Лабораторная работа 5. НЕПРИВИЛЕГИРОВАННЫЕ КОМАНДЫ ПРОЦЕССОРОВ x86

Семейство микропроцессоров 80x86 американской фирмы Intel включает в себя ряд программно совместимых процессоров.

В 1979 г. появился первый 16-битный микропроцессор семейства - 8086. Его особенность по сравнению с другими процессорами того времени - сегментная организация памяти, что позволило при 16-битных регистрах расширить адресное пространство памяти до 1 Мбайт.

Появившийся позже микропроцессор 8088, отличающийся от микропроцессора 8086 лишь 8-битной шиной данных, был применен фирмой IBM в персональных компьютерах PC и PC/XT.

Следующая модель этого семейства, микропроцессор 80286, принципиально отличается от 8086. Он может работать в двух режимах - реальном и защищенном. В реальном режиме процессор 80286 отличается от 8086 только скоростью работы, а в защищенном режиме работает совершенно по-другому: используется иной метод формирования адресов, введены средства поддержки мультизадачности, средства защиты, адресное пространство расширено до 16 Мбайт.

Дальнейшее развитие семейства – появление 32-разрядного процессора 80386, работающего с 4 Гбайт физической памяти и с 64 Тбайт виртуальной памяти. Введен новый режим – виртуального процессора 8086, улучшена производительность, расширена система команд и набор регистров.

Главной особенностью процессоров 80486 и Pentium является объединение на одном кристалле процессора, сопроцессора и кэш-памяти первого уровня. В этих процессорах сильно изменена архитектура (особенно в процессоре Pentium) с целью повышения производительности.

При подаче сигнала сброса или при включении питания устанавливается реальный режим работы. В этом режиме процессоры имеют практически такую же базовую архитектуру, что и процессор 8086, но работают значительно быстрее, и, кроме того, процессор обеспечивается доступ к 32-разрядным

регистрам. Механизм адресации, размеры памяти и обработка прерываний в этом режиме полностью совпадают с процессором 8086.

ОРГАНИЗАЦИЯ ПАМЯТИ

Физическая память в микропроцессорной системе организована как линейная последовательность 8-битных байтов. Каждому байту соответствует уникальный адрес, который может находиться в интервале от 0 до (2^32)-1 (4 Гбайт) для процессора 80386 и от 0 до (2^24)-1 (16 Мбайт) для процессора 80286.

Организация памяти в реальном режиме наиболее проста и ничем не отличается от процессора 8086. Память организована в виде сегментов размером 64 Кбайт и базовым адресом, находящимся на 16-байтной границе. Для обращения к памяти нужно определить базу сегмента и смещение в сегменте. Каждому логическому адресу соответствует уникальный физический адрес.

Недостатки организации памяти в реальном режиме:

- ограничение размера адресного пространства (1 Мбайт);
- произвольное размещение сегментов в памяти;
- сегменты памяти не имеют никаких атрибутов защиты или ограничения доступа, и программа может обратиться по любому физическому адресу.

В защищенном режиме процессоров 80x86 организация памяти сильно изменена. В нем, как и в реальном режиме, существуют понятия логического и физического адреса. Логический адрес состоит из двух компонент - селектора сегмента и смещения. Селектор заносится в сегментный регистр. Преобразование логического адреса в физический выполняется не просто сложением со сдвигом, а при помощи специальных таблиц преобразования адресов.

Таблица преобразования адресов называется ДЕСКРИПТОРНОЙ таблицей. Она состоит из последовательности 8-байтных структур данных - ДЕСКРИПТОРОВ. Дескрипторные таблицы располагаются в оперативной памяти и формируются программно (как правило - операционной системой).

Каждый сегмент в работающей системе описывается своим дескриптором. Дескриптор сегмента содержит информацию, необходимую для правильной работы в защищенном режиме:

- базовый адрес сегмента в памяти;
- предел (размер) сегмента;
- различные атрибуты, такие как права доступа, размер по умолчанию, гранулярность и др.

Селектор является индексом в дескрипторной таблице, и адресует дескриптор соответствующего сегмента. Конкретный адрес нужного сегмента выбирается из дескриптора, складывается со смещением, и если не нарушены условия защиты, то полученная сумма является искомым линейным адресом.

Если в процессоре отключен механизм страничной адресации, то полученный линейный адрес выставляется на шину адреса, т.е. является физическим.

Страничный механизм обеспечивает другой уровень организации памяти. Он разбивает линейное адресное пространство на блоки фиксированной длины (4 Кбайт), называемые страницами. Логическое адресное пространство отображается в линейное адресное пространство, которое в свою очередь отображается на несколько страниц. Старшие 20 бит линейного адреса используются для выбора страницы из таблицы страниц, а младшие 12 бит - это смещение внутри страницы. Физический адрес получается после еще одной ступени преобразования.

Механизм преобразования адреса в защищенном режиме, дескрипторы, дескрипторные таблицы и связанные с ними регистры будут рассмотрены при описании защищенного режима.

РЕГИСТРЫ

1. Регистры общего назначения

Микропроцессор x86 содержит восемь 32-разрядных регистров общего назначения: EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI,

которые, по сути, являются расширением 16-разрядных регистров микропроцессоров 8086 и 80286 до 32 бит (рис. 1.1).

Данные регистры используются для хранения операндов логи ческих и арифметических команд. Также они могут использоваться для хранения операндов при вычислении адресов (кроме регистра ESP, который не может быть использован как индексный операнд).

Рис. 1.1. Регистры общего назначения

Однако некоторые команды используют фиксированные регистры для хранения операндов: команды умножения и деления, ввода/вывода, обработки строк, перекодирования, цикла, сдвига и циклического сдвига, операции со стеком.

2. Сегментные регистры

Имеется шесть 16-разрядных сегментных регистров (рис. 1.2):

CS - адресует кодовый сегмент;

DS - сегмент данных;

SS - сегмент стека;

ES,FS,GS - дополнительные сегменты данных (FS и GS - ачиная с процессора 80386).

Эти регистры содержат 16-разрядные селекторы, по которым вычисляется базовый адрес соответствующего сегмента. Способ вычисления базового адреса в реальном и защищенном режимах различен.

В реальном режиме физический адрес базы конкретного сегмента вычисляется аналогично процессору 8086, т.е. получается путем расширения значения селектора до 20 бит (приписывая четыре нулевых бита слева).

