Федеральное агентство по образованию

ГОУ ВПО «Уральский государственный технический университет – УПИ»

**В.П.Мокрецов**

**Микропроцессоры и МПС**

Часть 3

**Микропроцессоры IA-32.**

Конспект лекций.

Научный редактор доц., канд. техн. наук В.И.Паутов

Екатеринбург

2006

УДК 681.3

Рецензенты: Кафедра физики **Уральского государственного горного университета**; канд. техн. наук **А.А.Старостин**

(Институт теплофизики УрО РАН)

Автор: **В.П.Мокрецов**

**Микропроцессоры и МПС**

Часть 3

**Микропроцессоры IA-32.**

Конспект лекций./ В.П.Мокрецов.

Екатеринбург: УГТУ, «2006. 115 с.

Описывается микропроцессоры семейства INTEL P6/P7, основанные на популярной архитектуре IA-32. Рассматриваются архитектурные и структурные принципы процессоров Pentium, работа в реальном и защищенном режимах, способы адресации и система команд, реализация прерываний и исключений. Кратко описывается особенности архитектуры 64-разрядных микропроцессоров фирм Intel и AMD.

Предназначено для студентов специальностей 220201 -Управление и информатика в технических системах и 230101 – Вычислительные машины, комплексы, системы и сети.

Библиогр. 7 назв. Табл.19. Рис.39.

© Уральский государственный технический университет,

В.П. Мокрецов, 2006

**Предисловие**

Микропроцессоры фирм Intel и AMD, широко используются в современных персональных компьютерах, системах управления, телекоммуникационном оборудовании и многих других областях. В данном пособии подобрана информация по архитектуре микропроцессоров фирмы Intel семейств Р6 и Р7.

**1. Структура и функционирование процессоров Intel P6**

Процессоры семейства Р6, продолжая общую линию микропроцессоров Intel 80x86, имеют ряд архитектурных и структурных особенностей по сравнению с предыдущими моделями микропроцессоров фирмы Intel. Наиболее характерными из этих особенностей являются:

1. гарвардская структура с разделением потоков команд и данных с помощью введения отдельных внутренних блоков кэш-памяти для хранения команд и данных, а также шин для их передачи;
2. суперскалярная архитектура, обеспечивающая одновременное выполнение нескольких команд в параллельно работающих исполнительных устройствах;
3. динамическое исполнение команд, реализующее изменение последовательности команд (выполнение команд с опережением — спекулятивное выполнение), использование расширенного регистрового файла (переименование регистров), эффективное предсказание ветвлений;
4. двойная независимая шина, содержащая отдельную шину для обращения к кэш-памяти 2-го уровня (выполняется с тактовой частотой процессора) и системную шину для обращения к памяти и внешним устройствам (выполняется с тактовой частотой системной платы).

Процессоры семейства Р6 имеют следующие характеристики:

1. 32-разрядная внутренняя структура;
2. использование системной шины с 36 разрядами адреса и 64 разрядами данных;
3. раздельная внутренняя кэш-память 1-го уровня (L1) для команд и данных емкостью по 16 Кбайт;
4. поддержка общей кэш-памяти команд и данных 2-го уровня (L2) емкостью до 2 Мбайт;
5. конвейерное исполнение команд с реализацией 12 ступеней конвейера;
6. предсказание направления программного ветвления с высокой точно­стью;
7. ускоренное выполнение операций с плавающей точкой;
8. приоритетный контроль при обращении к памяти (защищенный режим);
9. поддержка реализации мультипроцессорных систем;
10. наличие внутренних средств, обеспечивающих самотестирование, отладку и мониторинг производительности.

Эти характеристики позволяют процессорам Р6 эффективно работать с разнообразным программным обеспечением под управлением операционных систем MS-DOS, Windows, OS/2 и ряда других. Исполняемый код для этих процессоров полностью совместим с кодом предыдущих моделей микропроцессоров семейства Intel 80x86 (8086, 8088, 80186, 80188, 80286, 80386, 80486, Pentium, Pentium MMX, Pentium Pro), поэтому для них может использовать­ся весь объем ранее разработанного программного обеспечения.

**1.1. Суперскалярная архитектура и организация конвейера команд**

Общая структура процессоров семейства Р6 показана на рис.1.1. Гарвардская внутренняя структура реализуется путем разделения потоков команд и данных, поступающих от системной шины через блок внешнего интерфейса в отдельные кэш-память команд и кэш-память данных, размещенные на кристалле процессора (кэш-память 1-го уровня, L1).

Блок внешнего интерфейса (BIU — Bus Interface Unit) реализует протоколы обмена процессора с системной шиной, к которой подключается память, контроллеры ввода/вывода, другие активные устройства системы, и шиной кэш-памяти второго уровня (L2), реализованной в виде отдельной микросхемы, размещаемой в общем корпусе (картридже) с микропроцессором. Обмен по системной шине осуществляется с помощью 64-разрядной двунаправленной шины данных, 41-разрядной шины адреса (тридцать три адресных линии А35-3 и восемь линий выбора байтов ВЕ7-0#) и ряда линий для передачи сигналов управления. Внутренняя кэш-память команд и данных 1-го уровня (L1) емкостью по 16 Кбайт каждая служит для размещения наиболее часто используемых команд и данных. Доступ к ним выполняется с частотой, соответствующей тактовой частоте процессора (сотни МГц). Благодаря этому существенно повышается производительность процессора за счет значительного сокращения числа обращений к внешней памяти, которые выполняются с частотой, определяемой возможностями системной шины (десятки МГц). Процессоры семейства Р6 содержат также кэш-память 2-го уровня (L2) емкостью 256 Кбайт, 512 Кбайт или 1 Мбайт, которая изготавливается на отдельном кристалле, но размещается в общем корпусе с процессором. Ее применение позволяет дополнительно повысить производительность. Процессор содержит блок выборки-декодирования, который выбирает из кэш-памяти команд 32 байта командного кода (строка кэш-памяти длиной 256 бит), затем производит разделение и декодирование команд. При этом из выбранных команд выделяются простые команды, для выполнения которых достаточно одной микрооперации, и сложные команды, требующие выполнения нескольких микроопераций. К числу простых относятся, например, команды сложения, вычитания, сравнения, логические операции и ряд других, использующие регистровую адресацию операндов.

Декодирование этих команд производят декодеры DC1, DC2, формирующие соответствующие микрокоманды.

Декодер DC3 декодирует сложные команды(например, сложение, вычитание, сравнение, логические операции с выборкой операнда из памяти), выполнение которых требует до четырех микроопераций. Соответствующие микрокоманды формируются на выходах этого декодера. Более сложные команды (умножение, деление, операции с плавающей точкой и ряд других) требуют формирования последовательности нескольких микрокоманд. Для их реализации используется устройство микропрограммного управления с ПЗУ микропрограмм, которое обеспечивает на выходе DC3 необходимую последовательность микрокоманд. Таким образом блок выборки декодирования формирует шесть потоков микрокоманд, обеспечивающих параллельное выполнение трех команд программы.

Если в потоке команд оказывается команда условного перехода (ветвления программы), то включается блок предсказания ветвления, который формирует адрес следующей выбираемой команды до того, как будет определено условие выполнения перехода.

После формирования потоков микрокоманд производится выделение регистров, необходимых для выполнения декодированных команд. Эта процедура реализуется блоком распределения регистров (RAT — Register Alias Table), который выделяет для каждого указанного в команде логического регистра (регистра целочисленных операндов ЕАХ, ЕСХ и др. или регистра операндов с плавающей точкой ST0-ST7) один из 40 физических регистров, входящих в блок регистров замещения (RRF — Retirement Reg-ister File). Эта процедура позволяет выполнять команды, использующие одни и те же логические регистры, одновременно или с изменением их последовательности.

Реализация наиболее эффективного выполнения потока декодированных команд обеспечивается блоком, который позволяет изменять последовательность команд (ROB — Re-Order Buffer). Этот блок реализован в виде буфера, в который поступают микрокоманды, реализующие выполнение ряда декодированных команд. Одновременно в буфере могут содержаться до 40 микрокоманд, которые затем направляются в исполнительные устройства по мере готовности операндов, поступающих из блока регистров RRF или выбираемых из памяти. При этом команды выполняются не в порядке их поступления, а по мере готовности соответствующих операндов и исполнительных устройств. В результате команды, поступившие позже, могут быть выполнены до ранее выбранных команд. Таким образом естественный порядок следования команд нарушается, чтобы обеспечить более полную загрузку параллельно работающих исполнительных устройств и повысить производительность процессора.

Микрокоманды поступают в исполнительные устройства через блок распределения (RS — Reservation Station), который направляет их в соответствующее устройство после его освобождения. Блок распределения имеет пять выходных портов и обеспечивает, в среднем, выполнение трех команд одновременно..

Суперскалярная архитектура реализуется путем организации исполнительного ядра процессора в виде ряда параллельно работающих блоков. Исполнительные блоки IU1, IU2 (IU — Integer Unit) производят обработку целочисленных операндов, блок FPU (Floating-Point Unit) выполняет операции над числами с плавающей точкой, блок ММХ реализует одновременную обработку нескольких упакованных символов, блок SSE, введенный в процессорах Pentium III, обеспечивает выполнение операций над потоком чисел с плавающей точкой. Один из целочисленных блоков выполняет также проверку условий ветвления для команд условных переходов и выдает сигналы перезагрузки конвейера команд в случае неправильно предсказанного ветвления.

Адреса операндов, выбираемых из памяти, вычисляются блоком MIU (Memory Interface Unite), который реализует интерфейс с кэш-памятью данных или внешней памятью. В его состав входят устройства генерации адреса AGU (Address Generated Unit), которые в соответствии с заданными в декодированных командах способами адресации одновременно формируют адреса двух операндов: один для операции чтения, второй — для операции записи. При этом MIU может формировать адреса и производить предварительное чтение операндов для команд, которые еще не поступили на выполнение. Такая процедура предварительного чтения данных для последующей их обработки в исполнительных блоках называется спекулятивной выборкой. Если команда, для которой проведена спекулятивная выборка операнда, не поступает на исполнение, то считанные данные теряются. Такой случай имеет место, например, для выбранных и декодированных команд, которые оказываются в нереализуемой ветви программы.

При выборке операнда из памяти производится обращение к кэш-памяти данных, которая имеет отдельные порты для чтения и записи. Таким образом обеспечивается одновременная выборка операндов для двух команд. При отсутствии адресуемого операнда в кэш-памяти данных (кэш-промах) с помощью буферного блока обращения к памяти MOB (Memory Order Buffer) осуществляется обращение к кэш-памяти 2-го уровня или основной памяти (по системной шине). Данный блок реализует спекуля­тивную выборку при чтении операнда.

Для промежуточного хранения данных, записываемых в основную память, в составе блока MOB имеется буфер данных записи. Эта буферная память емкостью 32 байта позволяет задержать запись результата до того момента, когда системная шина завершит выполнение других циклов обмена, например, выборки команд или чтения процессором новых операндов. Запись результатов производится, когда процессор выполняет команды, не требующие обращения к системной шине. Таким образом обеспечивается дополнительное повышение производительности системы. Отметим, что буферизация не выполняется при обращении процессора к устройствам ввода-вывода с помощью команды OUT. Вывод данных в этом случае реализуется в очередном цикле шинного обмена.

Обмен с основной памятью при использовании кэш-памяти производится с помощью пакетных циклов обращения, которые позволяют за один цикл переслать содержимое целой строки кэш-памяти (32 байта). Внешняя 64-разрядная шина данных позволяет выполнить такую пересылку за 5 тактов машинного времени: первый такт служит для установки адреса строки, а в течении следующих четырех тактов идет передача данных. При этом необходимые сигналы управления обменом устанавливаются только один раз (в начале цикла), а изменение младших разрядов адреса в тактах передачи осуществляется автоматически основной памятью.

При формировании адресов обеспечивается обращение к заданному сегменту памяти. Каждый сегмент может делиться на страницы, размещаемые в различных разделах адресного пространства. Блоки трансляции адреса обеспечивают формирование физических адресов команд и данных при использовании страничной организации памяти. При этом для сокращения времени трансляции используется внутренняя буферная память TLB (Translation Look-aside Buffer), которая хранит базовые адреса наиболее часто используемых страниц.

В процессорах Р6 реализован конвейер команд с 12 ступенями их выполнения. При прохождении семи первых ступеней (до блока изменения последовательности ROB) сохраняется исходный порядок следования команд, на трех исполнительных ступенях последовательность команд может быть нарушена, две заключительные ступени обеспечивают запись полученных результатов в память или регистры с восстановлением исходного порядка их следования. Такое восстановление выполняется буферным блоком MOB при записи результатов в память или блоками изменения последовательности и распределения команд (ROB и RS) при записи результатов в регистр (блок RRF).

