И.И. Шагурин
В.Б. Бродин
Г.П. Мозговой

80386
Описание и система команд

Москва
«Малип»
1992

ББК 32.973.2
Ш15
УДК 681.3

Шагурин И.И., Бродин В.Б., Мозговой Г.П.

Ш15 80386: описание и система команд.-М.:МП "Малип",1992.-160с.
ISBN 5-88547-004-9

Книга знакомит читателя с архитектурой и функционированием 32-разрядного микропроцессора Intel 80386. Описаны структура, способы адресации и система команд, организация защиты памяти, механизм прерываний, приведены временные диаграммы основных рабочих циклов.

Для специалистов в области цифровой техники, занимающихся применением высокопроизводительных микропроцессоров

Игорь Иванович Шагурин
Владимир Борисович Бродин
Георгий Павлович Мозговой

80386: описание и система команд

Редактор Т.Н.Шестакова
Художник Л.А.Степанов

Подписано к печати 30 .03.92. Формат 60×84/16.
Бумага офсетная. Печать офсетная. Усл.п.л. 10.
Заказ 90
МП "Малип", 129626, Москва, а/я 23.
При участии А/О "Таймер"
Отпечатано в типографии ИПО "Полиграф"
125438, Москва, Пакгаузное шоссе, д.1

ISBN 5-88547-004-9

Предисловие

Первые 32-разрядные микропроцессоры появились на мировом рынке в 1983-84гг., но их широкий прорыв в сферу высокопроизводительных вычислительных систем начался с 1985г., когда были выпущены модели 68020 фирмы Motorola и 80386 фирмы Intel. Эти БИС стали родоначальниками нового поколения микропроцессоров, которое характеризуется реализацией полного набора механизмов обработки информации, присущих ранее только "большим" ЭВМ. Соответствующей величины достигли размер адресуемой памяти, разрядность обрабатываемых данных, производительность, что позволило перевести такие микропроцессорные системы под управление многозадачных операционных систем. Указанные микропроцессоры послужили базовыми моделями для последующих разработок фирмы Motorola (68030, 68040) и Intel (80486). Наряду с оригинальными архитектурными особенностями в этих микропроцессорах использованы принципы, положенные в основу более ранних изделий (Motorola 68000, Intel 8086, 80286), что обеспечило совместимость программного обеспечения и многих аппаратурных решений.

Не вдаваясь в сравнительную оценку микропроцессоров Motorola и Intel, следует отметить, что обе эти фирмы занимают лидирующие позиции в микропроцессорной технике и имеют весьма широкую область применения своих изделий, поэтому знакомство со структурными решениями и особенностями функционирования последних моделей микропроцессоров этих фирм необходимо для каждого специалиста в области современной цифровой техники.

Настоящая книга знакомит читателя с архитектурой и функционированием микропроцессора 80386 (iAPX386). В настоящее время фирма Intel под этим общим названием выпускает две модели микропроцессоров : 386TMDX, имеющий 32-разрядные внутренние и внешние шины адреса и данных, и 386TMSX, имеющий практически такую же внутреннюю структуру, но 16-разрядную внешнюю шину данных и 24-разрядную шину адреса (аналогично микропроцессору 80286). В данной книге рассматривается модель 386TMDX, но большинство материала - структура микропроцессора, система команд,

организация защиты, прерываний и многое другое - полностью применимо и к 386TMSX.

Микропроцессор 80386 представляет собой сложное цифровое устройство, поэтому для его освоения требуется определенная предварительная подготовка. Желательно, чтобы читатель был знаком с базовой моделью - микропроцессором 8086 фирмы Intel (отечественный аналог КМ1810ВМ86). Этот микропроцессор описан в отечественной оригинальной и переводной литературе [1-5] и справочниках [6-8]. Многие особенности функционирования микропроцессора 80386, связанные с реализацией защиты памяти от несанкционированного доступа, реализованы в другой модели - 16-разрядном микропроцессоре 80286, архитектура которого описана в работе [9]. Предварительное знакомство с этим микропроцессором также будет весьма полезно.

В этой книге авторы старались при весьма ограниченном об'еме дать читателю достаточно полные сведения о структуре и особенностях функционирования микропроцессора 80386. Ввиду малого об'ема издания в нем не приведены примеры программирования микропроцессора, не рассмотрены вопросы построения на его основе цифровых систем с применением вспомогательных микросхем: сопроцессора 80387, контроллеров 82586, 82258, 8259, 8274 и других. С этими вопросами можно познакомиться в переводной работе [10]. Книга написана вице-президентом Международного компьютерного клуба, профессором Шагуриным И.И. (гл. 1,3,4,5), Бродиным В.Б. (гл. 1,2,6,7), Мозговым Г.П. (разд. 3.1, 5.1.1, 5.1.2). При написании книги авторами использованы материалы фирмы Intel [11,12], любезно предоставленные представителями фирмы на выставке "Комтек - 91"(Москва, апрель 1991г.). В связи с этим авторы выражают фирме Intel глубокую благодарность.

Глава 1. Архитектура микропроцессора 80386

Высокопроизводительный 32-разрядный микропроцессор 80386 ориентирован на эффективное выполнение программ в среде многозадачных операционных систем. Микропроцессор (МП) имеет 32-разрядные регистры и шины, которые обеспечивают хранение и передачу 32-разрядных адресов и данных. На кристалле БИС МП интегрирован диспетчер памяти, аппаратные средства многозадачности и механизма защиты для поддержки операционных систем. МП 80386 допускает одновременную работу нескольких операционных систем.

Конвейер команд, транслятор адресов на кристалле, высокая скорость передачи по магистрали - все это позволяет выполнять 3-4 млн. команд/с, обеспечивает хорошие системные параметры.

Необходимо отметить следующие важные свойства, характеризующие МП 80386:

* Гибкость структуры, включающей восемь 32-разрядных регистров общего назначения (РОН), которая позволяет оперировать с 8-, 16-, 32-разрядными данными.
* Физическое адресное пространство 4Гбайт, виртуальное - 64Тбайт(2^{46}), максимальный размер сегмента 4Гбайт.
* Наличие встроенного диспетчера памяти, обеспечивающего поддержку виртуальной памяти, страничную организацию памяти средствами МП, четыре уровня защиты памяти.
* Полная совместимость с МП 80286, совместимость по коду с семейством 8086.
* Наличие виртуального режима МП 8086, позволяющего выполнять программы этого микропроцессора с использованием защиты и страничной организации памяти.
* Тактовая частота до 33МГц, скорость обмена по магистрали при этой частоте - 66Мбайт/с.
* Высокая скорость выполнения арифметических операций при работе с сопроцессором 80387.

Тестовые возможности МП 80386 включают самотестирование и прямое обращение к кэш-памяти трансляции страниц. Четыре регистра прерываний обслуживают условные и безусловные прерывания по выполняемому коду или выбираемым

данным, что обеспечивает эффективную отладку систем, в том числе включающих ПЗУ.

Поскольку объектный код совместим с кодом микропроцессоров семейства 8086 (8086, 8088, 80186, 80188, 80286), то для МП 80386 может использоваться большой объем накопленного программного обеспечения.

1.1. Структура и функционирование микропроцессора

Микропроцессор 80386 включает подсистемы центрального процессора, выборки команд, диспетчера памяти и интерфейса магистрали (рис. 1.1).

Центральный процессор состоит из операционного блока и устройства управления. Операционный блок включает арифметико-логическое устройство (АЛУ) и восемь 32-разрядных регистров общего назначения. Особенностью АЛУ является наличие 64-разрядного сдвигателя, используемого при быстрых арифметических и циклических сдвигах, операциях умножения и деления. Алгоритм умножения останавливает итерацию, когда старшие значащие разряды становятся нулевыми, что позволяет выполнить обычное 32-разрядное умножение меньше, чем за одну микросекунду.

Подсистема выборки команд реализует двухступенчатый алгоритм конвейеризации, состоит из модулей предвыборки кодов и преддешифрации команд. Первый из них осуществляет заполнение очереди кодов длиной 16 байт, причем выборка байтов из памяти производится в промежутках между магистральными циклами команд. Во втором модуле производится преддешифрация, определяется тип и формат команд, выделяется поле относительного смещения, содержимое которого поступает в блок сегментации для вычисления линейного адреса. Команды, подготовленные к выполнению, хранятся в очереди команд, куда помещается в среднем три команды.

Диспетчер памяти (MMU-memory management unit) состоит из блока сегментации и блока управления страницами, осуществляет двухступенчатое формирование физического адреса ячейки памяти. Его наличие определяет два режима работы МП 80386 - режим реальных адресов (реальный режим)