В защищенном режиме селектор является индексом, по которому из таблицы дескрипторов выбирается базовый адрес сегмента, его граница (предел) и права доступа.

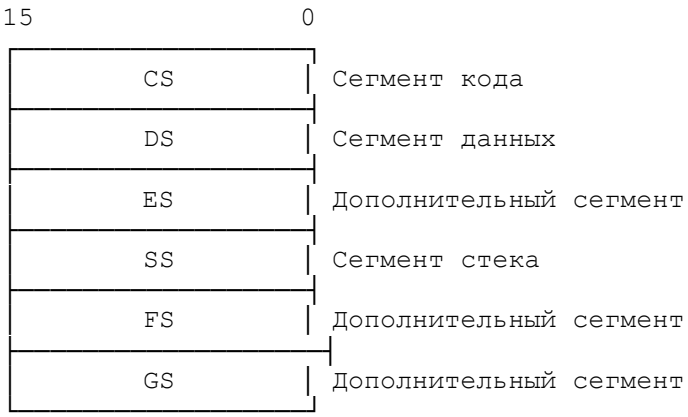

Рис. 1.2. Сегментные регистры

3. Регистр флагов

Микропроцессор 80386 содержит 32-разрядный регистр EFLAGS с 13 полями флагов (рис. 1.3).

Флаги можно разбить на три группы:

1. Флаги состояния:
   - флаг переноса CF (бит 0);
   - флаг четности PF (бит 2);
   - вспомогательный флаг переноса AF (бит 4);
   - флаг нулевого результата ZF (бит 6);

- флаг знака (бит 7);
- флаг переполнения OF (бит 11).

Эта группа ничем не отличается от 8086.

2. Флаги управления:

- флаг трассировки TF (бит 8);
- флаг разрешения прерываний IF (бит 9);
- флаг направления DF (бит 10);
- флаг виртуального режима VM (бит 17).

Когда этот флаг установлен, процессор переходит из защищенного режима в виртуальный режим процессора 8086. Он может быть изменен только двумя способами: командой IRET в защищенном режиме при нулевом текущем уровне привилегий или переключением задачи.

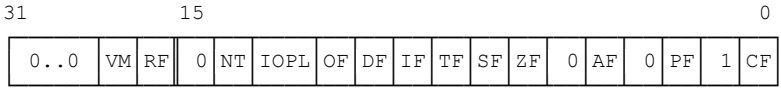

Рис. 1.3. Регистр флагов

3. Специальные флаги. Используются только в защищенном режиме:

- флаг уровня привилегии ввода/вывода IOPL (2-разрядный, биты 12-13). Показывает наивысший уровень привилегий, позволяющий выполнять команды ввода-вывода, не создавая исключительной ситуации. Может быть изменен командами POPF и IRET н нулевом уровне привилегий или при переключении задач;

- флаг вложенной задачи NT (Nested Task) - бит 14.

Если NT=1, то значит текущая задача связана с предыдущей;

- флаг рестарта RF (бит 16). Используется после ошибки при отладке, приведшей к прекращению работы программы.

4. Регистр EIP (указатель команды)

Этот регистр содержит 32-разрядный относительный адрес следующей команды, подлежащей выполнению. Физический адрес образуется путем сложения содержимого регистра EIP с базовым адресом сегмента текущей программы.

Младшие 16 бит называются IP и могут быть использованы

независимо при исполнении программ процессоров 8086 и 80286, которые имеют только регистр IP.

5. Регистры состояния и управления

В эту группу входят четыре 32-разрядных регистра CR0-CR3 (рис. 1.4).

Младшие 16 бит регистра CR0 называются словом состояния машины (MSW). Оно впервые появилось в микропроцессоре 80286. В более поздних процессорах MSW сохранено для совместимости и являются частью регистра CR0.

Назначение разрядов CR0:

- PE (Protection Enable, бит 0) - разрешение защиты. Используется для активизации защищенного режима. При сброшенном PE процессор работает в режиме реальной адресации. Установить этот бит можно загрузкой MSW или CR0, а сбросить - только загрузкой CR0;
- MP (Math Present, бит 1) - сопроцессор присутствует. Используется совместно с битом TS для определения, будет ли команда WAIT генерировать особую ситуацию 7 ("сопроцессор отсутствует"). Данная особая ситуация генерируется при установленных битах MP и TS;
- EM (Emulation, бит 2) - эмуляция сопроцессора. Показывает необходимость эмуляции команд сопроцессора. Установка этого бита приводит к генерации особой ситуации 7 ("сопроцессор отсутствует") при попытке выполнить любую команду сопроцессора. Значение EM не влияет на выполнение команды WAIT;
- TS (Task Switched, бит 3) - задача переключена. Устанавливается автоматически при каждом переключении задач и анализируется при поступлении команды сопроцессора. Позволяет откладывать сохранение/восстановление числовых данных до их фактического использования. Этот бит можно сбросить командой CLTS;
- ET (Extension Type, бит 4) - тип расширения (только в процессоре 80386). Показывает тип сопроцессора (80287 или 80387). Если ET=1, то используется 80387, если ET=0, то 80287. При необходимости этот бит может быть сброшен или установлен загрузкой CR0, но не изменяется командой LMSW;

- PG (PaGing, бит 31) - включение механизма разбиения на страницы.

Регистр CR1 зарезервирован фирмой Intel для возможного использования в будущих моделях процессоров.

Регистр CR2 используется только при установленном бите PG в регистре CR0, т.е. при включенном механизме страничной адресации и содержит полный 32-разрядный линейный адрес, поступление которого в блок страничной адресации вызвало особую ситуацию 14 ("страничная ошибка"). Обработчик этой ситуации использует содержимое этого регистра для выяснения причины возникновения ошибки.

Регистр CR3, как и CR2, используется при страничной адресации и содержит 20 старших бит физического адреса каталога страниц. Этот регистр также называется базовым регистром каталога страниц (PDBR). Поскольку каталог страниц всегда выровнен на размер страницы (4 Кбайт), младшие 12 бит этого регистра не используются, и при записи в регистр CR3 32-разрядного адреса изменяются только старшие 20 бит.