Последовательная работа конвейера команд нарушается при поступлении команд условных переходов, так как в случае выполнения условия перехода, которое проверяется в исполнительном устройстве, потребуется перезагрузка конвейера — очистка всех предыдущих ступеней и выборка команды из новой ветви программы. Чтобы сократить или исключить потери времени, связанные с перезагрузкой длинного 12-ступенчатого конвейера, используется блок предсказания ветвлений. Его основной частью является ассоциативная память, называемая буфером адресов ветвлений (ВТВ — Branch Target Buffer), в которой хранятся 512 адресов ранее выполненных переходов. Кроме того ВТВ содержит четыре бита предыстории ветвления, которые указывают, выполнялся ли переход при четырех предыдущих выборках данной команды. При поступлении очередной команды условного перехода указанный в ней адрес сравнивается с содержимым ВТВ. Если этот адрес не содержится в ВТВ, то есть ранее не производились переходы по данному адресу, то предсказывается отсутствие ветвления. В этом случае продолжается выборка и декодирование команд, следующих за командой перехода. При совпадении указанного в команде адреса перехода с каким-либо из адресов, хранящихся в ВТВ, производится анализ предыстории. В процессе анализа определяется чаще всего реализуемое направление ветвления, а также выявляются чередующиеся переходы. Если предсказывается выполнение ветвления, то выбирается и загружается в конвейер команда, размещенная по предсказанному адресу. Одновременно в блоке выборки-декодирования сохраняется декодированная следующая команда. Если после анализа условия ветвления выясняется, что предсказание было неправильным, эта декодированная команда поступает из УУ в исполнительное устройство, обеспечивая сокращение потерь времени на перезагрузку конвейера.

Используемый алгоритм предсказания ветвлений ориентирован на достаточно частое повторение обращения к процедуре, которая обеспечивается определенной ветвью программы. При этом рекомендуется более часто используемые процедуры располагать в ветвях, следующих непосредственно за командой перехода, чтобы сократить время перезагрузки конвейера при ошибочно предсказанных ветвлениях. По имеющимся оценкам данный алгоритм обеспечивает вероятность правильного предсказания ветвлений на уровне 90% [1].

**1.2. Режимы работы процессора и организация памяти**

Процессоры Р6, как и предыдущие модели микропроцессоров Intel 80286, 80386, 80486, Pentium имеют три основных режима функционирования:

1. режим реальных адресов (реальный режим),
2. режим защищенных виртуальных адресов (защищенный режим),
3. режим системного управления.

В реальном режиме процессор Р6 работает как очень быстрый микропроцессор 8086, выполняющий обработку 16-разрядных операндов и адресующий 1 Мбайт оперативной памяти (20-разрядная шина адреса). При этом процессор реализует расширенный набор команд семейства Р6. Допускается также увеличение разрядности операндов и адресов до 32 с помощью префиксов, вводимых перед командами программы.

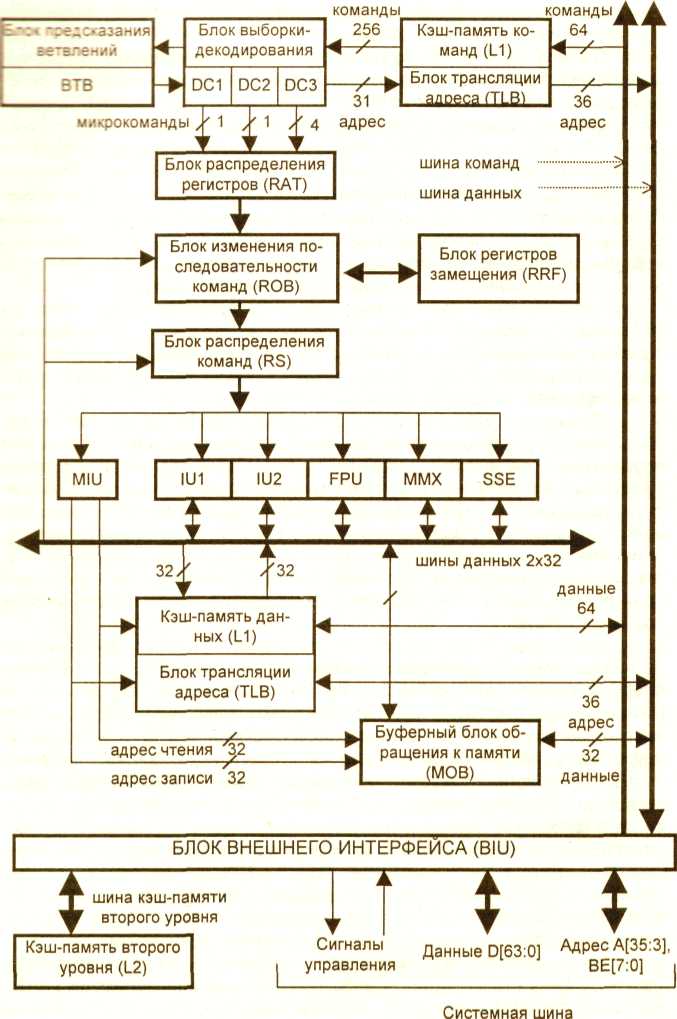


Рис 1.1. Общая структура процессоров семейства Intel P6

В защищенном режиме могут одновременно выполняться несколько отдельных задач (программ), которые защищены одна от другой и от операционной системы процессора. Специальный механизм обеспечивает переключение задач. В этом режиме процессор Р6 может также выполнять программы, написанные для микропроцессора 8086, если реализуется модификация защищенного режима — режим виртуального 8086. При работе в режиме виртуального 8086 процессор формирует 20-разрядный адрес, как и в реальном режиме, но может обрабатывать 32-разрядные операнды. В этом режиме обеспечивается, в случае необходимости, страничная организация памяти и защита системных программ, реализуемых процессором Р6, от пользовательских программ, выполняемых виртуальным процессором 8086 (двухуровневый механизм привилегий).

В процессорах Р6, как и в предыдущих моделях 80286, 80386, 80486, Pentium используется четыре уровня защищенности (0, 1, 2, 3), определяющих возможность доступа к различным сегментам памяти: уровень 0 — наибольшая защищенность сегмента, уровни 1, 2, 3 — более низкая степень защиты, которая уменьшается при возрастании номера уровня. Для каждой выполняемой программы (задачи) устанавливается один из четырех уровней привилегий, дающих право вызова сегментов: уровень 0 — максимальные привилегии, позволяющие вызывать сегменты с любым уровнем защищенности; уровень 3 — минимальные привилегии, допускающие вызов сегментов только с минимальным уровнем защищенности 3; уровни 1, 2 имеют, соответственно, промежуточные привилегии. Устанавливаемые с помощью этих уровней правила обращения обеспечивают эффективную защиту сегментов от несанкционированного доступа. Предусмотрены дополнительные возможности для управления доступом к устройствам ввода/вывода.

В защищенном режиме реализуется также многозадачное функционирование, когда процессор работает как несколько отдельных (виртуальных) процессоров, переключающихся под управлением операционной системы (ОС) на решение различных задач. При этом для управления процессором должна использоваться многозадачная ОС, например, Windows NT, которая осуществляет распределение во времени возможностей доступа к имеющимся ресурсам системы: памяти, устройствам ввода-вывода, обеспечивая наиболее эффективное выполнение нескольких заданий.

Режим системного управления (SMM — System Management Mode) используется для реализации специальных системных функций, например, для перевода системы в режим пониженного энергопотребления. При этом процессор обращается к отдельному адресному простран­ству, где размещается системное ОЗУ. Переход в режим системного управления осуществляется путем подачи специального внешнего сигнала прерывания SMI#=0, при этом процессор сохраняет в памяти контекст прерванной программы. В режиме SMM процессор функционирует как в реальном режиме с запрещенной обработкой прерываний. Выход из ре-жима SMM производится с помощью команды RSM, при этом восстанавливается контекст прерванной программы.

После включения питания или повторного запуска (процедура RESET) процессор начинает работу в реальном режиме. Переход процессора в защищенный режим реализуется с помощью команд LMSW или MOV CR0, которые выполняются только ядром операционной системы (программой, имеющей высший уровень привилегии 0). Эти команды уста­навливают в регистре управления CR0 значение бита разрешения защиты PЕ =1. Обратное переключение в реальный режим производится только командой MOV CR0, устанавливающей значение бита РЕ=0. При работе процессора в защищенном режиме переход в режим виртуального 8086 выполняется путем установки в регистре флагов EFLAGS значения бита VM=1. Установка этого значения бита или его сброс в состояние VM=0 (выход из режима виртуального 8086) производится с помощью команды возврата из прерывания IRET, выполняемой программой с высшим уровнем привилегий 0, или в процессе переключения задач.

Процессор оперирует с физической памятью объемом до 64 Гбайт. Каждый байт памяти имеет свой физический адрес — от 000000000h до FFFFFFFFFh. В памяти могут храниться 8-разрядные байты, 16-разрядные слова, 32-разрядные двойные слова и 64-разрядные счетверенные слова. При выполнении блоком SSE потоковых SIMD-команд используются 128-разрядные операнды. Слова занимают два смежных байта памяти, двойные и счетверенные слова — четыре и восемь смежных байт, причем младший байт располагается в ячейке с меньшим адресом, а старший в ячейке с большим адресом. Адресом слова служит адрес младшего байта..

Процессор выполняет обращение к памяти, использующей два способа организации — сегментация и разбиение на страницы.

Сегментация памяти обеспечивается при любом режиме работы процессора. Сегментация реализуется путем разбиения адресного пространства на отдельные блоки — сегменты, доступ к которым производится в соответствии с определенными правилами. Сегментированная память представляет собой набор блоков, характеризуемых определенными атрибутами — такими, как расположение, размер, тип (стек, программа, данные), характеристика защиты. В системе на основе процессора Р6 каждой задаче доступны до 8192 сегментов величиной до 4 Гбайт каждый. Таким образом обеспечивается возможность использования до 64 Тбайт виртуальной памяти. Для обращения к ячейке сегментированной памяти используется составной (логический) адрес, который состоит из селектора, задающего базовый адрес сегмента (начало), и относительного адреса ячейки (байта, слова) в сегменте. Арифметическое сложение базового и относительного адреса дает физический (линейный) адрес байта или слова (одинарного, двойного или счетверенного). Правила определения базового адреса сегмента зависят от режима работы микропроцессора — реальный или защищенный.

Страничная организация памяти обеспечивается только в защищенном режиме. Для ее реализации необходимо с помощью команды LMSW или MOV CR0 установить в регистре CR0 значение бита страничной адресации PG=1. При этом сегменты делятся на отдельные страницы емкостью 4 Кбайт или 4 Мбайт, размещаемые в различных позициях адресного пространства ОЗУ. Страницы могут группироваться в разделы. Размещение разделов и страниц в ОЗУ производится диспетчером памяти, входящим в состав операционной системы. Диспетчер задает базовые адреса разделов и страниц, которые хранятся в памяти в виде адресных таблиц. При использовании страничной организации блоки трансляции адреса, входящие в состав процессора, преобразуют адрес, сформиро­ванный MIU (при выборке операнда) или блоком выборки-декодирования (при выборке команды), в реальный физический адрес байта или слова. В процессе этого преобразования, который называется страничной транс­ляцией адреса, блок трансляции обращается к хранящимся в ОЗУ ад­ресным таблицам. Чтобы уменьшить число таких обращений и повысить производительность процессора, базовые адреса 32 страниц памяти ко­манд и данных, к которым выполнялись последние обращения, записыва­ются во внутренние буферы трансляции адреса (TLB). При обращении к этим страницам их базовые адреса считываются из буфера, поэтому фор­мирование физического адреса происходит существенно быстрее.

Страничная трансляция позволяет расширить объем адресуемой памяти до 64 Гбайт, используя 36-разрядную шину адреса. Такая возмож­ность реализуется, если установить в регистре управления CR4 значение бита РАЕ=1. При этом в адресных таблицах задаются 36-разрядные базовые адреса разделов и страниц, а размер страниц может составлять 4 Кбайт или 2 Мбайт.

**1.3. Регистры процессоров IA-32**

Регистровая модель процессоров IA-32 содержит набор регистров, большинство из которых входят в состав более ранних моделей семейства Intel 80x86/Pentium. Эти регистры составляют следующие группы.

Основные функциональные регистры:

• регистры общего назначения ЕАХ.ЕВХ, ЕСХ, EDX, ESP, EBP, ESI, EDI;

• указатель команд EIP;

• регистр флагов EFLAGS;

• регистры сегментов CS, SS, DS, ES, FS, GS.

Регистры блока обработки чисел с плавающей точкой (регистры FPU):

• регистры данных R7-0 (ST7-0);

• регистр тегов TW;

• регистр состояния FPSR;

• регистр управления FPCR;

• регистры-указатели команды и операнда FIP, FDP.

Регистры блока обработки пакетов чисел с плавающей точкой (регистры SSE)

• регистры пакетов данных ХММ 7-0;

• регистр управления-состояния MXCSR.

Системные регистры:

• регистры управления режимом CR4-0;

• регистры системных адресов GDTR, LDTR, IDTR, TR;

• регистры отладки DR7-0.

Служебные (модельно-специфические) регистры.

Регистры двух первых групп используются при выполнении прикладных программ, регистры третьей группы — при выполнении системных программ и отладке, регистры четвертой группы — при тестировании микропроцессора и контроле эффективности выполнения программ. Системные и служебные регистры доступны только программам с высшим уровнем привилегий 0.

Номенклатура и содержимое служебных (модельно-специфических) регистров MSR (Model-Specific Registers) определяется конкретной моделью: процессора. Запись и чтение их содержимого производится с помощью команд WRMSR и RDMSR. Обращение к некоторым из них может вызвать исключение, если эти регистры отсутствуют в данной модели. Регистры MSR используются в процессе отладки систем, содержат информацию о процессе выполнения программы.