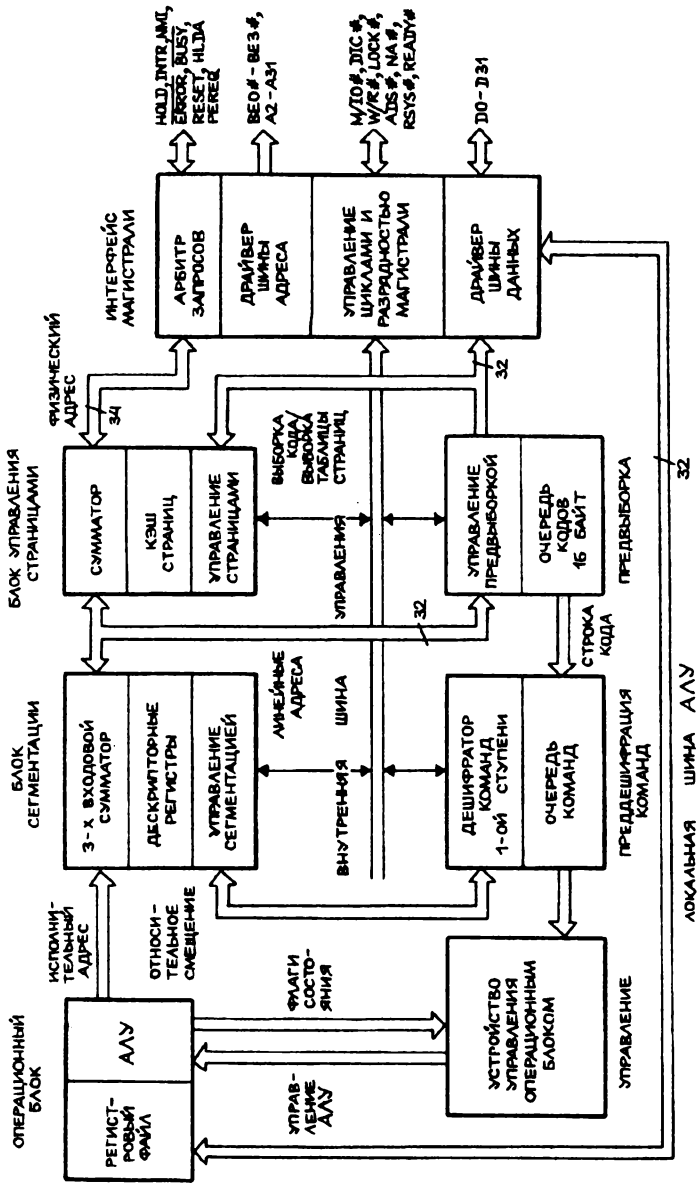


Рис.1.1. Структура МП 80386

и режим защищенных виртуальных адресов (защищенный режим). В реальном режиме МП 80386 работает как очень быстрый 8086, но при необходимости с расширениями разрядности операндов и адресов до 32. В защищенном режиме могут осуществляться переключения и выполнение нескольких задач, предназначенных для режима виртуального МП 8086. Каждая задача (прикладная программа или операционная система) работает с семантикой 8086, задачи изолированы и защищены одна от другой и от операционной системы инструментального МП 80386.

Сегментация является средством управления пространством логических адресов, обеспечивает мобильность программ. Сегментированная память представляет собой набор логических блоков, характеризуемых определенными атрибутами, такими, как расположение, размер, тип (стек, программа, данные), характеристика защиты. В системе на основе МП 80386 каждой задаче доступны до 16387 сегментов величиной до 4Гбайт каждый. Следовательно обеспечивается до 64Тбайт виртуальной памяти для каждой задачи.

Блок управления страницами действует на более низком уровне, разбиение памяти на страницы возможно только в защищенном режиме. Каждый сегмент делится на страницы фиксированного размера по 4Кбайта каждая.

Наличие блока сегментации и блока управления страницами, их одновременное функционирование обеспечивают максимальную гибкость проектируемой системы. Сегментация полезна для организации в памяти локальных модулей и является инструментом программиста, в то время как страницы нужны системному программисту для распределения физической памяти системы.

Подсистема интерфейса магистрали реализует протоколы обмена МП 80386 с памятью, арифметическим сопроцессором, контроллерами ввода/вывода и другими устройствами. Обмен осуществляется с помощью 32-разрядной двунаправленной шины данных, 34-разрядной шины адреса и 16-разрядной шины управления.

Особенностью шины данных является возможность динамического изменения ее разрядности. За один цикл могут быть переданы 8, 16 или 32 бита.

По шине адреса передаются 32-разрядные адреса, она включает тридцать адресных выводов (A31-A2) и четыре вывода выбора байтов (BE3#-BE0#). Сигналы выбора байтов определяют, какие байты 32-разрядной шины данных участвуют в текущем цикле обмена. Это позволяет без дополнительной аппаратуры согласовать 32-разрядную шину данных с байтной организацией памяти системы.

Шина управления включает восемь выводов, сигналы которых управляют циклами магистрали, и восемь выводов управления состоянием процессора, взаимодействием с другими активными устройствами магистрали.

Магистраль МП 80386 характеризуется восемью типами магистральных циклов, причем для улучшения системных характеристик предусмотрено конвейерное формирование адресов. Циклы магистрали могут перекрываться, что увеличивает время обмена с памятью или устройством ввода/вывода.

1.2. Регистры

Набор регистров МП 80386 включает:
- -регистры общего назначения;
- -сегментные регистры;
- -указатель команд и регистр флагов;
- -регистры управления;
- -регистры адреса системы;
- -регистры отладки;
- -регистры тестирования.

Все 16-разрядные регистры микропроцессоров МП 8086, 80186 и 80286 содержатся в 32-разрядных регистрах МП 80386. На рис. 1.2 приведены основные регистры первых трех групп. Их содержимое определяется текущей задачей, т.е. в эти регистры автоматически загружается новое значение при операции переключения задач.

1.2.1. Регистры общего назначения

Восемь 32-разрядных регистров общего назначения для хранения данных и адресов (рис. 1.2) имеют имена EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP. ESP. Они поддерживают работу

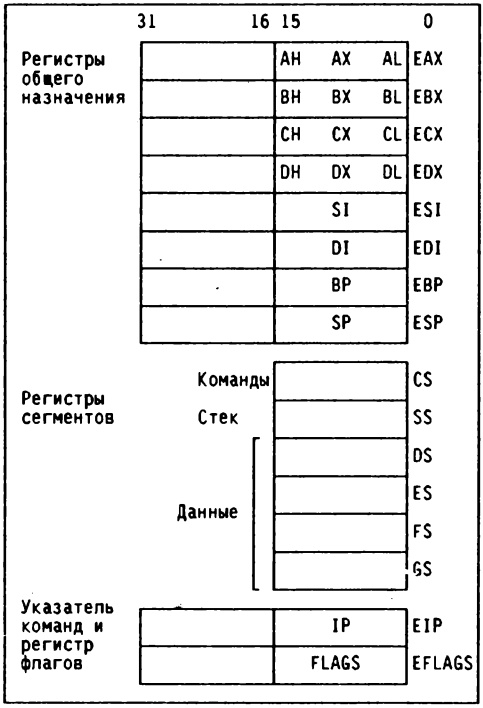


Рис.1.2. Основные регистры МП 80386

с операндами в 1, 8, 16, 32 и 64 бита и с битовыми полями от 1 до 32 бит. Адреса имеют размер 16 и 32 бита. Младшие 16 разрядов этих регистров доступны отдельно при использовании имен AX, BX, CX, DX, SI, DI, BP, SP. При операциях с байтами можно отдельно обращаться к младшему байту (разряды 0-7) и старшему байту (разряды 8-15) регистров AX, BX, CX и DX. Младшие байты имеют имена AL, BL, CL, DL, старшие байты - AH, BH, CH, DH. Доступ к отдельным байтам дает дополнительную гибкость при операциях с данными.

1.2.2. Регистры сегментов и дескрипторов сегментов

Шесть 16-разрядных сегментных регистров CS, SS, DS, ES, FS, GS содержат значения селекторов сегментов, указывающих на текущие адресуемые сегменты памяти. С каждым из них связан недоступный программно регистр дескриптора сегмента (рис. 1.3). В защищенном режиме каждый сегмент может иметь размер от 1 байта до 4Гбайт, в режиме реальных адресов максимальный размер сегмента составляет 64Кбайта. Селектор в CS указывает текущий сегмент кода команд, селектор в SS указывает текущий сегмент стека, селекторы в DS, ES, FS, GS - текущие сегменты данных. Каждый регистр дескриптора содержит 32-разрядный базовый адрес сегмента, 32-разрядную границу сегмента, другие необходимые атрибуты. Когда в регистр сегмента загружается новое значение селектора, содержимое соответствующего регистра дескриптора автоматически корректируется. В режиме реальных адресов непосредственно обновляется только базовый адрес (он получается путем сдвига значения селектора на 4 разряда влево), так как максимальный размер и атрибуты сегмента в реальном режиме фиксированы.

В защищенном режиме корректируются все параметры. При каждом обращении к памяти регистр дескриптора сегмента, связанный с выбранным сегментом, автоматически вовлекается в эту операцию. Базовый адрес сегмента (32 разряда) становится компонентой вычисления линейного адреса, 32-разрядная граница используется при операции контроля размера, атрибуты проверяются на соответствие типа памяти.

1.2.3. Указатель команд

Указатель команд (рис. 1.2) представляет собой 32-разрядный регистр с именем EIP, содержимое которого используется в качестве смещения при определении адреса следующей выполняемой команды. Смещение задается относительно базового адреса сегмента кода (CS). Младшие 16 бит (биты 0-15) EIP содержат 16-разрядный указатель команд с именем IP, который используется при 16-разрядной адресации.

Рис.1.3. Регистры сегментов и соответствующие регистры дескрипторов МП 80386

1.2.4. Регистр флагов

Регистр флагов является 32-разрядным, имеет имя EFLAGS (рис. 1.4.). Его разряды содержат признаки результата выполнения команды, управляют обработкой исключений и маскируемых прерываний, последовательностью прерываемых и вызываемых задач, вводом/выводом, переключением в режим виртуального МП 8086. Младшие 16 бит (биты 0-15) EFLAGS представляют 16 - разрядный регистр флагов FLAGS, используемый при

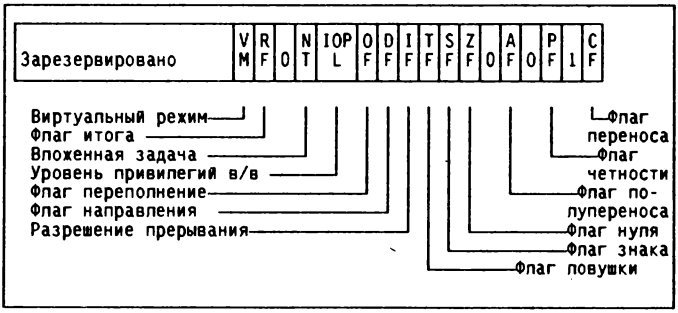


Рис.1.4. Регистр флагов

выполнении кода МП 8086 и 80286. Значение отдельных битов следующее:

VM (виртуальный режим 8086, бит 17). Включает виртуальный режим МП 8086 внутри защищенного режима МП 80386. Если этот бит устанавливается во время работы МП 80386 в защищенном режиме, то последний переключается в режим виртуального МП 8086, обрабатывая загрузку сегментов как МП 8086, но генерируя исключение 13 для команд, нарушающих привилегии. Бит VM может быть установлен только в защищенном режиме командой IRET (если текущий уровень привилегий равен 0) и переключением задачи при любом уровне привилегий. Значение бита VM не изменяется при выполнении команды POPF; команда PUSHF всегда заносит 0 в этот бит, даже в виртуальном режиме МП 8086. Значение EFLAGS, сохраняемое при прерываниях и переключениях задач, будет содержать 1 в этом разряде, если прерванная задача выполнялась как виртуальная МП 8086.