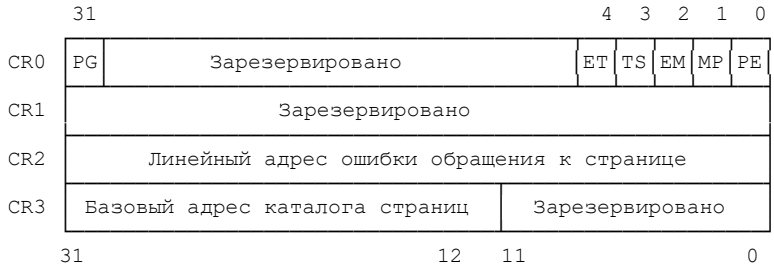

Рис. 1.4. Регистры состояния и управления

Регистры управления доступны системному программисту только через специальные варианты команды MOV, например:

MOV EAX, CR0

MOV CR3, EBX

Младшие 16 бит регистра CR0 (MSW) могут быть считаны или записаны командами SMSW и LMSW .

6. Регистры отладки

Для управления процессором при отладке используются 32-разрядные регистры DR0-DR7. Впервые эти регистры появились в процессоре 80386.

7. Тестовые регистры

Тестовые регистры TR6 и TR7 процессора 80386 используются для проверки работы буфера ассоциативной трансляции (TLB). TLB-это кэш-память, используемая для трансляции линейных адресов в физические.

Здесь не были рассмотрены четыре системных регистра: GDTR,IDTR, LDTR и TR. Все они используются в защищенном режиме и позже будут описаны подробно.

ПРЕРЫВАНИЯ И ОСОБЫЕ СИТУАЦИИ

В процессорах 80x86 предусмотрены два механизма прерывания нормального выполнения программы:

1. Прерывание - это асинхронное событие, активизируемое сигналом от внешнего по отношению к процессору устройства. Прерывания делятся на:

- маскируемые, которые активизируются сигналом на входе INTR процессора. Если флаг IF установлен, то маскируемые прерывания не проходят;
- немаскируемые, активизируются сигналом на входе NMI и не могут быть запрещены процессором.

2. Особая ситуация - это синхронное событие, которое активизируется самим процессором в ходе выполнения последовательности инструкций (команд). Особая ситуация также называется исключением (exception).

Все особые ситуации делятся на 2 группы:

1. Программные прерывания. Они могут быть вызваны командами INTO, INT3, INT n, BOUND.
2. Определяемые процессором особые ситуации. В свою очередь, они делятся на ошибки, ловушки и сбои.

Ошибка (fault) - это особая ситуация, обнаруживаемая процессором или до выполнения инструкции, или во время ее выполнения. Адрес возврата для обработчика ошибки указывает на команду, сгенерировавшую данную ошибку, а не на команду,

следующую за ней (т.е. после обработки ошибки происходит повторное выполнение этой же команды). Например, прикладная программа обратилась к данным в сегменте, не присутствующем в памяти. В этом случае обработчик особой ситуации должен загрузить отсутствующий нужный сегмент (возможно, с жесткого диска) и возобновить выполнение программы, начиная с команды, вызвавшей ошибку.

Ловушка (trap) - особая ситуация, обнаруживаемая после окончания выполнения ошибочной команды. После обработки ловушки управление передается на следующую команду.

Сбой (abort) - это особая ситуация, вызванная тяжелой ошибкой (например, аппаратной ошибкой, противоречивыми или недопустимыми значениями в системных таблицах). В этом случае теряется контекст прерванной программы, и продолжить ее невозможно.

Прерывания и исключения сходны по действию: они заставляют процессор временно приостановить выполнение программы, и передать управление обработчику соответствующего прерывания или особой ситуации.

Процессор ассоциирует с каждым отдельным типом прерывания или особой ситуации идентифицирующий его номер (вектор). Немаскируемым прерываниям и особым ситуациям присвоены векторы в диапазоне от 0 до 31. Не все из этих векторов используются процессором в настоящее время; не назначенные векторы из этого диапазона резервируются для возможного использования в будущем, поэтому использовать их не следует.

Векторы маскируемых прерываний определяются аппаратно. Контроллеры внешних прерываний передают вектор на шину процессора во время цикла подтверждения прерывания. Использоваться могут любые векторы в диапазоне от 32 до 255.

Вектор прерывания - это индекс, по которому в специальной таблице, называемой таблицей дескрипторов прерываний, выбирается специальная структура данных, называемая дескриптором прерывания. Этот дескриптор описывает процедуру или задачу, обслуживающую данное прерывание или особую ситуацию. Дескриптор прерывания содержит селектор

сегмента, смещение и некоторые атрибуты обслуживающей программы.

При инициализации процессора (в реальном режиме) таблица прерываний располагается по адресу 0000:0000. Элемент таблицы прерываний в реальном режиме занимает 4 байта - 16-разрядный сегментный адрес и 16-разрядное смещение.

При возникновении некоторых особых ситуаций в защищенном режиме процессором помещается на вершину стека (после вызова процедуры прерывания) код ошибки (рис. 1.5).

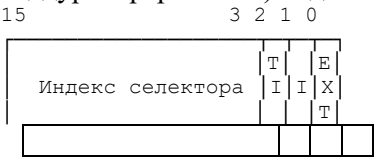

Рис. 1.5. Формат кода ошибки

Процессор устанавливает бит EXT, если особая ситуация вызвана событием, внешним по отношению к программе.

Процессор устанавливает I-бит (IDT-bit), если индексная часть кода ошибки ссылается к дескриптору шлюза в IDT.

Если бит IDT не установлен, то бит TI указывает на то, ссылается ли код ошибки к GDT (бит TI очищен), или же к LDT (бит TI установлен). Остальные 14 битов - это старшие биты селектора сегмента. В некоторых случаях код ошибки является пустым (т.е. все биты его младшего слова очищены).

В 32-битном процессоре x86 код ошибки помещается в стек в виде двойного слова, старшая половина которого резервируется.

СИСТЕМА КОМАНД

В микропроцессоре 80286 способы адресации и формат команд полностью соответствуют процессору 8086. В реальном режиме появилось несколько новых команд.

Разработчики процессора 80386 при проектировании системы команд решали следующие цели:

- обеспечение полной совместимости с предшествующими моделями процессоров с введением в команды 32-разрядных операндов и адресов;

- расширение диапазона применения регистров общего назначения в адресных вычислениях;
- введение новых команд, упрощающих решение часто встречающихся программных задач.