**1.3.1. Основные функциональные регистры**

Состав основных функциональных регистров (рис. 1.2) идентичен для всех 32-разрядных микропроцессоров семейства Intel 80x86 и Pentium.

Восемь 32-разрядных регистров общего назначения ЕАХ, ЕВХ, ЕСХ, EDX, ESI, EDI, EBP, ESP предназначены для хранения данных и ад­ресов. Младшие 16 разрядов этих регистров доступны под именами АХ, ВХ, СХ, DX, SI, Dl, BP, SP. При операциях с байтами можно отдельно об­ращаться к младшему (разряды 7-0) или старшему (разряды 15-8) байту регистров АХ, ВХ, СХ, DX: младшие байты имеют имена AL, BL, CL, DL, старшие —АН, ВН, СН, DH.

Сегментные регистры CS, SS, DS, ES, FS, GS содержат 16-разрядные значения селекторов сегментов, определяющих адресуемый сегмент памяти. В реальном режиме содержимое этих регистров непо­средственно задает значение базового адреса соответствующего сегмента.



В защищенном режиме содержимое этих регистров задает выбор дескриптора, который содержит значение базового адреса и другие атрибуты сегмента. Соответствующий дескриптор выбира­ется из таблицы, хранящейся в ОЗУ, и размещается в программно-недоступном (теневом) регистре дескриптора сегмента (рис. 1.2).

Селектор в регистре CS обеспечивает обращение к сегменту ко­манд, селектор в SS — к сегменту стека, селекторы в DS, ES, FS, GS — к сегментам данных.

Указатель команд EIP представляет собой 32-разрядный регистр, содержимое которого используется в качестве смещения при определении адреса следующей выполняемой команды. Смещение задается относи­тельно базового адреса сегмента команд, задаваемого регистром CS. Младшие 16 бит EIP (биты 15-0) содержат 16-разрядный указатель команд с именем IP, который используется при 16-разрядной адресации. Со­держимое EIP (IP) изменяется при выполнении команд передачи управления и прерываний.

Регистр флагов EFLAGS (рис. 1.3) содержит ряд битов, которые имеют различное назначение: признаки состояния CF, PF, AF, ZF, SF, OF указывают определенные характеристики результата, полученного при выполнении команды; управляющий признак DF определяет порядок адресации операндов при выполнении последовательности команд обра­ботки строк символов; системные признаки TF, IF, IOPL, NT, RF, VM, AC, VIF, VIP, ID задают режим процессора при обслуживании исключений и прерываний, организации ввода-вывода данных, решении последователь­ности вызываемых задач и реализации ряда других процедур. Младшие шестнадцать бит регистра EFLAGS (биты 15-0) представляют 16-разрядный регистр флагов FLAGS, который используется выполнении программ, написанных для микропроцессоров 8086, 80186, 80286.

Признаки состояния в регистре EFLAGS имеют следующее значе­ние:

CF — признак переноса, принимает значение CF=1 при возникновении переноса из старшего разряда обрабатываемых операндов;

PF — признак четности, принимает значение PF=1, если младшие восемь разрядов результата содержат четное число единичных бит;

AF — признак полу-переноса, принимает значение AF=1, если при выпол­нении операции возникает перенос между младшими тетрадами (из разряда 3 в разряд 4) обрабатываемых операндов; используется при опе­рациях с операндами, представленными в двоично-десятичном коде (BCD);

ZF — признак нуля, принимает значение ZF=1 при получении нулевого результата операции;

SF — признак знака, принимает значение старшего (знакового) разряда результата операции (S=0 — положительное число, S=1 — отрицатель­ное);

OF — признак переполнения, принимает значение OF=1 в случае пере­полнения разрядной сетки при обработке операндов со знаком; значение признака для n-разрядных операндов определяется логическим выражением OF=Сn+Сn-1, где Сn — перенос, возникающий в старшем (n-1)-ом разряде (знаковом), Сn-1 — перенос, возникающий в предыдущем (n-2)-ом разряде;

Значение управляющего признака DF устанавливается пользова­телем и задает порядок обработки строк символов при выполнении соот­ветствующих команд:

DF — признак направления, при значении DF=0 вызывает автоматиче­ский инкремент содержимого индексных регистров ESI.EDI (SI.DI) после выполнения команды обработки символа, при DF=1 — декремент содер­жимого этих регистров; таким образом обеспечивается обработка симво­лов слева направо (от младших адресов к старшим) или в обратном на­правлении (от старших адресов к младшим).

Системные признаки (кроме NT) устанавливаются операционной системой, которая с их помощью задает определенные режимы выполне­ния ряда процедур:

TF — признак трассировки, при значении TF=1 процессор переключается в режим пошагового выполнения команд с реализацией после каждой ко­манды соответствующего прерывания;

IF — признак разрешения прерывания, установка значения IF=1 разре­шает обслуживание запроса прерывания, поступающего на внешний вход INTR;

IOPL — уровень привилегий ввода/вывода, задает максимальную ве­личину уровня привилегий текущей программы, при котором разрешается выполнение команд ввода/вывода;

NT — признак вложенной задачи, принимает значение NT=1 при пере­ключении процессора на выполнение другой задачи с помощью команды вызова CALL; используется для организации многозадачного режима;

RF — признак маскирования ошибок отладки, при установке значения RF=1 возможные ошибки отладки игнорируются при выполнении следую­щей команды; используется в процессе отладки программ;

VM — признак режима виртуального 8086, установка значения VM=1 вызывает переключение процессор в V-режим, при котором эмулируются возможности микропроцессора 8086;

АС — признак контроля выравнивания, установка значения АС=1 и бита АМ=1 в регистре CR0 вводит контроль выравнивания операндов при обращениях к памяти; в этом случае при обращении к невыравненному операнду (выборка слова по нечетному адресу или двойного слова по ад­ресу, не кратному четырем) реализуется исключение типа #АС (ошибка выравнивания);

VIF — признак разрешения обслуживания виртуального прерывания, установка значения VIF=1 разрешает обслуживание запросов виртуальных прерываний, поступление которых фиксируется установкой признака VIP=1;

VIP — признак запроса виртуального прерывания, значение VIP=1 ус­танавливается операционной системой в защищенном режиме, если внешний запрос прерывания поступает при выполнении программ пользо­вателя (низший уровень привилегии CPL=3) с запрещенным обслуживани­ем прерываний;

ID — признак допустимости идентификации, возможность программного изменения значения ID указывает, что для данного процессора обеспечи­вается выполнение команды идентификации CPUID.

**1.3.2. Регистры блока обработки чисел с плавающей точкой**

В состав блока FPU процессоров Р6 входят восемь регистров дан­ных R7-R0, регистр тегов TW, регистры управления FPCR и состояния FPSR, указатели команды FIP и данных FDP (рис. 1.4).

Регистры данных R7-0 содержат по 80 разрядов, разбитых на три поля: знак, порядок и мантисса, в соответствии с форматом представления чисел с плавающей точкой.

Набор этих регистров организован в виде кольцевого стека, верши­на которого определяется содержимым поля ТОР в регистре состояния FPSR (рис.1.5. При выполнении различных операций над содер­жимым регистров данных расположение вершины стека изменяется.



Регистр тегов TW содержит 16-разрядное слово, включающее во­семь двух битных тегов tag7-0 (рис. 1.5. Каждый тег (признак) характеризует содержимое соответствующего регистра данных R7-0, ука­зывая, является ли регистр пустым (незаполненным) или в нем размеща­ется конечное число, нуль или неопределенное значение (например, бес­конечность). Значение тега позволяет проверить содержимое регистра, не проводя анализ хранящихся в нем данных.

Регистр состояния FPSR хранит 16-разрядное слово состояния FPU (рис. 1.5, отдельные биты и поля которого имеют следую­щее назначение:

В — признак занятости, включен в состав слова состояния для совмести­мости с младшими моделями процессоров; значение В дублирует значе­ние ES — общего признака ошибки FPU;

ТОР — поле, указывающее вершину арифметического стека (рис. 1.4); со­держит номер регистра данных FPU, являющегося в данное время верхним в стеке;

СЗ-СО — признаки результата, значение которых характеризует результат выполнения команды FPU; возможные значения кода в разрядах СЗ-СО и их воздействие на работу процессора FPU рассмотрены в гл. 4; ES — общий признак ошибки, принимает значение ES=1, если установился хотя бы один из признаков ошибки операции FPU в шести младших разрядах FPSR; одновременно выдается сигнал FERR# = 0 на соответствующий внешний вывод процессора;

SF — признак переполнения стека, принимает значение SF=1 при наруше­нии нормальной работы арифметического стека; используется совместно с признаком С1: если SF=1, то значение С1=1 указывает на выход за верхнюю границу (переполнение) стека, а С1=0 — на выход за нижнюю границу стека.

Младшие шесть разрядов содержимого FPSR хранят признаки оши­бок, возникающих при выполнении команд FPU:

РЕ — признак нарушения точности;

UE — признак антипереполнения;

ОЕ — признак переполнения;

ZE — признак деления на нуль;

DE — признак ненормализованного операнда;

IE — признак неправильной операции.

Регистр управления FPCR (рис. 1.5) содержит в младшем байте биты РМ, UM, ОМ, ZM, DM, IM, маскирующие соответствующие признаки ошибок, фиксируемых в регистре состояния FPSR. При единичном значе­нии бита маски запрещается прерывание при возникновении соответст­вующей ошибки FPU. При нулевом значении бита маски установка соот­ветствующего признака ошибки FPU вызывает прерывание процессора.

Старший байт в регистре управления FPSR определяет режим округления и точность представления результатов вычислений: RC — поле управления округлением (табл. 1.1) определяет выбор одного из методов округления результата операций FPU;

PC — поле управления точностью (табл. 1.2) задает точность пред­ставления результатов арифметических операций FPU (см. разд. 4.4): одинарную (23 разряда мантиссы, 8 разрядов порядка), двойную (52 разряда мантиссы, 11 разрядов порядка) или расширенную (64 разряда мантиссы, 15 разрядов порядка).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 1.1. Управление округле­нием при операциях FPU | |  | Таблица 1.2. Точность представле­ния результата при операциях FPU | |
| RC | Метод округления | PC | Точность представления |
| 00 | К ближайшему числу | 00 | Одинарная |
| 01 | К минус бесконечности | 01 | (не используется) |
| 10 | К плюс бесконечности | 10 | Двойная |
| 11 | К нулю | 11 | Расширенная |

Поле PC учитывается только при выполнении команд FADD, FSUB, FDIV, FMUL, FSQRT. Для остальных команд FPU используется рас­ширенная точность.

Регистры — указатели команд FIP и данных FDP служат для идентификации команды, вызвавшей ошибку операции FPU. Содержимое этих регистров зависит от режима работы процессора. В реальном режиме при возникновении ошибки при выполнении команды FPU в эти регистры заносятся адрес и код операции данной команды, адрес использованного операнда. В защищенном режиме в них заносятся селекторы сегментов и относительные адреса команды и операнда. Эта информация использует­ся подпрограммой обработки прерываний для выяснения причины ошибки FPU.

**1.3.3. Системные регистры**

В состав этой группы регистров (рис. 1.6) входят регистры управления CR4 — CR0, регистры системных адресов GDTR, LDTR, IDTR, TR и регистры отладки DR7 — DR0. Эти регистры доступны только в за­щищенном режиме для программ, имеющих максимальный уровень привилегий 0.

Форматы содержимого регистров управления CR0, CR3, CR4 при­ведены на рис. 1.7. Регистр CR1 в процессорах Р6 и более ранних моделях не используется (резервирован для последующих моделей). CR0 — содержит 11 битов, определяющие режим работы процессора (рис.1.7,а):

РЕ — разрешение защиты, установка значения бита РЕ =1 переводит процессор защищенный режим; бит РЕ совместно с битом PG опреде­ляет режим работы процессора (табл.1.1);

МР — управление работой FPU, при установке значения МР=1 и пере­ключении задач (установка бита TS=1) выборка команды WAIT(FWAIT) вызывает исключение типа #NM (отсутствие FPU;

ЕМ — эмуляция FPU, при установке значения ЕМ=0 указывает на при­сутствие в системе блока FPU; установка значения ЕМ=0 вызывает при выборке команды FPU исключение типа #NM (отсутствие FPU), обес­печивающее обращение к подпрограмме, эмулирующей работу FPU;

TS — признак переключения задачи, принимает значение TS=1 при каждом переключении задачи и проверяется при поступлении команд FPU: если TS=1, то реализуется исключение типа #NM (отсутствие FPU);

ЕТ — в процессорах семейств Р6 и Pentium не используется (резерви­рован); в процессорах Intel 386, 486 установка значения ЕТ=1 указыва­ет на включение в систему математических сопроцессоров Intel 387DX;

CR2 — содержит 32-разрядный базовый адрес страницы, при обращении к которой зафиксирована ошибка при страничной адресации,

CR3 — содержит базовый адрес таблицы каталога страниц (20 старших битов), а также биты, управляющие загрузкой страниц в кэш-память дан­ных (рис. 1.7,б),

CR4 — содержит биты, обеспечивающие расширение функциональных возможностей, реализуемое в процессорах Pentium и Р6 (рис. 1.7,в).