RF (флаг итога, бит 16). Используется вместе с точками останова регистров отладки и в пошаговом режиме, проверяется между командами перед прерыванием. Если этот бит установлен, то любая ошибка отладки в следующей команде игнорируется. Бит RF автоматически сбрасывается при успешном завершении каждой команды, сигнализируя об отсутствии ошибки. Исключение составляют команды IRET, POPF, JMP, CALL и INT, вызывающие переключение задач. Перечисленные команды устанавливают значение RF в соответствии с образом EFLAGS, вызываемым из памяти. Например, в конце подпрограммы обслуживания прерывания команда IRET может вызвать образ EFLAG, в котором бит RF установлен, и завершить выполнение программы с адреса прерывания без генерации ошибки в следующей контрольной точке.

NT (вложенная задача, бит 14). Флаг применяется в защищенном режиме. Бит NT устанавливается для указания на то, что текущая задача вложена в некоторую другую задачу и TSS (сегмент состояния) данной задачи имеет правильную обратную связь с TSS предыдущей задачи. Этот бит устанавливается или сбрасывается при передаче управления в другую задачу. Значение флага NT в регистре EFLAGS проверяется командой IRET для того, чтобы определить тип возврата (межзадачный или внутризадачный). Команды POPF и

IRET производят установку этого бита при любом уровне привилегий в соответствии с выгружаемым из стека значением EFLAGS.

IOPL (уровень привилегии ввода/вывода, биты 12-13). Эти два бита применяются в защищенном режиме. IOPL указывает максимальную величину CPL (текущего уровня привилегий), разрешающую выполнение команд ввода/вывода без генерации исключения 13. Он также показывает максимальную величину CPL, позволяющую изменить бит IF (флаг разрешения прерывания), когда новое значение выгружается из стека в регистр EFLAGS. Команды POPF и IRET могут в ходе выполнения изменять поле IOPL, если CPL=0. Переключение задач всегда изменяет поле IOPL, новое значение загружается из TSS прибывающей задачи.

OF (флаг переполнения, бит 11). Бит устанавливается, если операция приводит к переполнению знакового разряда. Это имеет место, когда при операции происходит перенос/заем в разряд знака (старший бит) результата, а переноса/заема из старшего бита нет, или наоборот. Для 8-, 16-, 32-разрядных операций флаг OF устанавливается при переполнении битов 7, 15, 31 соответственно.

DF (флаг направления, бит 10). Указывает на уменьшение или увеличение содержимого регистров ESI и/или EDI при выполнении команд обработки строк. Постувеличение имеет место, если бит DF сброшен; постуменьшение, если DF установлен.

IF (флаг разрешения прерывания INTR, бит 9). В установленном состоянии (IF=1) разрешает обрабатывать запросы внешних прерываний на входе INTR. Когда этот флаг сброшен(IF=0), такие запросы не воспринимаются.

TF (флаг ловушки, бит 8). Управляет генерацией прерывания типа исключение 1 при пошаговом выполнении. Если этот бит установлен, МП 80386 генерирует прерывание исключения 1 после выполнения команды. Когда бит сброшен, это прерывание имеет место только в точках останова, адреса которых загружены в регистры DR0-DR3.

SF (флаг знака, бит 7). Флаг устанавливается, если старший бит результата равен 1, в противном случае сбрасывается. Для 8-, 16-, 32-разрядных операций SF отражает значение битов 7, 15, 31 соответственно.

ZF (флаг нуля, бит 6). Флаг устанавливается, если все разряды результата равны 0, в противном случае сбрасывается.

AF (флаг полупереноса, бит 4). Используется для сложения и вычитания величин, представленных в двоично-десятичном коде (BCD). Флаг устанавливается, если результат операции приводит к переносу из разряда 3 (сложение) или заема в разряд 3 (вычитание), в противном случае он сбрасывается. На значение AF влияет только перенос или заем в/из разряд 3, независимо от общей длины операнда: 8, 16, 32 бита.

PF (флаг четности, бит 2). Флаг устанавливается, если младшие восемь разрядов операнда содержат четное число 1. Значение PF является функцией только младших восьми разрядов, независимо от величины операнда.

CF (флаг переноса, бит 0). Флаг устанавливается, если операция приводит к переносу (сложение), или заему (вычитание) в старший разряд, в противном случае CF сбрасывается. Для 8-, 16-, 32-разрядных операций CF устанавливается при переносе/заеме в разряды 7, 15, 31 соответственно.

1.2.5. Регистры управления

МП 80386 имеет три 32-разрядных регистра управления (CR0, CR2, CR3) для фиксации общего состояния процессора. Эти регистры вместе с регистрами системных адресов хранят информацию о состоянии МП, которое затрагивает все задачи системы. Для доступа к регистрам управления определены специальные команды загрузки и сохранения.

Регистр CR0: регистр управления машины. Содержит (рис.1.5) шесть разрядов для управления и определения состояния МП. Младшие 16 разрядов также называют словом состояния машины (MSW) для совместимости с защищенным режимом МП 80286. В командах загрузки и сохранения LMSW, SMSW для совместимости с МП 80286 используются только младшие 16 бит регистра CR0. Отдельные биты CR0 имеют следующее назначение:

PG (включение управления страницами, бит 31). Бит PG устанавливается для включения устройства управления страницами, сбрасывается для блокировки этого устройства.

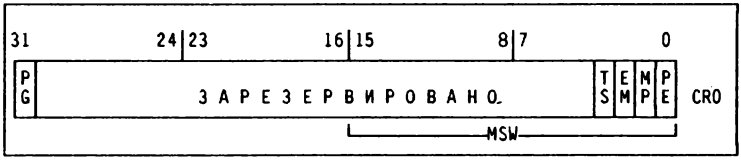


Рис.1.5. Регистр управления CR0

TS (переключение задачи, бит 3). Устанавливается автоматически, когда выполняется операция переключения задачи. Если TS установлен, код команды сопроцессора будет вызывать прерывание по исключению 7 - сопроцессор отсутствует, если бит MP=1. Обычно процедура обработки этого прерывания сохраняет состояние сопроцессора 80387, относящееся к предыдущей задаче, загружает в сопроцессор состояние текущей задачи, очищает бит TS перед возвратом к коду операции сопроцессора.

EM (эмуляция сопроцессора, бит 2). Устанавливается для того, чтобы все коды команд сопроцессора вызывали прерывание по исключению 7. Если EM сброшен, то команды сопроцессора выполняются реальным сопроцессором 80387 (режим по умолчанию после сброса). Отметим, что на команду WAIT установка бита EM не влияет.

MP (управление сопроцессором, бит 1). Используется вместе с битом TS для генерации прерывания по исключению 7 при идентификации кода команды WAIT. Если MP=1 и TS=1, код команды WAIT вызывает прерывание. Заметим, что TS автоматически устанавливается вне зависимости от операции переключения задач.

PE (включение защиты, бит 0). Устанавливается для переключения МП 80386 в защищенный режим. Если PE сброшен, процессор работает в реальном режиме. Бит PE может быть установлен загрузкой MSW или CR0; сбрасывается только загрузкой CR0. Заметим, что для совместимости с МП 80286 бит PE не может быть сброшен командой LMSW.

Регистр CR1 зарезервирован.

Регистр CR2: линейный адрес ошибки страницы. Регистр CR2 (рис. 1.6) содержит 32-разрядный линейный адрес, который вызвал последнюю ошибку страницы. Код ошибки загружается в

стек обработки ошибок страниц и обеспечивает дополнительную информацию об ошибке страницы.

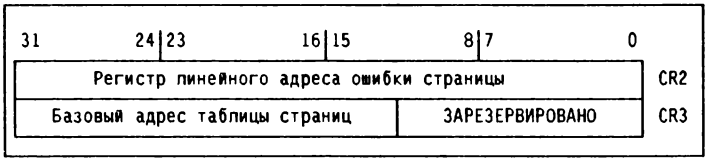


Рис.1.6. Регистры управления CR2 и CR3

Регистр CR3: базовый адрес каталога страниц. CR3 (рис. 1.6) содержит физический базовый адрес таблицы каталога страниц. Так как размер страниц у МП 80386 фиксирован (4Кбайта), младшие 12 бит регистра CR3 игнорируются при записи и являются неопределенными при сохранении.

Переключение задачи через TSS, которое изменяет значение в CR3, или прямая загрузка любого значения в CR3 вызовет нарушения всех входов таблицы страниц в кэш-памяти.

1.2.6. Регистры системных адресов

Регистры системных адресов GDTR, IDTR, LDTR и TR (рис. 1.7) служат для обращения к элементам, которые управляют механизмом сегментации памяти.

Такими элементами являются таблицы и сегменты, входящие в структуру защиты памяти МП 80286/80386:

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 линейные адреса базы и 16-разрядные величины границ GDT и IDT, которые являются глобальными по отношению ко всем задачам.