1. Способы адресации

Режимы адресации процессора 80386 мало чем отличаются от процессора 8086. Имеются следующие особенности:

- почти все команды могут оперировать 8/16/32-разрядными данными, и использовать 8/16/32-разрядные регистры;
- в качестве базовых и индексных регистров при обращении к памяти в командах с 32-разрядными данными теперь могут быть использованы любые регистры общего назначения, чего не было в предшествующих процессорах. (Замечание: регистр ESP не может быть индексным);
- при обращении к памяти теперь допускается масштабированное индексирование (т.е. содержимое индексного регистра при вычислении адреса может быть умножено на 2, 4 или 8). Это по сути является новым режимом адресации, который упрощает операции над многомерными массивами, исключая выполнение дополнительных операций сдвига или умножения;
- в большинстве команд при обращении к памяти можно использовать любой из шести сегментных регистров. (Регистры FS и GS впервые появились в процессоре 80386).

Если регистр EBP указывается как индексный регистр с масштабированием, то он не вызывает обращения к сегменту стека в отличие от использования его в качестве базового регистра. Регистр EBP считается базовым только в случае отсутствия у него масштабного коэффициента.

Примеры команд с использованием новых возможностей адресации:

```
MOV   EBX,[ECX]
ADD   EAX,GS:[EBX][ESI*4]+100h
DEC   BYTE PTR [EBP][EDX]          ; сегмент SS
NOT   BYTE PTR [EBP*8][EAX]        ; сегмент DS
```

2. Формат команды

Формат команды процессора 80386 является расширением формата процессора 8086. Рассмотрим особенности, характерные для процессора 80386.

Ниже описаны элементы команды в том порядке, как они расположены в команде (обязательным является лишь код операции):

1. Префикс: байт, предшествующий команде и модифицирующий операцию этой команды. Следующие префиксы могут быть использованы в прикладных программах:

- замена сегмента - в явной форме указывает, какой сегментный регистр должна использовать команда вместо регистра, принимаемого по умолчанию;.

- размер адреса - переключает разрядность адреса, определяя образование 32-разрядных или 16-разрядных адресов. Любой из этих размеров может быть выбираемым по умолчанию: данный префикс выбирает альтернативный размер. Если, например, размер эффективного адреса по умолчанию равен 16 битам, то наличие данного префикса заставляет команду вычислять 32-битный эффективный адрес. Код префикса - 67H;

- размер операнда - переключает разрядность операндов, устанавливая их 32-разрядными или 16-разрядными. Данный префикс также выбирает размер, альтернативный принимаемому по умолчанию. Код префикса - 66H.

Размер операндов и адресов по умолчанию в реальном и виртуальном режимах принимается процессором равным 16 битам. В защищенном режиме размер по умолчанию задается значением бита D в дескрипторе кодового сегмента;

- повторение - применяется с командами обработки строк; заставляет команду автоматически обрабатывать каждый элемент строки.

2. Код операции: описывает операцию, выполняемую командой.

3. Режим адресации: этот элемент, если он присутствует, описывает, является ли операнд содержимым регистра или ячейки памяти.

4. SIB (scale-index-base, масштаб-индекс-база) байт: когда описатель режима адресации указывает на использование индексного регистра для вычисления адреса операнда, SIB байт

используется для кодирования в команде базового регистра, индексного регистра и коэффициента масштабирования.

5. Смещение: представляет собой 8-, 16- или 32-разрядное целое число со знаком.

6. Непосредственный операнд: может быть 8-, 16 или 32-разрядным.

Для работы с процессорами 80286 и 80386 в ассемблер были введены новые директивы:

.286 - разрешает ассемблирование непривилегированных инструкций процессора 80286 (реальный режим) и инструкций сопроцессора 80287.

.286P - разрешает ассемблирование всех инструкций процессора 80286 (включая защищенный режим) и сопроцессора 80287.

.386 - разрешает ассемблирование непривилегированных инструкций процессора 80386 (реальный режим) и сопроцессора 80387.

.386P - разрешает ассемблирование всех инструкций процессора 80386 (включая защищенный режим) и сопроцессора 80387.

Кроме того, в директиве SEGMENT появилось поле "использование", которое задает используемый по умолчанию для сегмента размер операнда (если разрешена генерация кода процессора 386) и может быть USE16 или USE32. Для программ, работающих только в реальном режиме, необходимо при описании сегментов использовать USE16, иначе программа не будет работать корректно, поскольку процессор в реальном режиме использует всегда 16-разрядную длину по умолчанию.

В качестве примера рассмотрим простейшую программу для процессора 80386, которая очищает экран монитора путем вывода последовательности пустых символов непосредственно в видеопамять.

Последовательность выводится двойными словами. Атрибут сегмента USE16 задает размер операндов по умолчанию, далее ассемблер сам будет расставлять необходимые префиксы. Выводимое двойное слово записано в виде "атрибут-символ-атрибут-символ".

Программа транслируется в COM-файл:

```
      TASM <имя>
      TLINK <имя> /t

.386
CSEG    SEGMENT PARA USE16 'CODE'
        ASSUME  CS:CSEG,DS:CSEG,SS:CSEG
        ORG   100h
start:        cld ; установка направления
                  ; снизу вверх
      mov   ax, 0B800h; загрузка адреса видео-
      mov   es, ax     ; памяти в регистр ES
      mov   di, 00h    ; начальное смещение
      mov   eax, 07000700h; выводимое двойное слово
      mov   cx, 3E8h    ; количество повторений
      rep   stosd       ; пересылка двойного слова
                        ; в цикле
      int   20h         ;выход в DOS
CSEG  ENDS
END   Start
```

ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ

Составить подпрограмму, реализующую действия, соответствующие варианту задания, приведенному ниже. Номер варианта выбирается по порядковому номеру студента в списке группы. Для отладки подпрограммы составляется основная программа, определяющая необходимые структуры данных и вызывающая рабочую процедуру.

1. Написать процедуру, производящую поиск заданной цепочки бит в массиве. Адрес массива задается парой ES:BX, длина (в байтах) - регистром CX. Битовая строка записывается в регистр EAX, начиная с младшего бита, количество бит в строке указывается в регистре DL. Процедура возвращает в регистре DI значение 0FFFFh, если строка не найдена, или номер первого бита строки.

2. Написать подпрограмму, выводящую содержимое битовой строки на экран. Процедура просматривает каждый бит строки, и если он равен "1", печатает число "1", иначе - "0". Адрес

строки задается в паре регистров GS:DX, длина строки (в битах) - в регистре CX.

3. Написать подпрограмму, подсчитывающую количество установленных бит в битовой строке. Адрес строки задается в паре регистров ES:DX, длина строки (в битах) - в регистре ECX.