NE — управление прерываниями FPU, при установке значения бита NE=1 ошибки, возникающие при выполнении команд FPU, вызывают ис­ключение типа #MF (ошибка FPU ) при выборке следующей команды FPU; при установке значения NE=0 ошибка FPU вызывает выдачу на соответствующий вывод процессора сигнала FERR#=0, который посту­пает во внешний контроллер прерываний (этот способ обслуживания пре­рываний FPU реализуется в персональных компьютерах);

WP — защита от записи, при установке значения бита WP=1 программам супервизора, имеющим уровень привилегий 0-2, запрещается запись на страницы пользователя, имеющие уровень привилегии 3; при значении WP=0 такая запись разрешена;





AM — управление проверкой выравнивания, разрешает при установке значения бита АМ=1 проверку выравнивания адресов по границам слов (кратные двум) или двойных слов (кратные четырем) выборке операндов из памяти, если в регистре EFLAGS установлено значение бита АС=1 и выполняемая программа имеет уровень привилегии 3 (программа пользо­вателя);

NW — запрещение записи в кэш-память, совместно с битом CD опреде­ляет режим работы внутренней кэш-памяти процессора;

CD- запрещение заполнения кэш-памяти, совместно с битом NW опре­деляет режим работы внутренней кэш-памяти процессора;

PG — разрешение страничной трансляции адреса, при установке зна­чения PG=1 обеспечивает страничную адресацию памяти в защищенном режиме (см. табл. 1.3).

Таблица 1.3. Режимы работы процессора

|  |  |  |
| --- | --- | --- |
| PG | РЕ | Режим процессора |
| 0 | 0 | Реальный режим |
| 0 | 1 | Защищенный режим без использования страничной адресации |
| 1 | 0 | Запрещенная комбинация (вызывает исключение типа #GP-нарушение защиты) |
| 1 | 1 | Защищенный режим с использованием страничной адресации |

Отметим, что биты МР, ЕМ, TS влияют также на выполнение ко­манд, реализуемых блоком ММХ. В табл. 1.4 указано, как реагирует процессор на поступление команд FPU или ММХ в зависимости от установленного значения этих битов.

Таблица 1.4. Режимы работы блоков FPU и ММХ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Биты CR0 | | | Выполняемые команды | | |
| ЕМ | МР | TS | Команды FPU | Команда WAIT (FWAIT) | Команды ММХ |
| 0 | 0 | 0 | Выполняется | Выполняется | Выполняется |
| 0 | 0 | 1 | Исключение #NM | Выполняется | Исключение #NM |
| 0 | 1 | 0 | Выполняется | Выполняется | Выполняется |
| 0 | 1 | 1 | Исключение #NM | Исключение #NM | Исключение #NM |
| 1 | 0 | 0 | Исключение #NM | Выполняется | Исключение #UD |
| 1 | 0 | 1 | Исключение #NM | Выполняется | Исключение #UD |
| 1 | 1 | 0 | Исключение #NM | Выполняется | Исключение #UD |
| 1 | 1 | 1 | Исключение #NM | Исключение #NM | Исключение #UD |

Младшие 16 разрядов регистра CR0 для совместимости с защи­щенным режимом микропроцессора 80286 называют словом состояния машины MSW (Machine State Word). Команды загрузки и сохранения LMSW, SMSW для совместимости с микропроцессором 80286 работают только с младшими 16 разрядами регистра CR0. Для за­грузки или сохранения всего содержимого регистра CR0 используются MOV CRO, r/m или MOV r/m, CR0, выполняемые программами, имеющими максимальный уровень привилегии 0.

Регистр CR3 (рис. 1.7,6) содержит 20 старших разрядов базового адреса таблицы разделов, который используется при трансляции адреса в случае страничной организации памяти. Кроме того в этом регистре содержатся два бита, управляющих кэш-памятью при страничной адресации:

PCD — запрещает при установке значения PCD=1 загрузку содержимого страницы в кэш-память; при значении PCD=0 такая загрузка разрешена;

PWT — определяет режим работы кэш-памяти при страничной адресации: при установке значения PWT=1 реализуется режим сквозной записи (write-through), при установке PWT=0 реализуется режим обратной записи (write-back).

Следует отметить, что биты PCD, PWT влияют на работу кэш­памяти только при страничной адресации (когда в регистре CR0 установ­лены значения PE=PG=1). В этом случае они определяют режим работы как внутренней кэш-памяти (L1), так и внешней — 2-го уровня (L2).

В регистре CR4, который введен в процессорах Pentium, P6, содер­жатся девять управляющих битов, обеспечивающих расширение архитек­турных возможностей этих процессоров:

VME — разрешает при установке значения VME=1 реализацию виртуаль­ных прерываний в режиме виртуального 8086;

PVI — разрешает при установке значения VME=1 реализацию виртуальных прерываний в защищенном режиме;

TSD — разрешает при TSD=1 выполнение команды RDTSC (чтение со­держимого таймера реального времени) только программам с максималь­ным уровнем приоритета 0, при TSD=0 эта команда может выполняться программой с любым уровнем приоритета;

DE – при установке бита вызывает исключение #UD в случае обращения к регистрам отладки DR4.DR5, которые отсутствуют в процес­сорах Pentium, P6; при DE=0 при обращении к эти регистрам реализуется обращение к регистрам DR6, DR7;

PSE — расширяет размер адресуемых страниц до 4 Мбайт при установке значения PSE=1, при значении PSE=0 сохраняется размер страниц 4 Кбайт;

РАЕ — обеспечивает расширение разрядности физического адреса до 36 бит при установке значения РАЕ=1;

МСЕ — разрешает реализацию исключения машинного контроля #МС при установке значения МС=1;

PGE — позволяет при значении PGE=1 определять некоторые страницы (часто используемые или используемые несколькими процессорами) как глобальные (путем установки бита глобальности G=1 в указателе страни­цы или раздела; при очистке буфера страничной трансляции TLB командой MOV CR3 адреса глобальных страниц сохраняются;

РСЕ — разрешает при установке значения РСЕ=1 выполнение команды RDPMC (чтение содержимого счетчиков, характеризующих эффективность работы процессора, программой с любым уровнем привиле­гий; при значении РСЕ=1 эта команда может выполняться только програм­мой с максимальным уровнем привилегий 0.

OSFXCR — разрешает при установке значения OSFXCR=1 выполнение команд FXSAVE/FXRSTOR, используемых для сохране­ния/восстановления состояния регистров блоков FPU), MMX, SSE при пе­реключении задач.

Регистры системных адресов GDTR, IDTR, LDTR. TR (рис. 1.6) слу­жат для обращения к таблицам и сегментам, с помощью которых осущест­вляется адресация памяти в защищенном режиме:

GDT — таблица глобальных дескрипторов,

IDT — таблица дескрипторов прерываний,

LDT — таблица локальных дескрипторов,

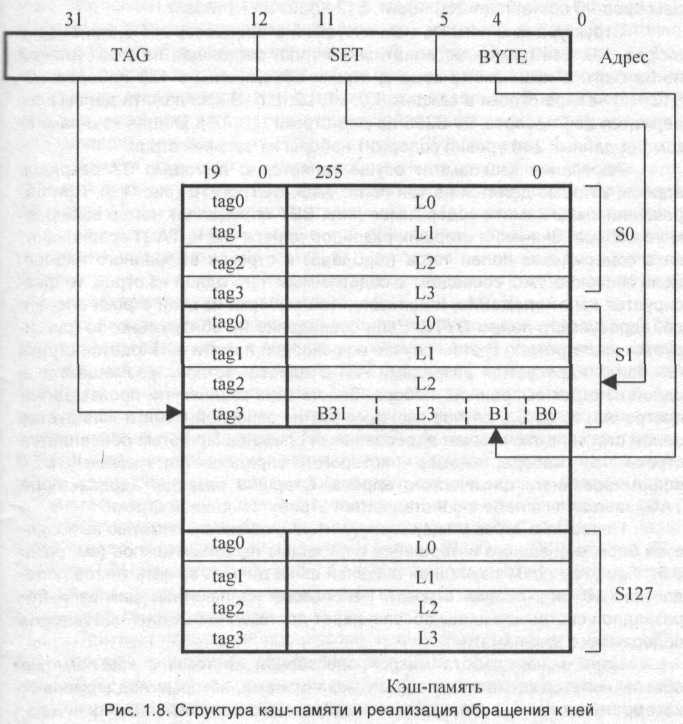
TSS — сегмент состояния задачи.

Регистр таблицы глобальных дескрипторов GDTR и регистр табли­цы дескрипторов прерываний IDTR содержат 32-разрядные базовые адреса и 16-разрядные размеры таблиц GDT и IDT. Эти таблицы являются общими для всех задач. В регистр таблицы локальных дескрипторов LDTR и регистр задачи TR заносятся 16-разрядные селекторы, позволяющие обратиться к таблице LDT и сегменту TSS, которые определены для каж­дой отдельной задачи. С каждым из этих регистров связан программно недоступный регистр дескриптора сегмента.

Процессоры Р6 содержат также восемь 32-разрядных регистров от­ладки DR7-0.

**1.4. Внутренняя кэш-память**

Кэш-память представляет собой быстродействующую буферную память ограниченного объема, которая располагается между процессором и относительно медленной основной памятью. В процессе работы отдель­ные блоки информации копируются из основной памяти (ОЗУ) в кэш­-память. Процедура загрузки информации из ОЗУ в кэш-память называется кэширование. Когда процессор обращается за командой или данными, то сначала проверяется их наличие в кэш-памяти. Если необходимая ин­формация находится там, то она быстро извлекается, так как обращение к кэш-памяти производится с тактовой частотой процессора. Такой случай обращения называют кэш-попаданием. Если необходимая информация в кэш-памяти отсутствует, то она выбирается из основной памяти и одно­временно заносится в кэш-память. Такой случай называют кэш-промахом.



Повышение быстродействия вычислительной системы достигается в том случае, когда кэш-попадания реализуются намного чаще, чем кэш-промахи. Высокий процент кэш-попаданий обеспечивается благодаря то­му, что в большинстве случаев программы обращаются к ячейкам памяти, расположенным вблизи от ранее использованных. Это свойство, которое называют локальностью программ, обеспечивает эффективность исполь­зования кэш-памяти.

В процессорах Р6 используется отдельная внутренняя кэш-память для хранения команд и данных, каждая из которых имеет объем 16 Кбайт. Кэш-память имеет несколько режимов работы, обеспечивающих гибкость при выполнении программ и в процессе их отладки. Отдельные области памяти могут быть определены программным обеспечением или внешней аппаратурой как не подлежащие кэшированию (загрузке в кэш-память).

Помимо внутренней кэш-памяти (уровень L1) в системах, реализуе­мых на базе процессоров Р6 используется дополнительная внешняя кэш-­памяти (уровень L2), которая размещается на отдельном кристалле, но монтируется в общем корпусе с процессорным кристаллом. Для обращения к этой кэш-памяти служит отдельная шина, подключаемая к блоку внешнего интерфейса (рис.1.1), которая работает с тактовой частотой процессора. Емкость кэш-памяти уровня L2 для различных моделей про­цессоров Р6 составляет 256 Кбайт, 512 Кбайт или 1 Мбайт.

Структура кэш-памяти, используемой в процессорах Р6, приведена на рис. 1.8. Эта память ассоциативного типа, состоящая из строк длиной по 32 байт. В кэш-памяти команд строки объединены в 128 наборов S0-S127 по четыре строки в каждом (LO, L1, L2, L3). В кэш-памяти данных со­держится 256 наборов S0-S255 по две строки (LO, L1). Общая кэш-память команд-данных 2-го уровня содержит наборы из четырех строк.

Адресация кэш-памяти осуществляется с помощью 32 разрядов адреса, которые делятся на три поля: TAG, SET, BYTE (рис. 1.8). При об­ращении к кэш-памяти содержимое поля SET определяет номер выбирае­мого набора. Значения старших разрядов адреса (поле TAG) сравнивают­ся с содержимым полей тегов (tag0-tag3) в строках выбранного набора. Если значение TAG совпадает с содержимым тега одной из строк, то фик­сируется кэш-попадание, и производится выборка из этой строки операн­да, адресуемого полем BYTE. Если совпадение не обнаружено, то фикси­руется кэш-промах. В этом случае из основной памяти выбирается строка (32 байт), адресуемая разрядами А31-5 адреса, которая размещается в одной из строк выбранного набора. Заполнение кэш-памяти производится построчно, то есть в случае кэш-промаха из основной памяти копируется целая строка, включающая адресованный операнд. При этом обновляется строка в наборе, номер которого определяется полем SET сформированного физического адреса. Старшие разряды адреса (поле TAG) заносятся в поле tag и становится атрибутом данной строки.

Пересылка строк между кэш-памятью и основной памятью выполня­ется блоком внешнего интерфейса с помощью пакетных циклов. При этом по 64-разрядной внешней шине данных за пять тактов пере­дается 256 бит (полная строка). Использование пакетных циклов и 64-разрядной системной шины обеспечивает достаточно быстрое обновление содержимого кэш-памяти.