Регистр таблицы локальных дескрипторов LDTR и регистр задачи TR содержат 16-разрядные селекторы сегментов LDT и TSS, которые определены для конкретной задачи. С каждым из

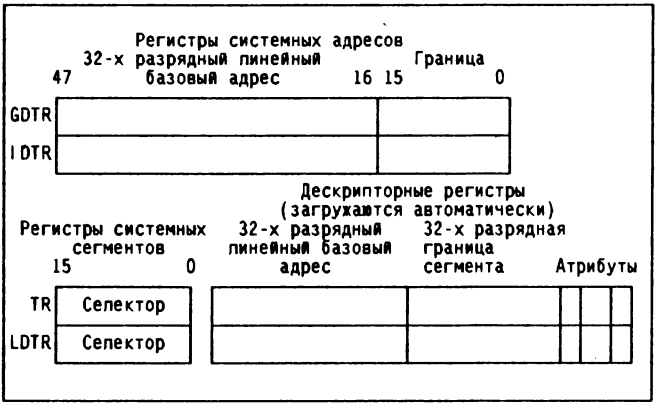


Рис.1.7. Регистры системных адресов и системных сегментов

них связан программно недоступный регистр дескриптора сегмента.

1.2.7.Регистры отладки и тестирования

МП 80386 имеет имеет десять регистров тестирования и отладки (рис. 1.8), из которых два регистра отладки DR4 и DR5 зарезервированы.

Шесть программно доступных регистров DR0-DR3, DR6, DR7 поддерживают процесс отладки внутри МП. Регистры отладки DR0-DR3 устанавливают четыре точки останова (содержат их линейные адреса). Регистр управления отладкой DR7 используется для установки контрольных точек, а регистр DR6 показывает текущее состояние микропроцессора при остановах.

Два регистра, TR6 и TR7, применяются для управления проверкой ассоциативной памяти и запоминающего устройства с произвольной выборкой в буфере ассоциативной трансляции. TR6 является командным регистром тестирования, а TR7 - регистром данных, который содержит данные для тестирования буфера трансляции.

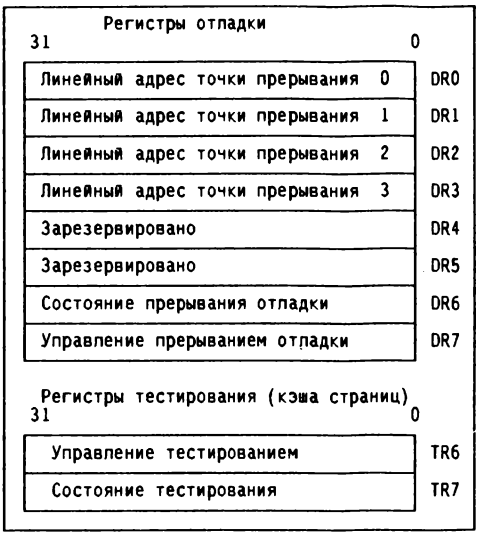


Рис.1.8. Регистры тестирования и отладки

1.3. Организация памяти и режимы работы

МП 80386 оперирует с физической памятью об'емом до 4Гбайт. Каждый байт памяти имеет свой физический адрес - от 0 (00000000H) до 2^{32}-1 (FFFFFFFFH). В памяти МП 80386 хранятся 8-разрядные байты, 16-разрядные слова и 32-разрядные двойные слова. Слова содержатся в двух смежных байтах памяти, а двойные слова - в четырех смежных байтах, причем младший байт располагается в ячейке с меньшим адресом, а старший байт в ячейке с большим адресом. Адресом слова или двойного слова является адрес младшего байта.

МП 80386 реализует два метода управления памятью - сегментирование и разбиение на страницы. Механизмом верхнего уровня является сегментирование; МП 80386 может работать с 16384 отдельными сегментами памяти об'емом до 4Гбайт каждый. Так как адресуемая физическая память содержит не более 4Гбайт, то полностью использовать все сегменты микропроцессор может только при использовании виртуальной памяти об'емом до

2^{46}=64Тбайт. Для обращения к байту сегментированной памяти вычисляется логический адрес. Он состоит из селектора, задающего базовый адрес сегмента (начало), и относительного адреса байта в сегменте. Арифметическое сложение базового и относительного адреса дает линейный адрес байта. Правила формирования относительного адреса зависят от метода адресации и описаны в главе 3. Правила определения базового адреса сегмента зависят от режима работы микропроцессора.

МП 80386 способен работать в двух режимах: режиме реальных адресов (реальный режим) и в режиме защищенной памяти (защищенный режим).

При работе в реальном режиме микропроцессор выполняет программы, написанные для МП 8086, 8088, 80186, 80286. При этом МП 80386 можно рассматривать как быстродействующий МП 8086, имеющий расширенный набор команд. При использовании префикса, изменяющего разрядность адреса, МП 80386 в реальном режиме может формировать 32-разрядные относительные адреса. Однако, если значение этого адреса превышает 65535, имеет место прерывание по нарушению границы сегмента. Линейный адрес в реальном режиме формируется также, как в микропроцесоре 8086. При этом 16-разрядный селектор, сдвигающийся влево на четыре разряда, дает значение 20-разрядного базового адреса сегмента. Младшие четыре разряда базового адреса заполняются нулями. Линейный адрес образуется посредством сложения 20-разрядного базового адреса и 16-разрядного относительного адреса, формируемого в соответствии с указанным методом адресации (разд. 3.2). Так как в реальном режиме страничная организация памяти не используется, то полученный 21-разрядный линейный адрес выдается микропроцессором как физический для выборки соответствующего байта.

После сигнала RESET МП 80386 автоматически начинает работу в реальном режиме, выполняя программу инициализации аналогично микропроцессору 8086. Затем МП 80386 может быть приведен в защищенный режим посредством загрузки в регистр управления CR0 слова состояния, в котором установлен бит включения защиты: PE=1. Загрузка слова состояния производится командой MOV или LMSW.

В защищенном режиме возможности МП 80386 раскрываются наиболее полно - появляется возможность

многозадачной обработки информации, защиты памяти с помощью четырехуровнего механизма привилегий и ее страничной организации.

Многозадачность реализуется с помощью операционной системы, которая распределяет имеющиеся ресурсы - микропроцессора 80386, памяти, устройств ввода-вывода - и их использование во времени для наиболее эффективного выполнения нескольких заданий. При этом каждая задача выполняется как бы отдельным процессором, процессоры называются виртуальными. Функционально виртуальные процессоры соответствуют микропроцессорам 8086/186, 80286 или 80386. Таким образом, при многозадачной обработке МП 80386 работает как несколько виртуальных микропроцессоров, имеющих общую память. Переходя с задачи на задачу МП 80386 может менять тип функционирования, например, переключаясь из режима виртуального 8086 на 80386 или 80286. Планирование выполнения задач операционной системой приводит к формированию для каждой задачи соответствующего сегмента состояния TSS. Этот сегмент хранится в памяти и содержит данные, необходимые для начала функционирования виртуального процессора: содержимое сегментных регистров, регистров общего назначения, указателя команд и др. Выбор TSS производится с помощью регистра задачи TR. Для переключения задач операционная система выдает команды перехода JMP или вызова CALL, которые обеспечивают переход к сегменту TSS новой задачи. При необходимости последующего возврата к решению предыдущей задачи предусмотрена возможность сохранения содержимого регистров при переключении задач. Таким образом допускается вложение задач, что фиксируется установкой признака NT=1 в регистре EFLAGS.

При работе МП 80386 в качестве виртуального МП 8086 устанавливается признак VM=1 в регистре EFLAGS. Этот признак может быть установлен при переключении задач, когда загружается новое содержимое регистра EFLAGS, или с помощью команды возврата из обслуживания прерывания IRET, когда содержимое EFLAGS восстанавливается из стека. При работе в качестве виртуального МП 8086 (режим V86) МП 80386 формирует 20-разрядный линейный адрес также, как в реальном режиме. Таким образом МП 80386 в режиме V86 работает как МП 8086, но обеспечивает, в случае необходимости, страничную

организацию памяти и защиту системных программ МП 80386 от исполняемых пользовательских программ МП 8086 (двухуровневый механизм привилегий).

Реализация защиты и страничной организации памяти описаны в главе 5.

Разрядность обрабатываемых операндов и адресов определяется режимом работы МП 80386. Если программы выполняются в реальном режиме или режиме V86, то по умолчанию используются 16-разрядные 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.

Если команде предшествуют определенные префиксы, то при ее выполнении принятая по умолчанию разрядность операнда или адреса изменяется. В результате разрядность операндов и адресов для МП 80386 в защищенном режиме определяется в соответствии с табл. 1.1. Для реального режима и режима V86 выбор разрядности определяется при значениях D=0.

Таблица 1.1. Разрядность операндов и адресов защищенном режиме

Бит D-разрядности по умолчанию	0	0	0	0	1	1	1	1
Префикс разрядности операнда (66H)	-	-	+	+	-	-	+	+
Префикс разрядности операнда (67H)	-	+	-	+	-	+	-	+
Разрядность операнда	16	16	32	32	32	32	16	16
Разрядность адреса	16	32	16	32	32	16	32	16
+ наличие префикса; - отсутствие префикса								

Обработка 8-разрядных байтов вместо 16- или 32-разрядных слов производится, если бит w в коде команды имеет значение w=0 (разд. 3.1).

1.4. Типы данных

МП 80386 обеспечивает обработку следующих типов данных.

Бит: одноразрядное двоичное число - 0 или 1.

Поле бит: последовательность, содержащая до 32 бит, покрывает максимум четыре байта.

Строка бит: множество следующих один за другим бит, которое может достигать длины до 4 Гбит.

Байт со знаком: 8-разрядная величина со знаком.

Байт без знака: 8-разрядная величина без знака.

Слово со знаком: 16-разрядная величина со знаком.

Слово без знака: 16-разрядная величина без знака.

Двойное слово со знаком: 32-разрядная величина со знаком. Все операции производятся в дополнительном коде.

Двойное слово без знака: 32-разрядная величина без знака.

Учетверенное слово со знаком: 64-разрядная величина со знаком.

Учетверенное слово без знака: 64-разрядная величина без знака.

Относительный адрес: 16- или 32-разрядное число, определяющее относительное положение ячейки памяти.

Указатель: логический адрес, состоящий из 16-разрядного селектора сегмента и 16- или 32-разрядного относительного адреса.

Символ: байт, представляющий букву, цифру или служебный символ в коде ASCII.