4. Написать процедуру, переписывающую битовую строку "задом наперед", т.е. меняющую порядок следования бит на противоположный. Адрес строки задается в паре регистров ES:DX, длина строки (в битах) - в регистре CX.

5. Написать подпрограмму, расширяющую битовую строку следующим образом: после нулевого бита вставляется единичный, и наоборот. Адрес исходной строки задается в паре регистров DS:BX, адрес результирующей строки - FS:DX, длина строки (в битах) – в регистре CX.

6. Даны два двоичных множества одинаковой длины, представленные в виде битовых строк. Написать процедуру, производящую в зависимости от значения регистра CL следующие действия: CL=0 - произведение, CL=1 - объединение, CL=2 - разность первого и второго множеств. Адрес первого множества - FS:DX, второго - GS:BX, длина - AX. Результат записывается на место первого множества.

7. Дан массив двойных слов. Написать подпрограмму, составляющую битовую строку из старших бит всех двойных слов. Адрес исходного массива задан парой ES:DX, длина (в двойных словах) - регистром CX, адрес результирующей битовой строки - парой FS:BX.

8. Написать подпрограмму, циклически сдвигающую массив на 4 бита влево. Адрес массива задан парой FS:BX, длина (в байтах) - регистром CX. При выполнении использовать команду двойного сдвига.

9. Задан массив и границы некоторой области памяти. Необходимо преобразовать массив следующим образом: просматривается по очереди каждый элемент массива, и если он лежит внутри заданных границ, то на его место записывается число "1", иначе - "0". Элементы массива - слова. Для проверки используется команда BOUND.

10. Написать подпрограмму, меняющую цвет на зеленый путем изменения битов в байтах атрибутов видеопамяти. При работе

процедура использует команды работы с битами непосредственно в памяти.

11. Написать программу, выводящую таблицу векторов прерываний на экран в виде Сегмент: Смещение для каждого из 256 векторов. Использовать базовую индексную адресацию с масштабированием для доступа к каждому вектору.

12. Написать программу, выводящую на экран названия и содержимое регистров процессора 32-битного процессора x86: EAX..EDI, CS..GS, CR0, CR2, CR3, EFLAGS, DR0..DR3, DR6, DR7.

13. Написать программу, выводящую на экран состояние регистра EFLAGS. Выводятся названия битов и их значение.

14. Написать подпрограмму, формирующую из заданной битовой строки новую путем удаления каждого второго бита (т.е. строка становится в два раза короче). Результат помещается на место исходной строки. Адрес строки задан парой GS:SI, длина (в битах) - регистром DX.

15. Написать подпрограмму, просматривающую битовую строку и заменяющую все имеющиеся последовательности "111" на "000". Адрес строки задается парой регистров FS:DI, длина-регистром ECX.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить основные сведения по работе и приложения 1, 2.

2. В соответствии с заданием составить основную программу и подпрограмму на языке ассемблера. В основной программе производятся подготовка параметров и вызов подпрограммы. Для контроля выполнения при необходимости используется отладчик.

3. Выполнить ввод, трансляцию, построение кода программы и получить результаты ее работы.

СОДЕРЖАНИЕ ОТЧЕТА ПО РАБОТЕ

1. Цель работы.
2. Текст задания и общая схема решения задачи.
3. Тексты программы и подпрограммы.
4. Результаты работы программы.
5. Выводы по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные недостатки организации памяти в реальном режиме.

2. Перечислите все регистры: а) процессора 80286;

б) процессора 80386;

3. Что такое прерывание и особая ситуация; чем они отличаются.

4. Какие новые команды появились в процессоре 80286 и 80386.

5. Как изменится содержимое регистров после выполнения команд: а) BTR AX,3 если AX = 0FFFFh

б) SHLD AX,BX,10 если AX = 1234h, BX = 0015h.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Брэй Б. Микропроцессоры Intel: 8086/8088…80486, Pentium: пер. с англ. СПб: BHV-Петербург, 2005. 1328 с

2. Зубков С.В. Ассемблер для DOS, Windows и UNIX. М.: ДМК-Пресс, 2013. 638 с.

3. Таненбаум Э. Архитектура компьютера: 5-е изд. пер. с англ. СПб.: Питер, 2013. 848 с.

4. Юров В.И. Assembler: учебник для вузов. СПб.: Питер, 2011. 640 с.

5. Юров В.И. Assembler: практикум: учеб. Пособие для вузов. СПб.: Питер, 2007. 400 с.

6. Андреева А.А. и др. Программирование на языке ассемблера в операционной системе Windows: лаб. практикум. Чебоксары: Изд-во Чуваш. ун-та, 2006. 104 с.

7. Андреева А.А. и др. Программирование микропроцессоров семейства Intel 80x86: лаб. практикум. Чебоксары: Изд-во Чуваш. ун-та, 1996. 144 с.

ПРИЛОЖЕНИЯ

1. Система команд

Все команды разделим условно на три группы:

I. Команды, работающие одинаково во всех процессорах. В эту группу входят команды, выполняющиеся процессорами старших моделей точно так же, как и процессором 8086. Они работают с теми же данными, не генерируют никаких особых ситуаций.

1. Команды двоично-десятичной арифметики:
   AAA: ASCII - коррекция после сложения;
   AAD: ASCII - коррекция перед делением;
   AAM: ASCII - коррекция после умножения;
   AAS: ASCII - коррекция после вычитания;
   DAA: десятичная коррекция после сложения;
   DAS: десятичная коррекция после вычитания.

2. Команды работы с регистром флагов:
   CLC: очистка флага переноса;
   CLD: очистка флага направления;
   CLI: очистка флага разрешения прерываний;
   CMC: инверсия флага переноса;
   STC: установка флага переноса;
   STD: установка флага направления;
   STI: установка флага разрешения прерываний;
   LAHF: чтение регистра флагов;
   SAHF: запись регистра флагов.

3. Команды повторений и организации циклов:
   LOOPcc: организация цикла;
   REPcc: повторение строковой операции.

4. Другие команды:
   CBW: преобразование байта в слово;
   CWD: преобразование слова в двойное слово;
   NOP: пустая операция;
   XLAT: табличное преобразование;
   HLT: останов. Эта команда в защищенном режиме может выполняться только на нулевом уровне привилегий.