Нормальная работа микропроцессорной системы с кэш-памятью обеспечивается соответствующими механизмами, которые поддерживают когерентность — соответствие содержимого основной памяти и кэш­-памяти. Это соответствие достигается с помощью механизмов сквозной записи (Write-Through) или обратной записи (Write-Back).

При сквозной записи выполняется одновременное изменение со­держимого кэш-памяти и основной памяти. Таким образом, при любом цикле записи, даже в случаев кэш-попадания, производится обращение к основной памяти по системной шине. Данное обстоятельство существенно снижает производительность системы, так как циклы обращения по этой шине выполняются с тактовой частотой системной платы, которая значи­тельно ниже тактовой частоты процессора.

При обратной записи изменение содержимого строки кэш-памяти вызывает установку признака модификации (состояние М). При обновлении содержимого кэш-памяти ее строки, находящиеся в состоянии М (имеющие установленный признак модификации), переписываются обратно в основную память. Таким образом, обращение к основной памяти в циклах записи производится только в случае кэш-промаха. При исполь­зовании кэш-памяти значительной емкости процессор может достаточно долго работать без обращения к основной памяти. Этот механизм наибо­лее часто используется современными процессорами, хотя многие из них могут реализовать как обратную, так и сквозную запись. В процессорах Р6 изменить механизм записи при страничной адресации можно путем соот­ветствующей установки бита PWT в регистре CR3.

Каждая кэш-память имеет собственную служебную память, где хранятся биты, характеризующие содержимое ее строк: биты обращения B0 (если наборы содержат по 2 строки) или В0-В2 (если наборы содержат по четыре строки), используемые при обновлении строки набора в соот­ветствии с алгоритмом LRU (биты LRU), биты состояния S0-S1, с помощью которых обеспечивается когерентность содержимого строки в кэш-памяти и основной памяти согласно протоколу MESI (биты MESI).

Если при чтении произошел кэш-промах и кэширование данной об­ласти памяти разрешено, то производится обновление содержимого одной из строк набора, заданного полем SET адреса, в которую вводится ин­формация из основной памяти. Новая информация замещает ранее имевшуюся в данной строке. В первую очередь замещается содержимое строки, отмеченной с помощью битов MESI как недостоверная (состояние I). Если в данном наборе нет недостоверной ("пустой") строки, то произво­дится удаление из кэш-памяти содержимого строки, которая дольше всего оставалась невостребованной. Используемый алгоритм обновления назы­вается "замещение менее используемой информации" (LRU — least recently used) и реализуется с помощью битов LRU. Один (ВО) или три (ВО-В2) бита LRU хранятся в служебной памяти для каждого из наборов строк кэш-памяти. Их содержимое определяет выбор строки, замещаемой в дан­ном наборе при кэш-промахе.

При инициализации процессора и очистке кэш-памяти биты LRU всех наборов кэш-памяти сбрасываются в 0. В ходе работы процессора биты В0-В2 принимают следующие значения в соответствии с тем, к каким из строк набора производились последние обращения. Если в данном на­боре последнее обращение выполнялось к одной из пары строк L0 или L1, устанавливается В0=1, если к L3 или L4, то устанавливается В0=0. Биты 81 и В2 указывают, к какой из строк пары L1-L2 и L3-L4 осуществлялось последнее обращение. Значение В1=0 устанавливается при обращении к L0, значение В1 = 1 — при обращении к L1. Соответственно, значение В2=0 или 1 устанавливается при обращении к строке L2 или L3. В соответствии с алгоритмом LRU для замены выбирается пара строк, к которым не про­изводилось последнее обращение, а в этой паре замене подлежит строка, к которой не производилось последнее обращение. Выбранная таким об­разом строка считается дольше всего неиспользуемой в данном наборе, поэтому ее содержимое замещается выбираемыми из основной памяти новыми 32 байтами.

Данное описание соответствует использованию наборов из четырех строк, что имеет место в кэш-памяти команд и общей кэш-памяти 2-го уровня. Для кэш-памяти данных, содержащей две строки в наборе, каждый набор имеет всего один бит LRU, который принимает значение В0=0 при обращении к строке L0, и В0=1 при обращении к L1.

При использовании обратной записи в процессорах Р6 когерентность содержимого основной памяти и кэш-памяти обеспечивается с по­мощью протокола MESI, который устанавливает четыре возможных со­стояния строки кэш-памяти:

М (modified) — модифицированное путем записи содержимое строки, которое имеется только в кэш-памяти данного процессора;

Е (exclusive) — не модифицированное содержимое строки, которое содержится только в кэш-памяти данного процессора и основной памя­ти;

S (shared) — содержимое строки может также находиться в кэш­-памяти других активных устройств (процессоров), входящих в состав системы;

I (invalid) — недействительное (аннулированное) содержимое строки.

Текущее состояние строки определяется значениями битов SO, S1, которые автоматически устанавливаются и изменяются в процессе работы процессора. При инициализации процессора или очистке (аннулирование содержимого) кэш-памяти для всех строк устанавливается состояние I, указывающее, что строки не заполнены. Дальнейшее функционирование процессора вызывает заполнение кэш-памяти. При этом для заполненной строки устанавливается состояние Е. Если в строку производится запись данных, то для нее устанавливается состояние М. Аннулирование содержи­мого всех строк (освобождение кэш-памяти) производится программно с помощью команд INVD и WBINVD. Команда INVD аннулирует содержимое внутренней кэш-памяти команд и данных, устанавливая состояние I для всех строк. Команда WBINVD перед аннулированием производит обратную запись (write-back) в основную память содержимого тех строк кэш-памяти данных, для которых установлено состояние М (в них была проведена мо­дификация). При выполнении этих команд процессор выдает внешние сиг­налы для реализации аналогичных процедур в кэш-памяти 2-го уровня. Данные команды являются привилегированными — их выполнение в за­щищенном режиме разрешается только программам с максимальным уровнем привилегии 0.

Если выполняется захват шины другим процессором (активным уст­ройством) в мультипроцессорной системе, то отключаемый от шины про­цессор продолжает выполнение команд и обработку данных, содержащих­ся в кэш-памяти. Активный процессор может обратиться к данным, копии которых уже находятся в отключенном от шине процессора. Поэтому необ­ходимо принять меры для обеспечения идентичности копий данных, ис­пользуемых различными процессорами. Такие меры реализуются с помощью специальных средств, выполняющих снупинг (snoop) — слежение за содержимым кэш-памяти разных процессоров, входящих в систему. При этом отключенный от внешней шины процессор принимает адреса, выда­ваемые на адресные линии активным процессором. Если этот адрес сов­падает с адресом данных, хранящихся в одной из заполненных строк кэш­-памяти данных отключенного процессора, то выполняются следующие процедуры. Если активный процессор производил запись по данному ад­ресу, то содержимое соответствующей строки в отключенном от шине про­цессоре аннулируется (устанавливается состояние I). Если активный про­цессор производил чтение по адресу строки, имеющей состояние М, то отключенный процессор выполняет специальный цикл выдачи адресован­ных данных на внешнюю шину, чтобы активный процессор получил их мо­дифицированный вариант.

Когда отключенный процессор снова становится активным, то есть принимает на себя управление системной шиной, то для заполненных строк его кэш-памяти данных устанавливается состояние S. Оно показы­вает, что содержимое строки может находиться в кэш-памяти данных дру­гого процессора, который был активным ранее. Запись данных в строку, имеющую состояние S, вызовет также их запись в основную память, после чего строка устанавливается в состояние Е.

Отметим, что описанный протокол MESI, обеспечивающий коге­рентность содержимого основной и кэш-памяти, действует только при об­ращениях к внутренней кэш-памяти данных и общей кэш-памяти 2-го уров­ня. Во внутренней кэш-памяти команд процедура записи (модификации) не реализуется, поэтому при обращении к ней когерентность сохраняется бея использования дополнительных средств.

Управление функционированием кэш-памяти осуществляется бита­ми CD и NW регистра CRO (рис.1.7,а), которые определяют три возмож­ных режима работы в соответствии с табл.1.5.

Таблица 1.5. Режимы работы кэш-памяти

|  |  |  |
| --- | --- | --- |
| CD | NW | Режимы работы кэш-памяти |
| 0 | 0 | Нормальное функционирование кэш-памяти |
| 0 | 1 | Запрещенная комбинация, ее установка вызывает исклю­чение типа #GP (нарушение защиты) |
| 1 | 0 | Заполнение кэш-памяти запрещено. Поддерживается когерентность с основной памятью. |
| 1 | 1 | Заполнение кэш-памяти запрещено. Не поддерживается когерентность с основной памятью. |

При нормальном функционировании кэш-памяти ( биты CD=NW=0) кэш-попадание вызывает чтение данных из кэша или их запись в кэш. Кэш-промах при чтении приведет к заполнению соответствующей строки кэша из основной памяти. Запись в основную память выполняется при кэш-промахе, а также в случаях кэш-попадания при записи в строку, имеющую S-состояние (для обеспечения когерентности в мультипроцессорных сис­темах).

При запрещенном заполнении кэш-памяти ( бит CD=1) ее содержи­мое сохраняется и может использоваться. В этом режиме кэш-память мо­жет служить в качестве быстрого статического ОЗУ, если предварительно загрузить в нее содержимое определенной области основной памяти. При кэш-попадании выполняется чтение данных из кэша или их запись в кэш. Кэш-промах при чтении или записи вызывает обращение к основной памя­ти для выборки операнда, но изменение содержимого кэша (заполнение строки) не производится. Если установлено значение бита NW=0. то кэш-попадание при записи в строку, имеющую S-состояние, вызовет также за­пись данных в основную память для обеспечения когерентности. Если ус­тановлено значение NW=1, то при кэш-попадании в строку, имеющую S-состояние, запись в основную память не производится, то есть когерент­ность не поддерживается. В этом случае строка, в которой зафиксировано кэш-попадание сохраняет состояние S.

Чтобы полностью исключить использование кэш-памяти обоих уровней (L1 и L2) в процессоре Р6 необходимо выполнить команду WBINVD и затем установить в регистре CR0 значения битов CD=NW=1. Обычно использование кэш-памяти запрещают в процессе отладки, чтобы по состоянию системной шины можно было контролировать каждый цикл функционирования системы.

При использовании страничной адресации памяти управление кэ­шированием осуществляется с помощью битов PCD, PWT в регистре CR3 и бита PGE в регистре CR4. Кроме того, используются биты G, PCD, PWT, которые содержатся в указателях разделов и страниц, хранящихся в ос­новной памяти в виде специальных таблиц. Эти биты устанавливают раз­личные режимы кэширования для отдельных разделов и страниц памяти.

В архитектуру процессоров Р6 введены специальные регистры MTRR (Memory Type Range Registers), которые устанавливают режимы кэширования для определенных адресных зон памяти. Эти регистры вхо­дят в состав модельно-специфических регистров (MSR). Они определяют для 96 адресных зон фиксированного или произвольного размера сле­дующие режимы кэширования:

UC (Uncacheable, код 0) — некэшируемая зона, все обращения для чте­ния и записи поступают на системную шину и выполняются в соответствии с порядком следования команд (без изменения их последовательности и без буферизации записываемых данных), спекулятивная выборка операн­дов не реализуется;

WC (Write Combining, код 1) — некэшируемая зона с объединением дан­ных при записи; все обращения выполняются как для некэшируемой зоны, но допускается спекулятивная выборка при чтении операндов и буфериза­ция данных при их записи в память;

WT (Write Through, код 4) — кэшируемая зона с использованием меха­низма сквозной записи; кэш-промах при записи не вызывает заполнения строки кэша, кэш-попадание при записи переводит соответствующую строку в1 I-состояние (освобождение строки); допускается спекулятивная вы­борка при чтении и буферизация данных при записи; WB (Write Back, код 6) — кэшируемая зона с использованием механизма обратной записи; кэш-промах при записи вызывает заполнение строки кэ­ша, кэш-попадание при записи изменяет содержимое соответствующей строки в кэше с установкой ее в М-состояние, не вызывая обращение к основной памяти; допускается спекулятивная выборка при чтении и буфе­ризация данных при записи;

WP (Write Protected, код 5) — кэшируемая зона для чтения, некэшируе­мая зона для записи; при чтении реализуется обычная процедура обраще­ния к кэшу; при записи выполняется цикл обращения к системной шине без изменения содержимого кэша, причем во всех процессорах системы стро­ка кэша, адресованная при записи (если такая имеется), устанавливается в l-состояние (освобождается); допускается спекулятивная выборка при чтении, буферизация данных при записи не выполняется.

Таким образом, архитектура Р6 позволяет реализовать большой набор вариантов использования кэш-памяти и различных разделов основ­ной памяти, что обеспечивает возможности эффективного применения процессоров для решения широкого круга задач с помощью разнообраз­ных типов современной памяти и устройств ввода-вывода.

**1. 5. Форматы команд и способы адресации**

Набор команд, реализуемый процессорами Р6, обеспечивает вы­полнение операций над операндами, которые находятся в регистре, памя­ти или непосредственно в команде. В набор входят безадресные, одно- и двухадресные команды. Процессор реализует следующие шесть типов двухадресных команд:

• регистр — регистр;

• память — регистр;

• непосредственные данные — регистр;

• регистр — память;

• память — память;

• непосредственные данные — память.