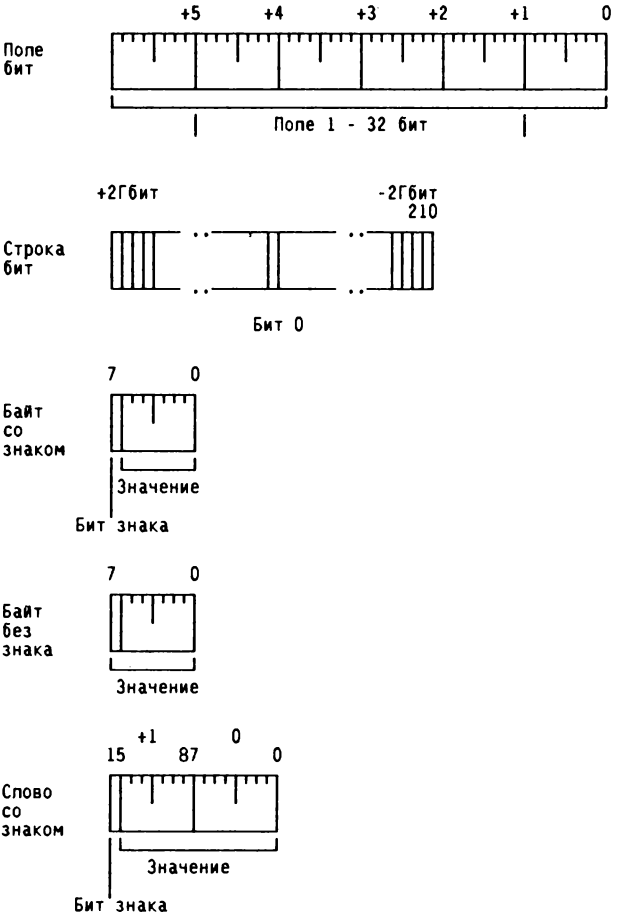
Строка: непрерывная последовательность байтов, слов или двойных слов. Строка может содержать от 1 байта до 4 Гбайт.

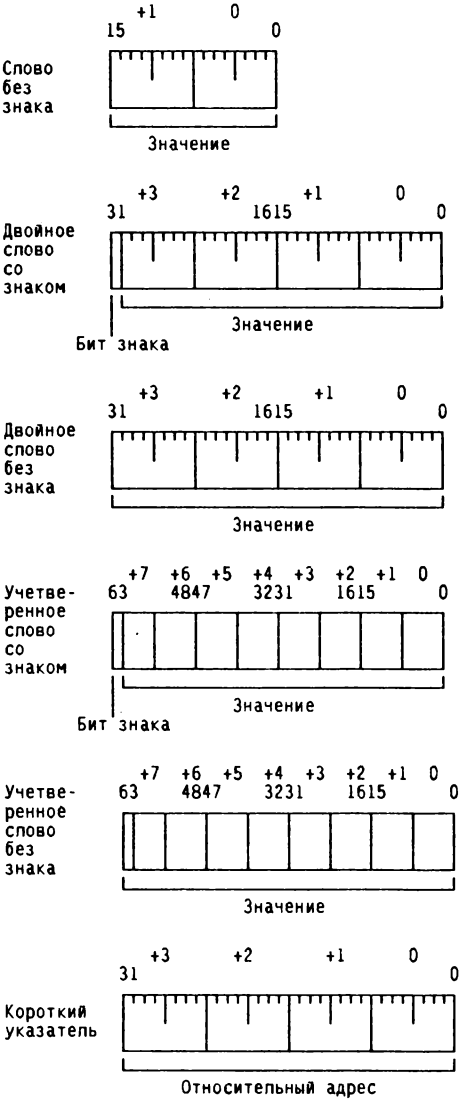
Двоично-десятичное число (BCD): распакованное представление десятичного числа от 0 до 9 в двоичном коде, занимающее младшую тетраду (четыре младших бита). Старшая тетрада может иметь любое значение при сложении и вычитании, но должна быть равна нулю при умножении и делении.

Упакованное двоично-десятичное число (BCD): упакованное представление десятичных чисел от 0 до 9, каждое из которых занимает свою тетраду. При этом один байт представляет десятичные числа от 0 до 99.

Число с плавающей точкой: если МП 80386 используется совместно с цифровым сопроцессором, то обеспечивается представление и обработка 32-, 64- или 80-разрядных чисел со знаком в формате с плавающей точкой.

Рис. 1.9 иллюстрирует типы данных, поддерживаемые МП 80386 и 80387. Поддержка этих типов данных обеспечивает эффективную реализацию программ на языках высокого уровня.




+1
0
15
0
Слово без знака
Значение
+3
+2
+1
0
31
1615
0
Двойное слово со знаком
Значение
Бит знака
+3
+2
+1
0
31
1615
0
Двойное слово без знака
Значение
+7
+6
+5
+4
+3
+2
+1
0
Учетверенное слово со знаком
63
4847
3231
1615
0
Значение
Бит знака
+7
+6
+5
+4
+3
+2
+1
0
Учетверенное слово без знака
63
4847
3231
1615
0
Значение
+3
+2
+1
0
31
0
Короткий указатель
Относительный адрес

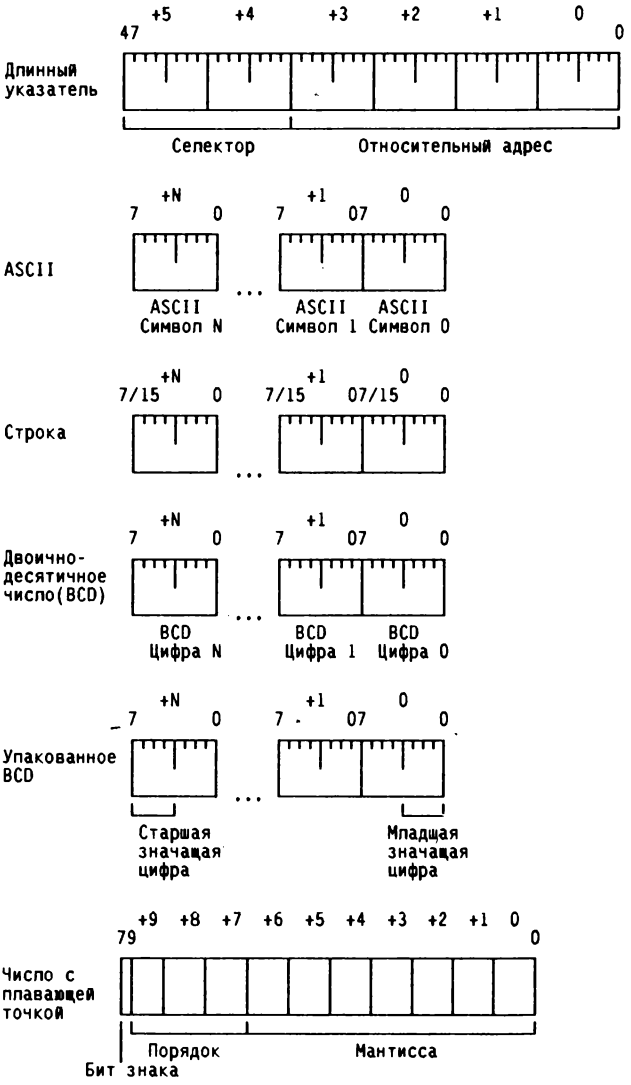


Рис.1.9. Типы данных, обрабатываемые МП 80386

Глава 2. Обмен по магистрали

Микросхема МП 80386 заключена в 132-выводный корпус с матричным расположением выводов. Изображение микросхемы со стороны выводов представлено на рис. 2.1, а список выводов по функциональным группам приведен в табл.2.1. В этой главе описывается назначение сигналов на выводах МП 80386 и рассматриваются магистральные циклы обмена. Символ # в конце названия сигнала указывает, что активным является низкий уровень сигнала. Если символ # отсутствует, то сигнал активен при высоком уровне.

2.1. Описание сигналов микропроцессора

Взаимодействие микропроцессора с другими устройствами системы осуществляется при помощи 32-разрядной двунаправленной шины данных, 34-разрядной шины адреса и шины управления, линии которой объединены в несколько функциональных групп (рис. 2.2).

2.1.1. Синхросигнал (CLK2)

Синхросигнал CLK2 обеспечивает тактирование работы МП 80386. Обычно для генерации CLK2 используется микросхема генератора синхросигналов типа 82384. Внутренняя (рабочая) частота МП 80386 в два раза меньше частоты сигнала CLK2. Рабочий такт включает два периода синхросигнала CLK2, которые называются фазами синхронизации - Ф1 и Ф2. Соотношение внешнего и внутреннего сигналов синхронизации иллюстрируется рис. 2.3.

2.1.2. Шина данных (D0-D31)

Двунаправленная шина данных с возможностью установки выходов в третье состояние обеспечивает передачу данных между МП 80386 и другими устройствами. Имеется возможность пересылки 32- и 16-разрядных данных; изменение разрядности шины осуществляется с использованием входа управления BS16#.

При операциях чтения необходимо следить за соблюдением правильных значений времени установки и удержания данных.