II. Команды, получившие дополнительные возможности. В большинство команд процессора 80386 и выше, унаследованных от предыдущих процессоров были введены изменения и дополнения:

- почти все команды могут оперировать 32-битными регистрами;
- большинство команд при обращении к памяти могут пользоваться всеми шестью сегментными регистрами, использовать в качестве базовых и индексных регистров любые регистры общего назначения, использовать масштабированное индексирование. Рассмотрим кратко все такие команды.

1. Команды арифметических и логических операций, пересылок и сдвигов. Новым для команд этой группы является возможность оперирования 32-битными операндами. Например, там где раньше использовался регистр AX, теперь может быть использован EAX, где операндом являлось слово памяти, теперь можно использовать двойное слово и т.п. Отметим, что все старые возможности этих команд полностью сохранены. Перечислим все эти команды:

ADC: сложение с переносом;
ADD: сложение;
SBB: вычитание с заемом;
SUB: вычитание;
DEC: декремент;
INC: инкремент;
CMP: сравнение;
NEG: изменить знак;
TEST: побитное сравнение;
AND: логическое "И";
OR: логическое "ИЛИ";
XOR: логическое "Исключающее ИЛИ";
NOT: логическое "НЕ";
MOV: переслать данные;
XCHG: обмен;
RCL: циклический бит влево через флаг переноса;
RCR: циклический бит вправо через флаг переноса;
ROL: циклический сдвиг влево;
ROR: циклический сдвиг вправо;
SAL: арифметический сдвиг влево;
SAR: арифметический сдвиг вправо;

SHL: логический сдвиг влево;
SHR: логический сдвиг вправо;
DIV: беззнаковое деление;
IDIV: деление со знаком;
MUL: беззнаковое умножение;
LEA: загрузка эффективного адреса.

2. Строковые команды:
CMPS: сравнение строк;
LODS: загрузить строковый операнд;
MOVS: пересылка строк;
SCAS: сравнить строковый операнд;
STOS: сохранить строковый операнд.

Все эти команды могут использовать 32-разрядную адресацию (в этом случае используются индексные регистры ESI и EDI) и пересылать 32-разрядные операнды. Мнемоники 32-разрядных вариантов этих команд выглядят следующим образом:

CMPSD;
LODSD;
MOVSD;
SCASD;
STOSD.

3. Команды ввода-вывода:
IN: ввод из порта байта, слова или двойного слова;
OUT: вывод в порт байта, слова или двойного слова.

4. Команды условного перехода:
Jcc: переход, если выполнено условие cc;
JCXZ: переход, если CX = 0;
JECXZ: переход, если ECX = 0.

Смещение в этих командах может быть 8-,16- и 32-битным числом.

5. Команды работы со стеком. Все имевшиеся ранее стековые команды теперь могут работать с непосредственными операндами, с 32-разрядными операндами (кроме того, в процессоре 80386 добавлены новые команды):

PUSH: поместить операнд в стек;

PUSHF: поместить в стек регистр FLAGS;

PUSHFD: поместить в стек регистр EFLAGS;

POP: извлечь операнд из стека;

POPF: извлечь из стека регистр FLAGS;

POPFD: извлечь из стека регистр EFLAGS;

PUSHA: поместить все 16-разрядные регистры общего назначения в стек;

PUSHAD: поместить все 32-разрядные регистры общего назначения в стек;

POPA: извлечь все 16-разрядные регистры общего назначения из стека;

POPAD: извлечь все 32-разрядные регистры общего назначения из стека.

Команды PUSHA и POPA появились впервые в процессоре 80286, а команды PUSHAD, POPAD, PUSFD и POPFD – в процессоре 80386.

Команда PUSHA помещает регистры в стек в следующем порядке: AX, CX, DX, BX, SP, BP, SI, DI, а команда POPA извлекает их в обратном порядке (но не извлекает регистр SP). Команды PUSHAD и POPAD точно также работают с регистрами EAX, ECX, EDX, EBX, ESP, EBP, ESI и EDI.

6. Команды загрузки полных указателей. В процессоре 8086 имелось две таких команды – LDS и LES. В микропроцессоре 80286 все осталось по-прежнему, а в 80386 произошли следующие изменения:

– появились команды загрузки сегментных регистров SS, FS и GS (LSS, LFS и LGS соответственно).

– второй операнд любой из этих команд может быть как 16-, так и 32-разрядным регистром. Это значит, если атрибут размера операнда равен 32 битам, то из памяти читается 48-битный указатель (16 бит – в сегментный регистр и 32 бита – в регистр, указанный в команде).

7. Команда IMUL – знаковое умножение. Эта команда, кроме возможности работы с 32-разрядными данными, также может использоваться в следующих вариантах:

IMUL <регистр>,<непосредственный операнд> – результат помещается в регистр;

IMUL <регистр-приемник>, <регистр-множитель>, <непосредственный операнд-множитель>.

III. Новые команды процессоров 80286, 80386, 80486, Pentium. При описании особых ситуаций в некоторых командах использовано выражение "стандартные". К стандартным особым ситуациям здесь относится следующий набор:

– реальный режим: прерывание 13, если любая часть операнда находится вне пространства эффективных адресов от 0 до 0FFFFh.

– защищенный режим: особая ситуация 13 (общая защита), если результат находится в сегменте, запрещенном для записи, или если недопустимый эффективный адрес в сегментах CS, DS, ES, FS или GS; особая ситуация 12 (ошибка стека), если недопустимый адрес в сегменте SS; особая ситуация 14 (страничная ошибка) при страничной ошибке.

– виртуальный режим: то же, что и в реальном режиме; особая ситуация 12 при страничной ошибке.

При описании команд использованы следующие обозначения:

– r8: один из регистров AL, CL, DL, BL, AH, CH, DH, BH;

– r16: один из регистров AX, CX, DX, BX, SP, BP, SI, DI;

– r32: один из регистров EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI;

– imm8, imm16, imm32: непосредственная одно-, двух- или четырехбайтная величина соответственно;

– r/m8, r/m16, r/m32, r/m64: одно-, двух-, четырех- или восьмибайтный регистр или операнд в памяти;

– m8, m16, m32: байт, слово или двойное слово в памяти.

1. Команда BOUND (80286) – проверка нахождения индекса массива в заданных границах.