Операнды могут содержать 8, 16 или 32 разряда. Для реализации различных типов команд определены форматы, задающие порядок размещения информации о выполняемой операции и способах выбора операндов.

Общий формат команды (рис. 1. 11) содержит следующие поля:

ОРС — код операции;

MODR/M, SIB — байты адресации;

DISP — байты смещения;

IMM — непосредственно заданный операнд.

Для конкретной команды отдельные поля могут иметь различное число байт или вообще отсутствовать. Поэтому команды могут содержать от 1 до 12 байт. Перед кодом операции в ряде случаев вводятся один или несколько префиксных байтов, модифицирующих выполняемую команду.

Код операции ОРС (operation code) занимает 1 или 2 байта Во мно­гих командах пересылок, а также в логических и арифметических командах первый байт ОРС содержит бит w, значение которого определяет разрядность операндов: w=0 — операция с байтами; w=1 — операция со словами (16 или 32 разряда). Разрядность слов (16 или 32 разряда) определяется режимом работы процессора. В реальном режиме и режиме виртуального 8086 по умолчанию используются 16-разрядные слова. В защищенном режиме разрядность устанавливается значением бита D в дескрипторе сегмента команд. При выполнении отдельных команд разрядность операндов может меняться соответствующим префиксом (см. табл. 1. 14).



В ряде команд первой байт ОРС содержит поля reg или sreg, определяющие выбор используемых регистров. Трех битовое поле reg за­дает выбираемый регистр в соответствии с разрядностью обрабатываемых операндов (табл. 1. 6). Поле sreg определяет выбор сег­ментных регистров (табл. 1. 7). Двух битовые коды sreg (указаны в скобках) используются для выборки регистров CS, SS, DS, ES в программах, напи­санных для микропроцессоров 8086, в которых регистры FS, GS отсутство­вали.

Байт адресации MODR/M содержит три поля (рис. 1. 12). Поля MOD и R/M задают адрес одного из операндов, который может храниться в регистре или ячейке памяти. Кодировка этих полей определяет вы­бираемый способ адресации.

В одноадресных командах поле REG/OPC содержит дополнитель­ные биты кода операции. В двухадресных командах поле REG содержит код регистра, в котором хранится второй из операндов. Тип команды (од­но- или двухадресная) определяется первым битом ОРС. При этом в ОРС содержится бит d, который задает выбор регистров, используемых в каче­стве источника и приемника информации при выполнении ряда двух­адресных арифметических и логических операций типа регистр-регистр:

. d=0 — код источника содержится в поле REG/OPC, код приемника в поле R/M;

. d=1 — код источника содержится в поле R/M, код приемника в поле REG/OPC.

Кодировка регистров указана в таблицах 1. 6, 1. 7.

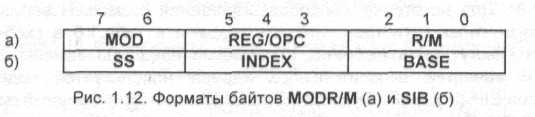
Для реализации ряда способов адресации используется байт SIB. Он содержит 3-битные поля INDEX и BASE, определяющие выбор регистров, используемых в качестве индексного и базового регистров, и поле SS, задающее масштабный коэффициент для модификации значения индекса (рис. 1. 12). Правила формирования адреса при использовании байта SIB изложены ниже.

Таблица 1. 6. Кодировка регистров общего назначения

|  |  |  |  |
| --- | --- | --- | --- |
| Поле REG | Разрядность операндов | | |
| 8 | 16 | 32 |
| 000 | AL | АХ | ЕАХ |
| 001 | CL | СХ | ЕСХ |
| 010 | DL | DX | EDX |
| 011 | BL | ВХ | ЕВХ |
| 100 | АН | SP | ESP |
| 101 | СН | ВР | ЕВР |
| 110 | DH | SI | ESI |
| 111 | ВН | DI | EDI |

Таблица 1. 7. Кодировка сегментных регистров

|  |  |
| --- | --- |
| Поле sreg | Сегментный регистр |
| 000 (00) | ES |
| 001 (01) | CS |
| 010 (10) | SS |
| 011 (11) | DS |
| 100 | FS |
| 101 | GS |



Если поле MOD байта MODR/M имеет значение 00 (при некоторых значениях R/M) или 01, 10, то для формирования адреса используется 8-, 16- или 32-разрядное смещение (табл. 1. 8, 1. 9). Это смещение задается соответствующими байтами в поле команды, которые располагаются по­сле байтов адресации.

При выполнении операций с непосредственной адресацией один из операндов imm задается в последних байтах команды (поле IMM на рис. 1. 11). В этом случае ОРС ряда команд содержит бит s, определяющий способ использования непосредственно задаваемых данных. Если операция выполняется над байтами (ОРС команды содержит бит w=0), то в качестве операнда используется один байт непосредственных данных im8, содержащихся в формате команды. Если операция выполняется над 16- или 32-разрядными словами (в ОРС команды бит w=1 или отсутствует), то возможны следующие варианты. При s=0 непосредственные данные содержат два или четыре байта, и в качестве одного из операндов исполь­зуются im16 или im32. При s=1 непосредственные данные содержат один младший байт 16- или 32-разрядного операнда, остальные разряды ко­торого принимают значение старшего (знакового) разряда младшего байта (расширением знаком).

При обращении к памяти байты MODR/M и SIB определяют адрес младшего байта операнда. Если операция выполняется с 16- или 32-разрядными операндами, то старшие байты выбираются из ячеек памяти, значение адреса которых на 1 или 3 больше определяемых байтами адресации.

Процессоры Р6, как и предыдущие модели семейства 80x86, реализуют сегментную организацию памяти, при которой физический адрес ячейки памяти формируется путем сложения базового адреса сег­мента и относительного адреса ячейки внутри сегмента. Базовый адрес определяется содержимым 16-разрядного сегментного регистра и зависит от режима работы микропроцессора. Если микропроцессор работает в режиме обработки 16-разрядных данных (режим реальных адресов или режим виртуального 8086), то 20-разрядный базовый адрес формируется путем сдвига содержимого сегментного регистра на четыре разряда влево. Если микропроцессор работает в режиме обработки 32-разрядных данных (защищенный режим), то 32-разрядный базовый адрес содержится в дес­крипторе, выбор которого из таблицы дескрипторов осуществляется с по­мощью селектора- содержимого соответствующего сегментного регистра.

В зависимости от типа обращения к памяти производится выбор сегментного регистра и способа определения относительного адреса (табл. 1. 8). Для некоторых способов обращения возможны варианты вы­бора сегментных регистров, которые указаны в табл. 1. 8 в скобках. Эти варианты могут быть выбраны с помощью префикса замены сегмента SEG. В качестве относительного адреса используется содержимое регистров EIP (IP), ESP (SP), ESI (SI), EDI (Dl) или эффективный адрес ЕА, который формируется в соответствии с заданным способом адресации.

Таблица 1. 8. Выбор сегментных регистров и относительного адреса

|  |  |  |
| --- | --- | --- |
| Тип обращения к памяти | Сегментный регистр | Относительный адрес\* |
| 1. Выборка команды | CS | EIP (IP) |
| 2. Обращение к стеку | SS | ESP(SPi |
| 3. Адресация операнда | DS (CS, SS, ES, FS, GS) | EA |
| 4. Адресация элемента строки-источника | DS (CS, SS, ES, FS, GS) | ESI (SI) |
| 5. Адресация элемента строки-приемника | ES | EDI (Dl) |
| 6. Адресация операнда с использованием в качест­ве базового регистра ЕВР (ВР)или ESP (SP) | SS (CS, DS, ES, FS, GS) | EA |

\* В скобках указаны регистры, содержащие относительный адрес при обработке 16-разрядных операндов

Эффективный адрес операнда ЕА является 16- или 32-разрядным и формируется в зависимости от значения полей MOD и R/M в байте адресации MODR/M и содержимого байта SIB (для 32-разрядных адресов). В общем случае ЕА образуется путем арифметического сложения трех компонент:

• содержимого базового регистра ЕВР (ВР) или ЕВХ (ВХ);

• содержимого индексного регистра ESI (SI) или EDI (DI);

• 8-, 16- или 32-разрядного смещения d8, d16 или d32, заданного в од­ном, двух или четырех байтах поля DISP команды (рис. 1. 11).

В зависимости от значения полей MOD и R/M для формирования ЕА используются все или часть этих слагаемых в соответствии с таблицами 1. 9, 1. 10. В этих таблицах указаны также сегментные регистры (DS: и SS: ), используемые для определения базовых адресов сегмента.

Если при формировании 32-разрядного адреса значение поля R/M=100, то команда содержит дополнительный байт адресации SIB.

Таблица 1. 9. Формирование 16-разрядного ЕА

|  |  |  |  |
| --- | --- | --- | --- |
| Поле R/M | Поле MOD | | |
| 00 | 01 | 10 |
| 000 | DS: [BX+SI] | DS: [BX+SI+d8] | DS: [BX+SI+d16] |
| 001 | DS: [BX+DI] | DS: [BX+DI+d8] | DS: [BX+DI+d16] |
| 010 | SS: [BP+SI] | SS: [BP+SI+d8] | SS: [BP+SI+d16] |
| 011 | SS: [BP+DI] | SS: [BP+DI+d8] | SS: [BP+DI+d16] |
| 100 | DS: [SI] | DS: [SI+d8] | DS: [SI+d16] |
| 101 | DS: [DI] | DS: [DI+d8] | DS: [DI+d16] |
| 110 | DS: [d16] | SS: [BP+d8] | SS: [BP+d16] |
| 111 | DS: [BX] | DS: [BX+d8] | DS: [BX+d16] |

Таблица 1.10. Формирование 32-разрядного ЕА(байт SIB отсутствует)

|  |  |  |  |
| --- | --- | --- | --- |
| Поле R/M | Поле MOD | | |
| 00 | 01 | 10 |
| 000 | DS: [EAX] | DS: [EAX+d8] | DS: [EAX+d32] |
| 001 | DS: [ECX] | DS: [ECX+d8] | DS: [ECX+d32] |
| 010 | DS: [EDX] | DS: [EDX+d8] | DS: [EDX+d32] |
| 011 | DS: [EBX] | DS: [EBX+d8] | DS: [EBX+d32] |
| 100 | (см. табл. 1. 11) | (см. табл. 1. 11) | (см. табл. 1. 11) |
| 101 | DS: [d32] | SS: [EBP+d8] | SS: [EBP+d32] |
| 110 | DS: [ESI] | DS: [ESI+d8] | DS: [ESI+d32] |
| 111 | DS: [EDI] | DS: [EDI+d8] | DS: [EDI+d32] |

В таком случае правила образования ЕА определяются табл. 1. 11, где (IR\*F) — масштабированный индекс; IR — содержимое индексного регистра, который задается кодом INDEX; F — масштабный коэффициент, значение которого определяется кодом SS (табл. 1. 12). В качестве индекс­ного может использоваться любой из регистров общего назначения, кроме указателя стека ЕSР(табл. 1. 13).

Разрядность обрабатываемых операндов и адресов определяется режимом работы процессора. Если программы выполняются в реальном режиме или режиме виртуального 8086, то по умолчанию используются 16-разрядные относительные адреса и операнды. При работе в защищенном режиме дескриптор сегмента исполняемой команды содержит бит D, ко­торый определяет принимаемую по умолчанию разрядность адресов и операндов: 16 при D=0, 32 при D=1.

Таблица 1. 11. Формирование 32-разрядного ЕА (байт SIB присутствует)

|  |  |  |  |
| --- | --- | --- | --- |
| Поле  BASE | Поле MOD | | |
|  | 00 | 01 | 10 |
| 000 | DS: [EAX+(IR\*F)] | DS: [EAX+(IR\*F)+d8] | DS: [EAX+(IR\*F)+d32] |
| 001 | DS: [ECX+(IR\*F)] | DS: [ECX+(IR\*F)+d8] | DS: [ECX+(IR\*F)+d32] |
| 010 | DS: [EDX+(IR\*F)] | DS: [EDX+(IR\*F)+d8] | DS: [EDX+(IR\*F)+d32] |
| 011 | DS: [EBX+(IR\*F)] | DS: [EBX+(IR\*F)+d8] | DS: [EBX+(IR\*F)+d32] |
| 100 | SS: [ESP+(IR\*F)] | SS: [ESP+(IR\*F)+d8] | SS: [ESP+(IR\*F)+d32] |
| 101 | DS: [d32+(IR\*F)] | SS: [EBP+(IR\*F)+d8] | SS: [EBP+(IR\*F)+d32] |
| 110 | DS: [ESI+(IR\*F)] | DS: [ESI+(IR\*F)+d8] | DS: [ESI+(IR\*F)+d32] |
| 111 | DS: [EDI+(IR\*F)] | DS: [EDI+(IR\*F)+d8] | DS: [EDI+(IR\*F)+d32] |

Таблица 1. 12. Кодировка множителя F

|  |  |
| --- | --- |
| Поле SS | Масштабный множитель F |
| 00 | 1 |
| 01 | 2 |
| 10 | 4 |
| 11 | 8 |

Таблица 1. 13. Кодировка индексных и базовых регистров в байт SIB

|  |  |  |  |
| --- | --- | --- | --- |
| Поле INDEX | Индексный регистр | Поле BASE | Базовый регистр |
| 000 | EAX | 000 | EAX |
| 001 | ECX | 001 | ECX |
| 010 | EDX | 010 | EDX |
| 011 | EBX | 011 | EBX |
| 100 | - | 100 | ESP |
| 101 | EBP | 101 | EBP(d32) |
| 110 | ESI | 110 | ESI |
| 111 | EDI | 111 | EDI |

Если команде предшествуют префиксы OS, AS, то при ее выполне­нии принятая по умолчанию разрядность операнда или адреса изменяет­ся. В результате разрядность операндов и относительных адресов в за­щищенном режиме определяется в соответствии с табл. 1. 14. Для реального режима и режима виртуального 8086 выбор разрядности определяется при значениях D=0.