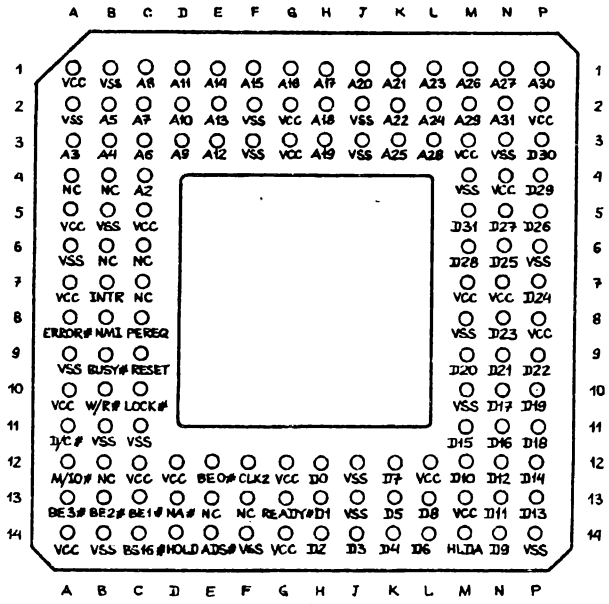


Рис.2.1. Микросхема МП 80386 (со стороны выводов)

Таблица 2.1. Список выводов по функциям

Сигнал	Вывод	Сигнал	Вывод	Сигнал	Вывод	Сигнал	Вывод
		A31	N2	ADS#	E13	Vcc	C12
		A30	P1	BS16#	C14	Vcc	D12
CLK2	F12	A29	M2	BUSY#	B9	Vcc	G2
D31	M5	A28	L3	D/C#	A11	Vcc	G3
D30	P3	A27	N1	ERROR#	A8	Vcc	G12
D29	P4	A26	M1	LOCK#	C10	Vcc	G14
D28	M6	A25	K3	M/IO#	A12	Vcc	L12
D27	N5	A24	L2	NA#	Q13	Vcc	M3
D26	P5	A23	L1	READY#	C13	Vcc	M7
D25	N6	A22	K2	W/R#	B10	Vcc	M13
D24	P7	A21	K1			Vcc	N4
D23	N8	A20	J1			Vcc	N7
D22	P9	A19	H3	HOLD	D14	Vcc	P2
D21	N9	A18	H2	HLDA	M14	Vcc	P8
D20	M9	A17	H1	INTR	B7	Vss	A2
D19	P10	A16	G1	NMI	B8	Vss	A6
D18	P11	A15	F1	PEREQ	C8	Vss	A9
D17	N10	A14	E1	RESET	C9	Vss	B1
D16	N11	A13	E2			Vss	B5
D15	M11	A12	E3			Vss	B11
D14	P12	A11	D1	Не подкл.	A4	Vss	B14
D13	P13	A10	D2	Не подкл.	B4	Vss	C11
D12	N12	A9	D3	Не подкл.	B6	Vss	F2
D11	N13	A8	C1	Не подкл.	B12	Vss	F3
D10	M12	A7	C2	Не подкл.	C6	Vss	F14
D9	N14	A6	C3	Не подкл.	C7	Vss	J2
D8	L13	A5	B2	Не подкл.	E13	Vss	J3
D7	K12	A4	B3	Не подкл.	F13	Vss	J12
D6	L14	A3	A3			Vss	J13
D5	K13	A2	C4	Vcc	A1	Vss	M4
D4	K14			Vcc	A5	Vss	M8
D3	J14	BE3#	A13	Vcc	A7	Vss	M10
D2	H14	BE2#	B13	Vcc	A10	Vss	N3
D1	H13	BE1#	C13	Vcc	A14	Vss	P6
D0	H12	BE0#	E12	Vcc	C5	Vss	P14

2.1.3. Шина адреса (BE0#-BE3#, A2-A31)

Шина адреса имеет выходы с возможностью установки в третье состояние, сигналы которых определяют физический адрес ячейки памяти или порта ввода/вывода. Шина позволяет адресовать пространство физической памяти объемом 4Гбайт (адреса 00000000H-FFFFFFFFH) и пространство ввода/вывода объемом 64Кбайт (адреса 00000000H-0000FFFFH). Сигналы адресации байтов BE0#-BE3# непосредственно показывают, какие байты 32-разрядной шины данных используются при текущей передаче:

BE0# адресует байт D0-D7
BE1# адресует байт D8-D15
BE2# адресует байт D16-D23
BE3# адресует байт D24-D31

Такая побайтная адресация наиболее удобна для обмена информацией с внешними устройствами. Количество адресуемых

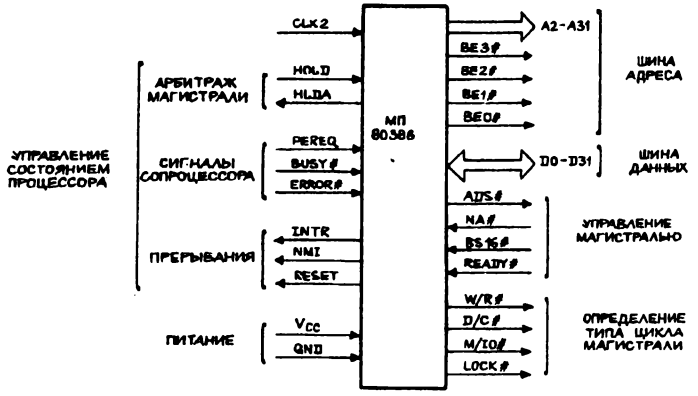


Рис.2.2. Функциональные группы сигналов магистрали

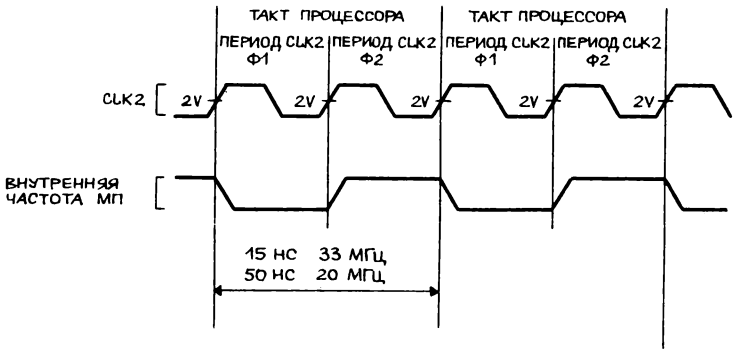


Рис.2.3. Синхросигнал CLK2

байтов определяет физический размер передаваемого операнда (1, 2, 3 или 4 байта). Если в цикле записи в память или устройство ввода/вывода пересылаемый операнд занимает только старшие 16 разрядов шины данных (D16-D31), для выравнивания нагрузки осуществляется дублирование данных в младшие 16 разрядов шины данных (D0-D15).

Соответствие формата данных и комбинаций сигналов на выводах BE0-BE3# при записи приведено в табл. 2.2.

Таблица 2.2. Дублирование данных при записи и сигналы BE0# - BE3#

Сигналы разрешения байтов				Байты данных				Автомат. дублирование
BE3#	BE2#	BE1#	BE0#	D24-D31	D16-D23	D8-D15	D0-D7	
1	1	1	0	неопред.	неопред.	неопред.	A	нет
1	1	0	1	неопред.	неопред.	B	неопред	нет
1	0	1	1	неопред.	C	неопред.	C	да
0	1	1	1	D	неопред.	D	неопред	да
1	1	0	0	неопред.	неопред.	B	A	нет
1	0	0	1	неопред.	C	B	неопред	нет
0	0	1	1	D	C	D	C	да
1	0	0	0	неопред.	C	B	A	нет
0	0	0	1	D	C	B	неопред	нет
0	0	0	0	D	C	B	A	нет

Ключи:
D - источник байта данных D24 - D31
C - источник байта данных D16 - D23
B - источник байта данных D8 - D15
A - источник байта данных D0 - D7

2.1.4. Сигналы определения цикла магистрали (W/R#, D/C#, M/IO#, LOCK#)

Эти выходы с третьим состоянием определяют тип выполняемого цикла шины. Сигнал W/R# разделяет циклы записи и циклы чтения, D/C# разделяет циклы данных и циклы управления, M/IO# разделяет циклы обращения к памяти и циклы ввода/вывода, LOCK# отличает блокированные и неблокированные циклы магистрали.

Основными являются сигналы W/R#, D/C# и M/IO#, поскольку они имеют силу с того момента, как становится

действующим сигнал ADS# (состояние адреса). Сигнал LOCK# становится действующим в начале первого блокированного цикла магистрали, а это из-за конвейеризации адресов происходит позже, чем устанавливается ADS# (разд. 2.4.2. "Чтение/запись с конвейером адресов").

Определение типа цикла магистрали, как функции W/R#, D/C# и M/IO# дано в табл. 2.3.

Таблица 2.3. Типы циклов магистрали

M/IO#	D/C#	W/R#	Тип цикла	Блокировка
0	0	0	ПОДТВЕРЖДЕНИЕ ПРЕРЫВАНИЯ	Да
0	0	1	Не имеет места	-
0	1	0	ЧТЕНИЕ ДАННЫХ ИЗ УСТРОЙСТВА ВВОДА/ВЫВОДА	Нет
0	1	1	ЗАПИСЬ ДАННЫХ В УСТРОЙСТВО ВВОДА/ВЫВОДА	Нет
1	0	0	ЧТЕНИЕ КОДА ИЗ ПАМЯТИ	Нет
1	0	1	ОСТАНОВ: Адрес = 2 (BE0# = 1 BE1# = 1 BE2# = 0 BE3# = 1 A2-A31 = 0) ОТКЛЮЧЕНИЕ: Адрес = 0 (BE0# = 1 BE1# = 1 BE2# = 1 BE3# = 1 A2-A31 = 0)	Нет
1	1	0	ЧТЕНИЕ ДАННЫХ ИЗ ПАМЯТИ	Некоторые циклы
1	1	1	ЗАПИСЬ ДАННЫХ В ПАМЯТЬ	Некоторые циклы

2.1.5. Сигналы управления магистралью

Сигналы управления магистралью МП 80386 определяют начало цикла магистрали, позволяют другим системным устройствам управлять конвейеризацией адресов, размером шины данных и завершением цикла магистрали.

Состояние адреса (ADS#). Этот выходной сигнал подтверждает код цикла магистрали и адрес (W/R#, D/C#, M/IO#, BE0#-BE3# и A2-A31) на выводах МП 80386. Он появляется в тактах T1 и T2P.

Подтверждение передачи (READY#). Этот входной сигнал показывает, что текущий цикл магистрали завершен и активные байты, определяемые BE0#-BE3# и BS16#, приняты или выставлены. Если READY# активен во время цикла чтения или цикла подтверждения прерывания, МП 80386 фиксирует входные данные и завершает цикл. Если READY# активен во время цикла записи, то процессор просто завершает магистральный цикл. Значение READY# игнорируется в первом такте всех циклов магистрали, а далее в каждом такте магистрали проверяется, пока не будет зафиксированно активное значение. Каждый цикл магистрали, включая циклы индикации останова и индикации выключения, должен быть завершен активным сигналом READY#.