Синтаксис: BOUND r16, m32

BOUND r32, m64

Действие: команда позволяет убедиться в том, что знаковый индекс массива находится в пределах заданных границ. Первый операнд содержит проверяемый индекс массива. Второй операнд

адресует два слова (если атрибут размера операнда равен 16 бит) или два двойных слова (размер операнда 32 бита) памяти, первое из которых содержит нижнюю границу массива, а второе – верхнюю. Если в результате проверки выявится, что индекс массива не находится в пределах границ (включая сами границы), то будет сгенерирована особая ситуация 5, адрес возврата из которой указывает на команду BOUND. При удовлетворительном результате выполняется следующая команда.

Изменяемые флаги: нет. Особые ситуации: стандартные. Если в качестве второго операнда задан регистр, то генерируется особая ситуация 6 ("неопределенная операция").

Пример выполнения:

```
Array16 dw  1111h, 3333h
Array32 dd  55555555h, 88888888h
MOV AX, 2222h
BOUND AX, Dword Ptr Array16 ; нормально
MOV ECX, 33333333h
BOUND ECX, Qword Ptr Array32; особая
                            ; ситуация 5.
BOUND CX, EBX; особая ситуация 6.
```

2. Команды MOVZX и MOVSX – пересылка с расширением (80386).

Синтаксис :

MOV*X r16, r/m8
MOV*X r32, r/m8
MOV*X r32, r/m16

Действие: команды читают содержимое операнда-источника, который является байтом или словом, расширяют его до атрибута размера операнда (16 или 32 бита), и пересылают результат в операнд-приемник. Команды различаются только способом расширения операнда:

– MOVSX (MOVe with Sign eXtension) заполняет старшие биты знаковым битом источника;

– MOVZX (MOVe with Zero eXtension) заполняет старшие биты нулями.

Приемник (первый операнд) обязательно должен быть регистром общего назначения, а источник – операндом памяти или регистром.

Изменяемые флаги: нет. Особые ситуации: стандартные.

Пример выполнения:

```
MOV    AL, 11h
MOV    ECX, 12345678h
MOVZX  ECX, AL; ECX:=00000011h
MOV    BX, 0FF00h
MOVSX  EDX,  BX; EDX:=FFFFFF00h
```

3. Команды установки байта при условии (80386).

Синтаксис: SETcc r/m8, где cc – любое из стандартных условий процессора (E, NE, C, NC и т.д.).

Действие: команда записывает в операнд значение 1, если проверяемое условие выполнено, в противном случае операнд обнуляется.

Изменяемые флаги: нет. Особые ситуации: стандартные.

Пример выполнения:

```
MOV X, 1
DEC AX
SETZ CL ; CL := 1
SETC BH; BH := 0
```

4. Команды BSF и BSR (80386).

BSF (Bit Scan Forward) – сканирование битов вперед;

BSR (Bit Scan Reverse) – сканирование битов назад.

Синтаксис: BS* r16, r/m16

BS* r32, r/m32

Действие: команды предназначены для поиска позиции первого единичного бита в слове или двойном слове. Второй операнд просматривается побитно начиная с младших (BSF) или старших (BSR) разрядов. Номер первого встреченного единичного бита помещается в операнд-приемник и очищается флаг ZF. Если все разряды операнда-источника нулевые, то устанавливается флаг ZF.

Изменяемые флаги: ZF. Особые ситуации: стандартные.

Пример выполнения:

```
MOV BX, 0101110001010000b   ;
BSF  AX, BX          ; AX := 4
BSR  CX, BX          ; CX := 14
```

5. Команды проверки и модификации бита.

BT (Bit Test) – проверка бита;

BTC (Bit Test with Complement) – проверка и дополнение бита;

BTR (Bit Test with Reset) – проверка и сброс бита;

BTS (Bit Test with Set) – проверка и установка бита.

Синтаксис: BT* r/m16, r16

BT* r/m32, r32

BT* r/m16, imm8

BT* r/m32, imm8

Действие: команды проверяют значение бита в первом операнде, номер которого задан вторым операндом, заносят это значение в флаг CF и, производят инвертирование (BTC), сброс (BTR) или установку (BTS) проверяемого бита. Значение смещения внутри операнда (регистра или ячейки памяти) задается непосредственным 8-разрядным числом или регистром и при выполнении команды берется по модулю 32, что позволяет выбрать любой из 32 разрядов операнда. Если операнд 16-разрядный, то смещение берется по модулю 16. При косвенной адресации ячеек памяти операция взятия по модулю не производится.

Изменяемые флаги: CF. Особые ситуации: стандартные.

Примеры:

```
BT   ESI, BX
BTC  [DI], 15
BTR  AX, 2
    BTS  [EBX+EDI], DX
```

6. Команда CWDE – преобразовать слово в двойное слово (80386).

Синтаксис: CWDE

Действие: команда преобразует слово в регистре AX в двойное слово в регистре EAX путем расширения знакового бита регистра AX в два старших байта регистра EAX. Эта команда похожа на команду CWD, однако та использует пару регистров DX:AX в качестве приемника.

Изменяемые флаги: нет. Особые ситуации: нет.

Пример выполнения:

```
MOV EAX, 00008000h
CWDE; EAX := FFFF8000h
```

7. Команда CDQ – преобразовать двойное слово в учетвереннное слово (80386).

Синтаксис: CDQ.

Действие: команда преобразует двойное слово в регистре EAX в учетверенное слово (квадрослово) в паре регистров EDX:EAX путем копирования знакового бита регистра EAX во все разряды EDX. Эта команда имеет тот же код операции, что и команда CWD (преобразующая слово в регистре AX в двойное слово в паре DX:AX). Какая из этих команд будет выполняться, определяется 16 или 32-битным сегментом или присутствием префикса размера операнда.

Изменяемые флаги: нет. Особые ситуации: нет.

8. Команды SHLD и SHRD – логический сдвиг двойной точности (80386).

Синтаксис: SH*D r/m16, r16, imm8
SH*D r/m32, r32, imm8
SH*D r/m16, r16, CL
SH*D r/m32, r32, CL

Действие: SHLD (двойной сдвиг влево) сдвигает биты операнда-приемника (первый операнд) влево, заполняя пустые биты значениями битов, вытесняемых из операнда-источника.

SHRD (двойной сдвиг вправо) сдвигает биты приемника вправо, заполняя пустые биты значениями битов, вытесняемых из операнда-источника.

Результат запоминается в приемнике, а операнд-источник не изменяется. Количество битов, на которое выполняется сдвиг, может быть задано в регистре CL или непосредственно значением байта в команде. Когда выполняется сдвиг на ноль позиций, ни один из флагов не подвергается изменениям. В противном случае флагу CF присваивается значение последнего бита, вытесненного из операнда-приемника, и изменяются значения флагов SF, ZF и PF.