Процессор реализует ряд способов адресации операнда, набор ко­торых обеспечивает эффективную работу с языками высокого уровня (Си, Фортран и др. ).

**Непосредственная адресация.** В качестве операнда Imm исполь­зуются один, два или четыре последних байта команды. Такой способ адресации реализуется при выполнении ряда команд пересылки (MOV, PUSH), арифметических операциях (ADD, ADC, SUB, SBB, CMP, IMUL), и логических операциях (AND, OR, XOR, TEST). Непосредственная адресация задается определенным значением кода ОРС, содержащегося в первом байте этих команд, или поля REG/OPC байта MODR/M. Разрядность используемых непосредственных данных (8, 16 или 32 разряда) зависит от режима работы процессора и может изменяться соот­ветствующим префиксом.

Таблица 1.14. Разрядность операндов и относительных адресов в защи­щенном режиме ( + наличие префикса, — отсутствие префикса)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Значение бита D | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Префикс OS ( 66H ) | - | - | + | + | - | - | + | + |
| Префикс AS ( 67H ) | - | + | - | + | - | + | - | + |
| Разрядность операнда | 16 | 16 | 32 | 32 | 32 | 32 | 16 | 16 |
| Разрядность относительного адреса | 16 | 32 | 16 | 32 | 32 | 16 | 32 | 16 |

**Регистровая адресация.** Операнд выбирается из регистра, определяемого полем R/M в байте MODR/M. Код, содержащийся в этом поле, задает выбираемый регистр в соответствии с табл. 1. 6. Данный спо­соб реализуется при задании в байте MODR/M значения поля MOD=11.

**Косвенно-регистровая адресация.** Относительный адрес ЕА со­держится в индексном (SI, Dl, ESI, EDI) или базовом (ВХ, ЕВХ) регистрах, или регистрах общего назначения EAX, ECX, EDX. Данный способ реализуется при значении поля MOD=00 (таблицы 1. 9, 1. 10).

**Прямая адресация.** Относительный адрес операнда ЕЛ со­держится в команде в виде смещения d16 или d32. Этот способ адресации реализуется при значениях полей R/M=110 и MOD=00, R/M=110 (табл. 1. 9) или R/M=101 (табл. 1. 10).

**Базовая адресация.** Относительный адрес ЕЛ формируется путем сложения содержимого базового регистра (ВХ, ВР) и смещения d8 или d16. Базовая адресация реализуется при MOD=01 или 10 и значениях R/M=110 или 111 (табл. 1. 9).

**Индексная адресация.** Относительный адрес ЕЛ формируется пу­тем сложения содержимого индексного регистра (SI, DI) и смещения d8 или d16. Для реализации этого способа задаются значения MOD=01 или 10 и R/M=100 или 101 (табл. 1. 8). При формировании 32-разрядных адресов в качестве базового или индексного, может использоваться любой из регистров EAX, ECX, EDX, EBX, EBP, ESI, EDI (табл. 1. 9).

**Базово-индексная адресация**. Относительный адрес ЕА образуется путем сложения содержимого базового (ВХ, ВР) и сегментного регистров (SI, DI). Такой способ адресации операнда реализуется при зна­чениях поля R/M=000, 001, 010, 011, если задано MOD=00 (табл. 1. 10).

**Базово-индексная адресация со смещением**. Это вариант базо­во-индексной адресации, при котором к относительному адресу ЕА допол­нительно прибавляется смещение d8 или d16. Данный способ адресации осуществляется при значениях поля MOD=01 или 10 (табл. 1. 8).

Дополнительные способы адресации реализуются при использова­нии 32-разрядных адресов, если задано значение R/M=100 и команда со­держит байт SIB. При этом поля INDEX и BASE определяют коды регистров, содержимое которых определяет индекс и базу для формирования адреса операнда (табл. 1.13). При коде INDEX=100 значе­ние индекса принимается равным 0. В этом случае при MOD=00 реализуется косвенно-регистровая или прямая (при BASE=101) адресация, а при MOD=01 или 10 — базовая адресация. Регистр ЕВР вы­бирается в качестве базового при значениях MOD=01 или 10; при MOD=00 в качестве базы используется 32-разрядное смещение (d32), задаваемое в четырех следующих байтах команды. В качестве индексного может ис­пользоваться любой из 32-разрядных регистров общего назначения, кроме ESP (табл. 1.6). Содержимое этого регистра умножается на масштабный коэффициент F (табл. 1.12), то есть сдвигается влево на число разрядов (0, 1, 3 или 4), задаваемое содержимым поля SS в байте SIB.

Таким образом, при введении байта SIB реализуются три дополни­тельных варианта индексной и базово-индексной адресации (табл.1.11).

**Индексная адресация с масштабированием.** Относительный адрес образуется сложением масштабированного индекса (содержимого индексного регистра, умноженного на коэффициент F) и смещения d32. Этот способ осуществляется при MOD=00 и значении поля BASE=101.

**Базово-индексная адресация с масштабированием.** Относитель­ный адрес образуется сложением масштабированного индекса и базы, в качестве которой используется содержимое одного из регистров: ЕАХ, ЕВХ, ЕСХ, EDX, ESP. ESI или EDI. Выбор базового регистра задается зна­чением поля BASE при MOD=00 (табл. 1.13).

**Базово-индексная адресация со смещением и масшта­бированием**. Относительный адрес формируется сложением масшта­бированного индекса, базы и смещения d8 или d32. Поле BASE задает выбор базового регистра (табл.1.13), а поле MOD=01 или 10 определяет разрядность смещения (табл.1.11). При описанных способах задается от­носительный адрес операнда в сегменте данных DS или стека SS (табл.1.11).Для адресации операндов в других сегментах перед командой необходимо ввести префиксный байт SEG.

**Относительная адресация** используется при выполнении ряда ко­манд управления (условные и безусловные переходы, вызовы под­программ, управление циклами), чтобы адресовать ячейку памяти, со­держащую следующую команду. При этом способе адрес формируется как сумма содержимого регистра EIP, соответствующего текущей команде, и смещения d8, d16 или d32, определяющего положение следующей коман­ды относительно текущей.

**2. Система команд: операции над целыми числами**

К этому классу относятся команды выполнения пересылок, арифметических и логических операций, сдвигов и других преобразований данных, в том числе операции над битами и строками символов. Ниже да­ется перечень команд этого класса, разбитых на несколько функциональ­ных групп.

Пересылка данных и адресов

К этому классу относятся команды выполнения пересылок, арифметических и логических операций, сдвигов и других преобразований данных, в том числе операции над битами и строками символов. Ниже да­ется перечень команд этого класса, разбитых на несколько функциональ­ных групп.

Пересылка данных и адресов

Пересылка данных без преобразования: MOV

Пересылка операнда

CMOVcc Условная пересылка операнда SETcc

Условная установка байта

PUSH Запись операнда в стек

PUSHA(D) Запись в стек содержимого всех регистров POP

Чтение операнда из стека

POPA(D) Чтение из стека содержимого всех регистров XCHG

Обмен между регистрами или памятью и регистром

XLAT Преобразование кодов

BSWAP Перестановка байтов

Пересылка данных с преобразованием:

MOVSX Пересылка байта или слова с расширением знака MOVZX

Пересылка байта или слова с расширением нулями

Ввод-вывод данных:

IN Ввод операнда из порта в аккумулятор

OUT Вывод операнда из аккумулятора в порт

Загрузка эффективного адреса и селекторов: LEA

Загрузка эффективного адреса ЕА в регистр

LDS Загрузка указателя адреса для сегмента DS

LES Загрузка указателя адреса для сегмента ES

LFS Загрузка указателя адреса для сегмента FS

LGS Загрузка указателя адреса для сегмента GS

LSS Загрузка указателя адреса для сегмента SS

Арифметические операции

Сложение:

ADD Сложение операндов

ADC Сложение операндов с признаком CF (перенос)

XADD Обмен операндами и сложение

INC Инкремент операнда

ААА ASCII-Коррекция результата сложения

DAA Десятичная коррекция результата сложения

Вычитание: SUB Вычитание операндов

SBB Вычитание операндов и признака CF (заем)

DEC Декремент операнда

NEG Изменение знака операнда (с переводом в дополнительный

код)

AAS ASCII-коррекция результата вычитания

DAS Десятичная коррекция результата вычитания

Сравнение:

СМР Сравнение операндов

CMPXCHG Сравнение и обмен операндами CMPXCHG8B Сравнение и обмен счетверенными словами (8 байтов)

Умножение:

MUL Беззнаковое умножение

IMUL Знаковое (целочисленное) умножение

ААМ ASCII-коррекция результата умножения

Деление:

DIV Беззнаковое деление

IDIV Знаковое (целочисленное) деление

AAD ASCII-коррекция результата деления

Изменение разрядности путем расширения знака: CBW

Преобразование байта (AL) в слово (АХ)

CWDE Преобразование слова (АХ) в двойное слово (ЕАХ)

CWD Преобразование слова (АХ) в двойное слово (DX, АХ)

CDQ Преобразование двойного слова (ЕАХ) в учетверенное слово

(EDX, ЕАХ)

Логические операции

NOT Инверсия операнда (логическое НЕ)

AND Конъюнкция операндов (логическое И)

OR Дизъюнкция операндов (логическое ИЛИ)

XOR Неравнозначность операндов (исключающее ИЛИ)

TEST Логическое сравнение операндов (установка признаков ZF,

SF, PF)

Сдвиги

SHL/SAL Сдвиг влево

SHR Логический сдвиг вправо

SAR Арифметический сдвиг вправо

SHLD Двухоперандный сдвиг влево

SHRD Двухоперандный сдвиг вправо

ROL Циклический сдвиг влево

ROR Циклический сдвиг вправо

RCL Циклический сдвиг влево через перенос (CF)

RCR Циклический сдвиг вправо через перенос (CF)

Операции с битами и байтами

ВТ Проверка бита

BTS Проверка и установка бита

BTR Проверка и сброс бита

ВТС Проверка и инверсия бита

BSF Прямое сканирование битов

BSR Обратное сканирование битов

Scc Условная установка байта

Операции со строками символов

LODS Загрузка символа в аккумулятор

STOS Запись символа из аккумулятора

INS Ввод символа

OUTS Вывод символа

MOVS Пересылка символа

CMPS Сравнение символов

SCAS Сканирование строки символов

В последующих разделах дается описание этих команд. При этом используется синтаксис команд, принятый для языка Ассемблер:

ОРС <dst>, <src>

где в качестве ОРС указывается мнемокод команды, dst (destination — приемник) — адрес первого операнда, на место которого помещается результат операции, srс (source — источник) — адрес второго операнда. Таким образом, порядок следования адресов в поле операндов команды определяет размещение результата. При описании команд приняты сле­дующие обозначения:

г(8,16,32) — 8, 16 или 32-разрядный регистр общего назначения;

m(8,16,32) — 8, 16 или 32-разрядная ячейка памяти (адресуется байтами MODR/M, SIB);

sreg — сегментный регистр CS, SS, DS, ES, FS или GS;

А — регистр-аккумулятор ЕАХ, АХ или AL (в зависимости от раз­рядности операндов).

Большинство команд данного класса выполняет операции над 8-, 16-или 32-разрядными операндами. Заданная разрядность определяется, как указано в разделе 1.4, режимом работы процессора (16 для реального режима и режима виртуального 8086; 16 или 32 для защищенного режима в зависимости от значения бита D в дескрипторе) и может меняться с по­мощью префиксных байтов.