Запрос следующего адреса (NA#). Применяется как запрос конвейеризации. Этот входной сигнал показывает готовность к приему новых значений BE0#-BE3#, A2-A31, W/R#, D/C# и M/IO# от МП 80386, даже если конец текущего цикла не подтверждается сигналом READY#.

Шестнадцатиразрядная шина (BS16#). Позволяет непосредственно связать МП 80386 с 32-разрядной и 16-разрядной шинами. При активности этого входного сигнала в текущем цикле шины используются только младшие разряды (D0-D15) шины данных, в соответствии с BE0#, BE1#. Он не оказывает влияния, если в текущем цикле активны только BE0# и/или BE1#. Если активны сигналы на выводах BE2# или BE3#, то активность BS16# приводит к дублированию байтов использованию только линий данных D0-D15. Если операнд занимает обе половины шины данных и активен BS16#, МП 80386 автоматически производит еще один цикл 16-разрядной шины. Циклы ввода/вывода для связи МП 80386 с сопроцессором формируются автоматически. МП 80386 и сопроцессор 80387 обмениваются 32-разрядными данными, поэтому во время циклов обмена с 80387 сигнал BS16# не должен быть активным.

2.1.6. Сигналы управления состоянием процессора

В этом разделе описываются входные сигналы МП 80386, которые изменяют его состояние в ходе выполнения или перед

выполнением программы. Это необходимо для реализации таких механизмов, как передача процессором управления по запросам от других активных устройств, обработка прерываний, сброс и инициализация.

Запрос захвата магистрали (HOLD). Входной сигнал запроса управления магистралью от активного внешнего устройства. Он должен оставаться активным все время, пока магистралью управляет другое устройство. Сигнал HOLD не воспринимается, пока активизирован сигнал RESET. Если эти сигналы активизированы одновременно, то RESET имеет приоритет.

Подтверждение захвата магистрали (HLDA). Активность этого выхода говорит о том, что МП 80386 передал управление магистралью в ответ на запрос HOLD и находится в состоянии подтверждения захвата. При этом единственным сигналом, вырабатываемым МП 80386 является HLDA, другие выходные и двунаправленные линии (D0-D31, BE0#-BE3#, A2-A31, W/ R#, D/C#, M/IO#, LOCK#, APS#) находятся в третьем состоянии. Запрос по входу NMI во время подтверждения захвата фиксируется для обработки после снятия сигнала HOLD.

Кроме обычного использования сигнала подтверждения захвата при работе с контроллерами прямого доступа к памяти и другими активными устройствами, перевод микропроцессора в третье состояние полезен во время тестирования системы, когда тестирующее оборудование осуществляет управление, а также при создании систем повышенной надежности.

Запрос сопроцессора (PEREQ). Активность этого сигнала указывает на запрос сопроцессора к МП 80386 по пересылке операнда в память или из памяти. В ответ микропроцессор пересылает информацию между сопроцессором и памятью. Поскольку у микропроцессора имеется код выполняемой команды сопроцессора, он производит запрошенную передачу данных с правильным направлением и адресом памяти.

Сопроцессор занят (BUSY#). Активность этого входного сигнала показывает, что сопроцессор все еще выполняет команду и не способен принять следующую. Когда МП 80386 сталкивается с любыми командами сопроцессора, которые оперируют с арифметическим стеком или командой WAIT, начинает автоматически проверяться вход BUSY#, и эта проверка выполняется до тех пор, пока сигнал на входе не станет пассивным. Это исключает наложение на выполнение

предыдущей команды сопроцессора. Команды сопроцессора FNINT и FNCLEX могут выполняться при активизированном BUSY#, поскольку применяются для инициализации сопроцессора и его сброса при исключительных ситуациях. BUSY# не должен быть синхронизирован с сигналом CLK2. Вывод BUSY# имеет альтернативную функцию - если BUSY# имеет низкий уровень в момент заднего фронта сигнала RESET, то МП 80386 производит самотестирование.

Ошибка сопроцессора (ERROR#). Сигнал указывает, что предшествующая команда сопроцессора привела к ошибке, не маскирумой регистром управления сопроцессора. Этот вход автоматически проверяется микропроцессором, когда встречается команда сопроцессора, и если сигнал активен, МП 80386 генерирует исключение 7 для передачи управления соответствующей процедуре обслуживания. Некоторые команды сопроцессора, в основном те, которые сбрасывают флаги арифметических ошибок в сопроцессоре или сохраняют состояние сопроцессора, выполняются без генерации исключения 7 при активном ERROR#. Такими командами являются: FNINIT, FNCLEX, FSTSW, FSTSWAX, FSTCW, FSTENV, FSAVE, FESTENV и FESAVE. Сигнал ERROR# идентифицируется по значению логического уровня и может не быть синхронным по отношению к сигналу CLK2.

Запрос маскируемого прерывания (INTR). Активный сигнал на этом входе является запросом на обслуживание прерывания, который может быть замаскирован битом IF регистра флагов МП 80386. Отвечая на сигнал INTR, МП 80386 производит два цикла подтверждения прерывания и в конце второго читает 8-разрядный вектор прерывания на выводах D0-D7 для идентификации источника прерывания. Сигнал INTR идентифицируется по уровню и может быть асинхронным к сигналу CLK2. Чтобы гарантировать распознавание запроса, INTR должен оставаться активным до начала первого магистрального цикл подтверждения прерывания.

Запрос немаскируемого прерывания (NMI). Активный сигнал на этом входе является запросом на обслуживание прерывания, который не может быть замаскирован программно. Обслуживание немаскируемого прерывания всегда производится процедурой, соответствующей входу 2 таблицы прерываний, поэтому циклы подтверждения прерывания не выполняются. Активность сигнала

NMI определяется нарастающим фронтом, он может быть асинхронным к сигналу CLK2. Для гарантии распознавания NMI должен иметь низкий уровень в течение по крайней мере 8 периодов CLK2, а затем высокий уровень в течение по крайней мере 8 периодов CLK2. Если обработка NMI началась, другие запросы NMI не обрабатываются до следующей команды IRET, которая завершает программу обслуживания. Однако, если сигнал NMI к тому времени вновь активизировался, то один запрос будет зафиксирован для обработки после выполнения следующей команды IRET.

Сброс (RESET). Этот входной сигнал приостанавливает все производимые действия и переводит МП 80386 в состояние инициализации. МП 80386 сбрасывается при активности сигнала RESET в течение 15 или более периодов CLK2 (80 или более периодов CLK2, до запроса самотестирования). Если сигнал на входе RESET активен, все остальные входные сигналы игнорируются и выводы переводятся в состояния, указанные в разд.7.1 "Сброс и инициализация". Если сигналы RESET и HOLD оказываются в какой-то момент времени активными одновременно, RESET имеет приоритет, даже если перед этим МП 80386 находился в состоянии подтверждения захвата магистрали. RESET идентифицируется по уровню и может быть асинхронным к сигналу CLK2.

2.2. Протоколы обмена по магистрали

Как отмечено выше, тип каждого цикла магистрали определяется комбинацией сигналов W/R#, D/C#, M/IO# и LOCK# (табл. 2.3). Одновременно с этими сигналами устанавливается физический адрес (A31-A2, BE3#-BE0#), а сигнал ADS# стробирует выдачу адреса и сигналов, определяющих тип цикла.

В активном состоянии интерфейс магистрали выполняет один из следующих циклов:

1) чтение из области памяти;
2) блокированное чтение из области памяти;
3) запись в область памяти;
4) блокированная запись в область памяти;
5) чтение из области ввода/вывода (или сопроцессора);

6) запись в область ввода/вывода (или сопроцессора);
7) подтверждение прерывания;
8) индикация останова или выключения.

Все типы циклов возможны как при 32- так и 16-разрядной шине данных. Если МП 80386 не производит ни одного из действий, перечисленных выше, он находится или в пассивном состоянии или в состоянии подтверждения захвата, что может быть определено внешней аппаратурой. Пассивное состояние идентифицируется в том случае, когда очередной цикл завершен, но МП 80386 больше не активизирует строб состояния ADS#. Состояние подтверждения захвата идентифицируется по активности выходного сигнала HLDA.

Элементарным интервалом времени при реализации протоколов обмена является такт магистрали, продолжительность которого равна одному периоду внутренней частоты процессора (или двум периодам CLK2). Полная передача данных происходит в течение цикла магистрали, состоящего из двух или более тактов. Каждый магистральный цикл продолжается до получения от внешней аппаратуры сигнала READY#. Если подтверждение приходит в конце первого такта T2, это приводит к самому короткому циклу магистрали, состоящему только из тактов T1 и T2. Если READY# не активен, то состояние T2 повторяется до тех пор, пока проверка входа READY# не укажет на его активность. На рис. 2.4 показаны три последовательных цикла чтения, каждый из них включает два такта магистрали, T1 и T2. С помощью такого цикла может быть осуществлено обращение по любому адресу памяти или ввода/вывода, если внешние устройства обладают достаточным быстродействием. При полной ширине шины данных двутактный цикл реализует весь потенциал быстрой основной памяти или кэш-памяти.

2.3. Конвейерное формирование адресов

МП 80386 может формировать два типа адресов: обычные и *конвейерные* (pipelined address). В зависимости от выбранного типа реализуются различные временные соотношения между сигналами, управляющими выполнением циклов магистрали. Наличие или отсутствие конвейера адресов определяется значением входного сигнала NA#.

