступ на чтение-запись). Если проверки не проходят, генерируется страничное нарушение (исключение #14).

Ограничение адресного пространства на уровне страниц обеспечивается двумя уровнями привилегий: режим супервизора (уровень 0) и режим пользователя (уровень 1). Режим супервизора соответствует привилегиям на уровне сегментов CPL = 0, CPL = 1 и CPL = 2. В режиме супервизора доступны все страницы. Режим пользователя соответствует привилегиям на уровне сегментов CPL = 3. В режиме пользователя доступны только страницы пользователя. В адресное пространство супервизора попадают страницы, для которых соответствующий элемент таблицы страниц или каталога таблиц содержит бит US = 0. В адресное пространство пользователя попадают страницы, для которых и соответствующий элемент таблицы страниц, и соответствующий элемент каталога таблиц содержат бит US = 1.

В МП Intel-386 ограничение типа доступа применяется только в адресном пространстве пользователя. Программа в режиме пользователя имеет право изменять только те страницы, для которых и соответствующий элемент таблицы страниц, и соответствующий элемент каталога таблиц содержат бит RW = 1. Если для какой-либо страницы элемент таблицы страниц или элемент каталога таблиц содержит бит RW = 0, то страница доступна только для чтения. В режиме супервизора все страницы доступны и для чтения, и для записи.

Когда используются оба вида защиты и на уровне страниц, и на уровне сегментов, процессор сначала выполняет проверки защиты сегментов, и только в случае успеха - проверки защиты страниц. Т.е. если доступ к памяти отвергается защитой на уровне сегментов, то будет сгенерировано нарушение общей защиты, а проверки защиты страниц выполняться не будут и дополнительно страничное нарушение не возникнет. Если проверки защиты сегментов проходят успешно, но нарушаются правила защиты страниц, генерируется страничное нарушение. Доступ к памяти предоставляется, только когда выполнены все правила и для сегментов, и для страниц.

В табл. 4.2 представлены сводные характеристики режимов IA-32.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 4.2. Сравнение режимов микропроцессоров IA-32 | | | | |
| **Характеристика** | **RM** | **PM, VM = 0** | **PM, VM = 1** | **SMM** |
| Формирование линейного адреса | без дескрипторных таблиц | через дескрипторную таблицу | без дескрипторных таблиц | без дескрипторных таблиц |
| Предел сегментов | 64 Кб | определяется дескриптором | 64 Кб | 4 Гб |
| Размер адреса/данных по умолчанию | 16 бит | определяется дескриптором | 16 бит | 16 бит |
| Максимальный объем доступной памяти (виртуальной) | ~1 Мб | ~64 Тб | ~1 Мб | ~4 Гб |
| Защита | нет1 | да | да | нет |
| Страничная трансляция | нет | да | да | нет |
| Многозадачность | нет | да | да | нет |
| Обработка прерываний | таблица векторов | дескрипторная таблица | дескрипторная таблица | нет2 |

**Вопросы для самоконтроля**

1. В каких режимах может работать IA-32?
2. Как формируется физический адрес при сегментной адресации?
3. Как формируется физический адрес при страничной адресации?
4. Что такое многозадачность? Какими средствами она поддерживается?
5. Какие правила на основе привилегий применяются для защиты сегментов кода, стека и данных?