Особые ситуации: стандартные.
Пример выполнения:

```
MOV   AX, 1111111111111111b
MOV   BX, 0101010100000000b
SHLD AX, BX, 4; AX := 1111111111110101b
SHRD  AX, BX, 4; AX := 0000111111110101b
```

9. Команды INS и OUTS – ввод/вывод строк.
Синтаксис: INSB – прием байта из порта;
INSW – прием слова из порта;
INSD – прием двойного слова из порта;
OUTSB – вывод байта в порт;
OUTSW – вывод слова в порт;
OUTSD – вывод двойного слова в порт.

Действие: номер порта всегда задается регистром DX. В командах INS* значение, считанное из порта, пересылается по адресу ES:(E)DI. В командах OUTS* в порт пересылается значение из переменной памяти, адресуемой парой DS:(E)SI. После выполнения команды содержимое индексного регистра (DI или SI) изменяется на 1 (INSB и OUTSB), 2 (INSW и OUTSW) или 4 (INSD и OUTSD). Если флаг DF=1 то индексный регистр уменьшается, в противном случае – увеличивается. Командам может предшествовать префикс REP, для ввода/вывода блока размером CX.

Изменяемые флаги: нет. Особые ситуации: стандартные, кроме того:

– защищенный режим: особая ситуация 13, если текущий уровень привилегий ниже, чем уровень привилегий ввода/вывода, и любой из соответствующих битов разрешения ввода/вывода в TSS равен 1.

– виртуальный режим: особая ситуация 13, если любой из соответствующих битов разрешения ввода/вывода в TSS равен 1.

10. Команда ENTER – создать кадр стека для параметров процедуры. Синтаксис: ENTER imm16, imm8

Действие: команда создает кадр стека, используемый в большинстве блочно-структурированных языков высокого уровня при вызове процедур. Первый операнд определяет число

байт, выделяемых в стеке для процедуры, в которую осуществляется вход. Второй операнд – лексический уровень вложенности (от 0 до 31) процедуры внутри исходного кода. Он определяет количество указателей кадра стека, которые будут скопированы в новый кадр из предыдущего. В регистре (E)BP сохраняется адрес первого двойного слова в индикаторе кадра. Это слово является указателем предыдущего стекового кадра. Формально команду ENTER STOR, LEVEL можно представить следующим алгоритмом:

```
PUSH (E)BP;
TEMP:= ESP;
While Level <> 0 Then
    Begin
      EBP:= EBP - 4;
      PUSH [EBP]
      Level:= Level - 1;
    End;
    If Level <>0 Then   PUSH TEMP;
    EBP:= TEMP;
    ESP:= ESP - STOR;
```

Переданные процедуре параметры адресуются с помощью положительных смещений относительно EBP, а локальные переменные – с помощью отрицательных.

Изменяемые флаги: нет. Особые ситуации:

– защищенный режим: особая ситуация 12, если ESP или EBP превысит предел стека в любой точке во время выполнения команды;

– реальный режим: нет;

– виртуальный режим: нет.

Пример: ENTER 1024, 1

11. Команда LEAVE - освободить кадр стека.

Синтаксис: LEAVE

Действие: команда производит действия, обратные команде ENTER. Она копирует указатель кадра в указатель стека, тем самым освобождая пространство, использовавшееся для локальных переменных. Старый указатель кадра выбирается из

стека в регистр (E)BP, восстанавливая кадр вызывающей процедуры.

Изменяемые флаги: нет. Особые ситуации:

– защищенный режим: особая ситуация 12, если (E)BP не указывает на местоположение внутри текущего сегмента стека;

– реальный режим: прерывание 13, если любая часть операнда лежит вне пространства адресов от 0 до 0FFFFh;

– виртуальный режим: то же, что и в реальном режиме.

12. Команда BSWAP – перестановка байтов 32-битного регистра (80486).

Синтаксис: BSWAP r32

Действие: обращает порядок байтов в 32-битном регистре

Изменяемые флаги: нет. Особые ситуации: стандартные.

Пример выполнения:

```
MOV   EAX, 12345678h
BSWAP  EAX; EAX = 78563412h
```

14. Команда XADD – обменять и сложить (80486).

Синтаксис: XADD r/m8, r8
XADD r/m16, r16
XADD r/m32, r32

Действие: команда выполняет сложение, помещает второй операнд в первый, а результат – во второй.

Изменяемые флаги: по результату сложения ZF, SF, CF, OF, PF, AF. Особые ситуации: стандартные.

15. Команда CMPXCH – сравнить и обменять (80486).

Синтаксис: CMPXCH r/m8, r8
CMPXCH r/m16, r16
CMPXCH r/m32, r32

Действие: команда сравнивает значения, содержащиеся в AL, AX, или EAX (в зависимости от размера операндов) с первым операндом. Если они равны, то второй операнд копируется в первый и флаг ZF устанавливается в 1, иначе первый операнд копируется в AL, AX или EAX, и флаг ZF устанавливается в 0.

Изменяемые флаги: ZF. Особые ситуации: стандартные.

16. Команда CMPXCH8B – сравнить и обменять 8 байт (Pentium).

Синтаксис: CMPXCH8B m64

Действие: команда сравнивает содержимое регистров EDX_EAX с памятью. Если они равны, то содержимое регистров ECX_EBX копируется в память, иначе память копируется в регистры EDX_EAX.

Изменяемые флаги: ZF. Особые ситуации: стандартные.

17. Команда CMOVcc – условная передача данных (Pentium Pro).

Синтаксис: CMOVcc r16, r/m16

CMOVcc r32, r/m32, где cc – любое из стандартных условий процессора (E, NE, C, NC и т.д.).

Действие: команда копирует второй операнд в первый, если проверяемое условие выполнено; в противном случае ничего не выполняется.

Изменяемые флаги: нет. Особые ситуации: стандартные.

18. Команда CPUID – идентификация процессора (Pentium).

Синтаксис: CPUID.

Действие: команда сообщает информацию о производителе, типе, модификации и возможностях процессора (Pentium), внутренней кэш-памяти (Pentium Pro), уникальном номере процессора (Pentium III). Результат работы команды зависит от значения регистра EAX.

Изменяемые флаги: нет. Особые ситуации: стандартные.

Остальные новые команды относятся к защищенному режиму и рассмотрены в [7].