***2.1. Команды пересылки***

*Команды этой группы приведены в табл. 2.1. Значения признаков в регистре EFLAGS при выполнении команд пересылок не изменяются.*

*Команда пересылки MOV имеет ряд модификаций, основная из ко­торых MOV r/m,m, MOV m.r/m выполняет передачу данных (регистр-регистр) или (регистр-память) между регистрами общего назначения и ячейками памяти.*

*Таблица 2.1. Команды пересылок*

|  |  |  |
| --- | --- | --- |
| *Синтаксис команды* | | *Операция* |
| *MOV* | *r/m(8, 16, 32), r(8, 16, 32) r(8, 16, 32), r/m(8, 16, 32)* | *r(8, 16, 32) 🡪 r/m(8, 16, 32) r/m(8, 16, 32) 🡪 r(8, 16, 32)* |
| *CMOVcc* | *r/m(8, 16, 32), r(8, 16, 32) r(8, 16, 32), r/m(8, 16, 32)* | *r(8, 16, 32) 🡪 г/m(8, 16, 32), если (сс)=1 r/m(8, 16, 32) 🡪 r(8, 16, 32), если (сс)=1* |
| *MOV* | *EAX(AX, AL), d(16, 32) d(16, 32), EAX(AX, AL)* | *(d16 или d32) 🡪 EAX(AX, AL) EAX(AX, AL) 🡪 (d16 или d32)* |
| *MOV* | *r/m(8, 16, 32), im(8, 16, 32)* | *lm(8, 16, 32) 🡪 r/m(8, 16, 32)* |
| *MOV* | *sreg, r/m16 r/m16, sreg* | *r/m16 🡪 sreg sreg 🡪 r/m16* |
| *MOV* | *CRi, r32 r32, CRi* | *r32 🡪 CRi CRi 🡪 r32* |
| *MOV* | *DRi, r32 r32, DRi* | *r32 🡪 DRi DRi 🡪 r32* |
| *XCHG* | *r(8, 16, 32), r/m(8, 16, 32) EAX(AX), r/m(16, 32)* | *r(8, 16, 32) ↔ r/m(8, 16, 32) EAX(AX) ↔ r/m(16, 32)* |
| *PUSH* | *r/m(16, 32) im(16, 32)* | *SP - (2или 4) 🡪 SP, r/m(16, 32) 🡪 (SP) SP - (2или 4) 🡪 SP, im(16, 32) 🡪 (SP)* |
| *POP* | *r/m(16, 32) sreg* | *(SP) 🡪 r/m(16, 32), SP+(2или 4) 🡪 SP (SP) 🡪 sreg, SP+ (2или 4) 🡪 SP* |
| *PUSHA* |  | *AX, CX, DX, BX, SP, BP, SI, Dl 🡪 стек* |
| *POPA* |  | *стек 🡪 DI, SI, BP, SP, BX, DX, CX, AX* |
| *PUSHAD* |  | *EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI 🡪 стек* |
| *POPAD* |  | *стек 🡪 EDI, ESI, EBP, ESP, EBX, EDX, ECX, EAX* |
| *IN* | *EAX(AX, AL), im8 EAX(AX, AL), DX* | *port(im8) 🡪 EAX(AX, AL) port(DX) 🡪 EAX(AX, AL)* |
| *Синтаксис команды* | | *Операция* |
| *OUT* | *im8, EAX(AX, AL) DX, EAX(AX, AL)* | *EAX(AX, AL) 🡪 port(im8) EAX(AX, AL) 🡪 port(DX)* |
| *MOVSX* | *r(16, 32), r/m8 r32, r/m16* | *r/m8 🡪 r(16, 32), расширение знаком r/m16 🡪 r32, расширение знаком* |
| *MOVZX* | *r(16, 32), r/m8 r32, r/m16* | *r/m8 🡪 r(16, 32), расширение нулями r/m16 🡪 r32, расширение нулями* |
| *BSWAP* | *r32* | *r(7-0) 🡪 r(31-24), r(15-8) 🡪 r(23-16), r(23-16) 🡪 r(15-8), r(31-24) 🡪 r(7-0)* |
| *XLATB*  *xlat* | *m8* | *(ЕВХ + AL) 🡪 AL* |
| *LEA* | *r(16, 32), m(16, 32)* | *ЕА 🡪 r(16, 32)* |
| *LDS* | *r16, m16: 16 r32, m16: 32* | *m16 🡪 DS, m16 🡪 r16 m16 🡪 DS, m32 🡪 r32* |
| *LES* | *r16, m16: 16 r32, m16: 32* | *m16 🡪 ES, m16 🡪 r16 m16 🡪 ES, m32 🡪 r32* |
| *LFS* | *r16, m16: 16 r32, m16: 32* | *m16 🡪 FS, m16 🡪 r16 m16 🡪 FS, m32 🡪 r32* |
| *LGS* | *r16, m16: 16 r32, m16: 32* | *m16 🡪 GS, m16 🡪 r16 m16 🡪 GS, m32 🡪 r32* |
| *LSS* | *r16, m16: 16 r32, m16: 32* | *m16 🡪 SS, m16 🡪 r16 m16 🡪 SS, m32 🡪 r32* |

*Модификация MOV A, m, MOV m, A, которая реализует загрузку или сохранение в памяти содержимого аккумулятора, имеет сокращенный формат (отсутствуют байты адресации MODR/M, SIB). При этом в команде задается относительный 16- или 32-разрядный адрес операнда d16 или d32 в адресуемом сегменте данных, то есть используется только прямая адресация памяти. Модификация MOV r/m, im производит запись в регистр или ячейку памяти непосредственно заданного операнда. Пересылки с участием управляющих регистров CR4-0 или регистров отладки DR7-0 (ко­манды MOV CRi, r, MOV r, CRi, MOV DRi, r, MOV r, DRi) выполняются только в защищенном режиме программами, имеющими высший уровень приви­легии 0. Они не реализуются в режиме виртуального 8086. В качестве ре­гистра r должен использоваться один из 32-разрядных регистров общего назначения.*

*В процессорах семейства Р6 введены команды условной пересылки CMOVcc. Эти команды производят передачу данных только в случае вы­полнения определенного условия (cc), которое задается различными зна­чениями признаков, установленных в регистре EFLAGS. Вид проверяемого условия указывается с помощью суффикса cc, вводимого в мнемокод ко­манды: например, CMOVZ — пересылка, если значение признака ZF=1. Виды условий и соответствующие суффиксы cc в командах CMOVcc ана­логичны используемым в командах условных переходов Jcc, которые рас­смотрены в разделе 3. 1 (см. табл. 3. 2).*

*Две модификации команды XCHG обеспечивают обмен со­держимым (регистр-регистр) или (регистр-память). В обмене могут участ­вовать только регистры общего назначения. Модификация XCHG A, r, реа­лизующая обмен (аккумулятор-регистр), имеет сокращенный формат ко­манды (без байтов адресации MODR/M, SIB).*

*Команды PUSH, POP производят запись в стек или извлечение из стека 16- или 32-разрядных операндов. В стек может быть загружено со­держимое регистра, ячейки памяти или непосредственный операнд im. При извлечении из стека данные могут поступать в регистр общего назначения, ячейку памяти или указанный в команде сегментный регистр sreg. Адресация стека производится с помощью регистра ESP (или SP), который содержит относительный адрес последней заполненной ячейки памяти в сегменте стека SS. При выполнении команды PUSH содержимое ESP (или SP) уменьшается на 4 (или на 2), адресуя выше расположенную (незапол­ненную) ячейку стека, которая становится его новой вершиной, Эта ячейка затем заполняется поступающими данными. При выполнении команды POP производится выборка данных из ячейки стека, адресуемой содержи­мым регистра ESP (или SP). После выборки содержимое этого регистра увеличивается на 4 (или 2), адресуя ниже расположенную заполненную ячейку стека, которая становится его новой вершиной.*

*Команды PUSHA, PUSHAD и POPA, POPAD выполняют запись в стек и извлечение из него всех восьми регистров общего назначения. При этом последовательно заполняются или освобождаются восемь 16- или 32-разрядных ячеек стека. Соответственно, содержимое регистра ESP (или SP) уменьшается или увеличивается на 32 или 16). Команды PUSHA, РОРА оперируют с 16-разрядными регистрами АХ, СХ, DX, BX, SP, BP, SI, DI, команды PUSHAD, POPAD — с 32-разрядными регистрами ЕАХ, ЕСХ, EDX, EBX, ESP, EBP, ESI, EDI.*

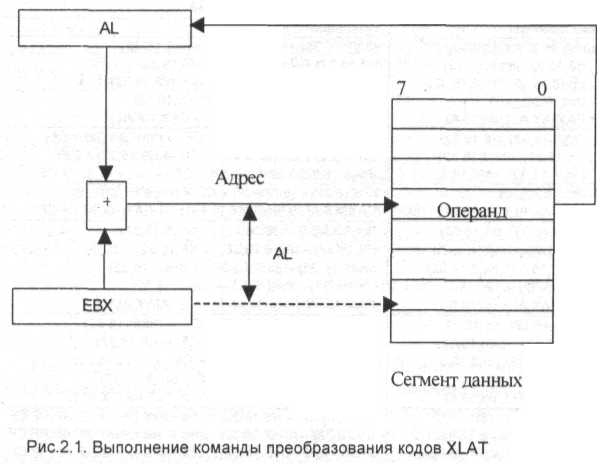
*При выполнении команд ввода-вывода IN, OUT производится пересылка данных между аккумулятором ЕАХ (АХ или AL) и адресуемым портом. Адрес порта может задаваться непосредственно операндом im8 (фиксированный порт) или содержимым регистра DX (изменяемый порт). При работе процессора с 8-разрядными портами возможно обращение к 256 портам, адреса которых фиксированы в программе, или к 65536 портам, адреса которых могут изменяться путем перезагрузки со­держимого регистра DX. При работе с 16-разрядными портами число адресуемых портов уменьшается вдвое, а их нумерация (адресация) должна производиться четными числами: 0, 2, 4...254 для фиксированных портов; 0, 2, 4...65532, 65534 для изменяемых портов. Соответственно, для 32-разрядныхпортов их число сокращается еще вдвое, а нумерация производится числами, кратными четырем: 0, 4, 8,...248, 252,...65528, 655332. Таким образом номера портов должны быть выровнены по границам передаваемых данных согласно их разрядности: 1, 2 или 4 бай­та. В этом случае пересылка осуществляется за один цикл. Если заданный в команде номер 16- или 32-разрядного порта имеет не выровненное зна­чение, например 03h, то для передачи данных требуется дополнительный цикл. Отметим, что номера портов от 00F8h до OOFFh зарезервированы фирмой Intel для внутреннего использования. Во избежание возможных слоев в работе системы применение этих номеров программистами долж­но быть исключено.*

*Выполнение команд IN, OUT в защищенном режиме зависит от зна­чений уровня привилегии текущей программы CPL и уровня привилегии ввода-вывода IOPL, задаваемого соответствующим полем регистра EFLAGS. При CPL < IOPL команда выполняется. При CPL > IOPL эта возможность обращения к адресованному порту проверяется по битовой карте ввода-вывода (БКВВ), содержащейся в сегменте состояния выполняемой задачи TSS. Если БКВВ разрешает обращение к данному порту, то команда IN, OUT выполняется, если не разрешает, то реализуется прерывание типа 13 "нарушение защиты". При работе процессора в режиме виртуального 8086 проверка допустимости обращения к заданному порту производится с помощью БКВВ при любых значениях CPL, IOPL.*

*Команды MOVSX, MOVZX осуществляют пересылку в регистр обще­го назначения содержимого другого регистра или ячейки памяти с одно­временным расширением разрядности операнда. При этом производится либо преобразование байта в слово (реальный режим и режим виртуаль­ного 8086) или двойное слово (защищенный режим, либо слово преобразуется в двойное слово (защищенный режим). При выполнении команды MOVSX расширение производится путем заполнения старшего байта или слова значением знака (старшего разряда), при выполнении команды MOVZX — заполнением нулями.*

*Команда BSWAP осуществляет перестановку байтов в заданном 32-разрядном регистре. При этом попарно меняются местами старший и младший байты (байт 3 и байт 0) и промежуточные байты (байт 1 и байт 2), т. е. порядок следования байтов меняется справа налево. Численные зна­чения байтов сохраняются.*

*Команды XLAT, XLATB заменяют содержимое младшего байта ак­кумулятора AL на байт, хранящийся в 256-байтовой таблице, начальный (базовый) адрес которой содержится в регистре ЕВХ (или ВХ в реальном режиме или режиме виртуального 8086). При этом содержимое AL (без зна­ковое число) используется как относительный адрес (индекс) выбираемого в таблице байта. Выполнение команд XLAT, XLATB иллюстрируется на рис. 2.1. Эти команды служат для табличного преобразования одного кода в другой. Отметим, что команда XLATB всегда реализует обращение к таблице, размещенной в сегменте данных DS. Команда XLAT позволяет обращаться к таблицам в различных сегментах данных путем задания со­ответствующего адреса операнда. Например, обращение к таблице в сег­менте ES реализуется с помощью команды XLAT ES:[EBX].*

**

*При выполнении команд LEA, LDS, LES, LFS, LGS, LSS операнд-источник <src> всегда размещается в памяти, операнд-приемник <dst> — в регистре общего назначения.*

*Команда LEA производит вычисление эффективного адреса ЕА операнда-источника и загружает его в указанный регистр.*

*Команды LDS, LES, LFS, LGS, LSS производят извлечение из памя­ти и запись в регистры процессора указателя адреса, который состоит из 16-разрядного селектора, поступающего в соответствующий сегментный регистр DS, ES, FS, GS, SS, и 16- или 32-разрядного относительного адре­са операнда, поступающего в заданный регистр общего назначения. Раз­рядность адреса определяется режимом работы процессора: 16 — в ре­альном режиме и режиме виртуального 8086, 32 — в защищенном режи­ме. Соответственно, из памяти выбираются 4 или 6 байт, начиная с адре­са, указанного в поле операндов команды. С помощью этих команд реали­зуется переключение программы с одного сегмента данных на другой. На­пример, команда LSS при заданном регистре-приемнике ESP по­зволяет перейти к новому сегменту стека.*