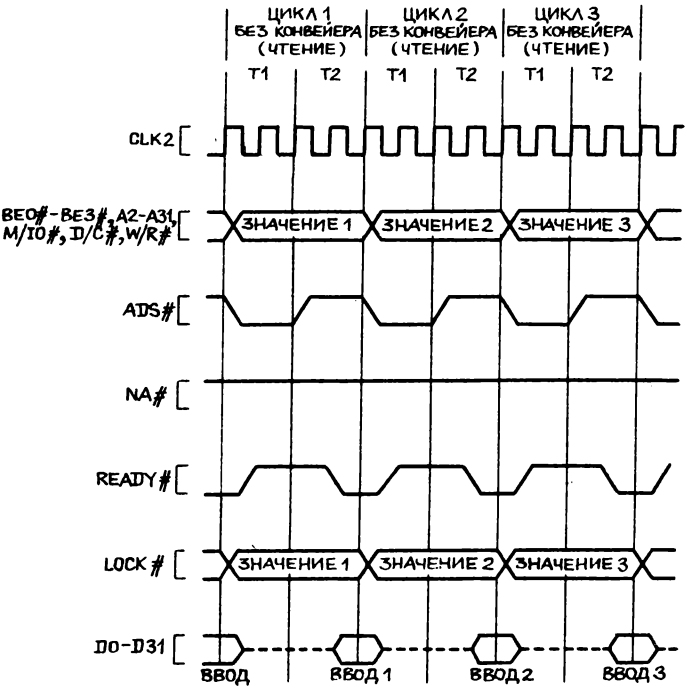


Рис 2.4. Самые короткие циклы без конвейера

При отсутствии конвейера адресов текущий адрес и код цикла остаются постоянными в течение всего цикла. Когда реализуется конвейер адресов, адрес и код следующего цикла устанавливаются до окончания текущего цикла, о чем свидетельствует активное состояние выхода ADS#. Рис. 2.5 демонстрирует самые быстрые циклы чтения с конвейером адресов. Эти циклы используют всего два такта магистрали, T1P и T2P (символ P в конце номера такта указывает на конвейер адресов в этом такте). Видно, что циклы с конвейером адресов предоставляют большее время для выборки данных, чем неконвейерные. Благодаря этому уменьшается потребность в тактах ожидания. Например, если при неконвейерном

формировании адресов требуется один такт ожидания, то при конвейерном их вообще не требуется. Конвейеризация полезна в системах, где используются регистры-защелки адреса. В таких системах конвейерное формирование следующего адреса позволяет декодирующим схемам с опережением генерировать сигналы выбора кристалла и другие необходимые сигналы управления. В результате выбранные устройства становятся доступными сразу же после начала следующего цикла. Таким образом, декодирование для следующего цикла магистрали может осуществляться в конце текущего цикла, т.е. обеспечивается частичное перекрытие циклов. Если система имеет память, состоящую из нескольких чередующихся банков, то конвейерное формирование адресов позволяет достичь еще большего перекрытия.

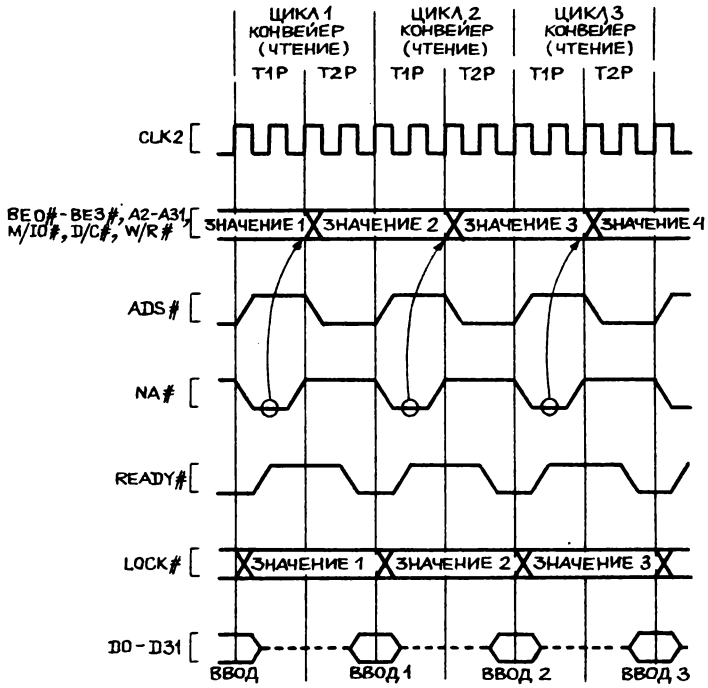


Рис 2.5. Самые короткие циклы с конвейером

2.4. Циклы чтения и записи

Магистральные циклы чтения и записи осуществляют передачу данных. Направление передачи определяется относительно процессора, т.е. при чтении данные вводятся в МП 80386, а при записи выводятся из него.

При выполнении рассматриваемых циклов возможен динамический выбор способа формирования адресов - неконвейерный или конвейерный. После пассивного состояния микропроцессор всегда использует неконвейерное формирование адреса, для конвейеризации адресов в следующем цикле должен быть активизирован сигнал на входе NA#. Этот вход проверяется в каждом цикле магистрали.

При функционировании МП 80386 возможен динамический выбор одного из двух вариантов шины данных - 32 разряда или 16 разрядов. Пассивный уровень сигнала BS16# указывает на 32-разрядную шину, активный уровень BS16# задает 16-разрядный размер шины. Вход BS16# проверяется незадолго до конца цикла для определения разрядности шины данных, используемой в текущем цикле. Если задан 16-разрядный размер шины, то МП 80386 автоматически выполняет все действия для полной передачи по 16-разрядной шине. В зависимости от размера и выравнивания операнда МП 80386 производит дополнительный цикл 16-разрядной шины, используя выводы D0-D15 для передачи старших байтов D16-D31 (табл. 2.2).

Завершение циклов чтения и записи, как и любого другого цикла магистрали, требует подтверждения в виде активного сигнала READY#. Пока подтверждение не получено, процессор вставляет такты ожидания в цикл магистрали, чтобы обеспечить согласование по скорости с внешним устройством. Значение READY# проверяется в конце второго такта цикла. Если внешнее устройство подтверждает цикл установкой активного значения READY#, то цикл завершается, как показано на рис.2.6. Если сигнал READY# не активизирован (рис. 2.7), то цикл дополняется еще одним тактом T2 (такт ожидания), в конце которого READY# вновь проверяется. Этот процесс продолжается, пока не будет зафиксировано активное значением сигнала READY#, подтверждающее завершение цикла.

Когда подтверждается цикл чтения, МП 80386 защелкивает информацию, находящуюся на линиях данных. Когда

подтверждается цикл записи, выдаваемые процессором данные сохраняются на шине данных в течение следующего такта, чтобы обеспечить время, необходимое для их записи в память или внешнее устройство.

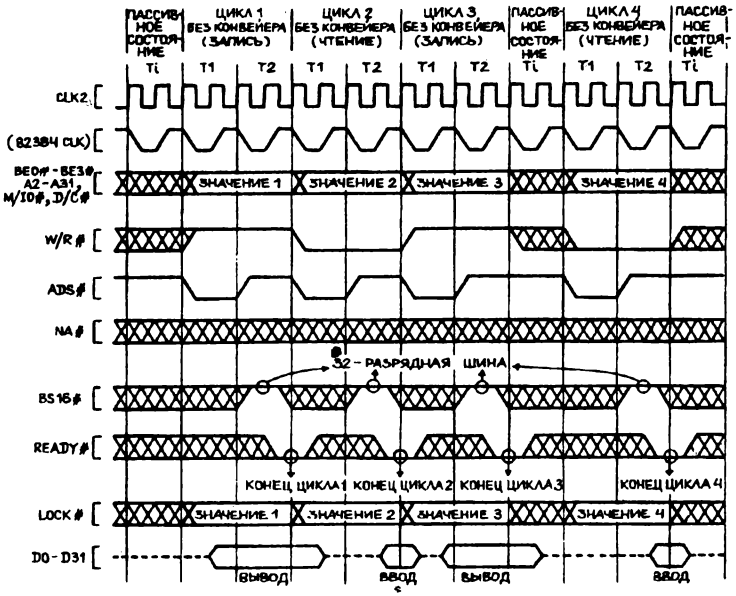


Рис.2.6. Циклы без тактов ожидания

2.4.1. Чтение/запись без конвейера

На рис. 2.8 показана диаграмма состояний магистрали и переходы между состояниями для случая, когда конвейерная организация адресов не используется. Переходы осуществляются между четырьмя состояниями шины - T1, T2, Ti и Th. Цикл состоит из тактов T1 и T2, причем для состояний ожидания такт T2 может повторяться. Кроме того, магистраль может находиться в пассивном состоянии Ti или в состоянии подтверждения захвата Th.

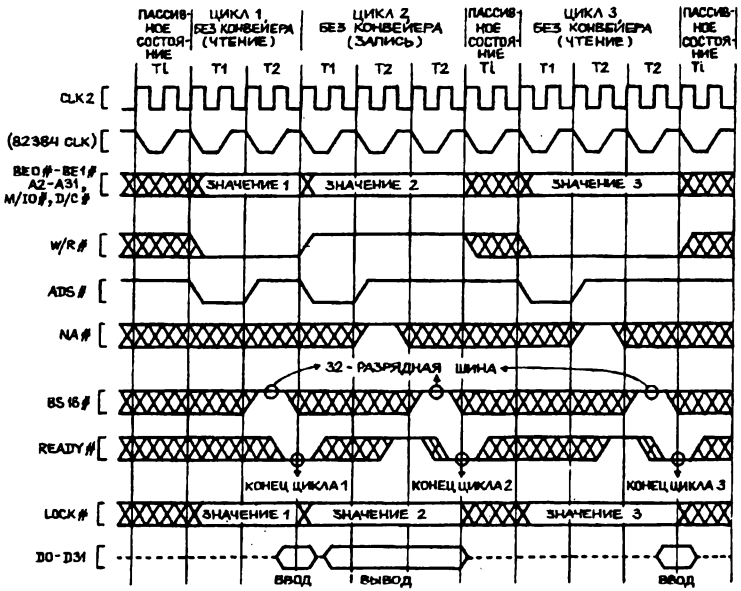


Рис.2.7. Циклы с тактами ожидания

Магистральные циклы всегда начинаются с такта Ti, после такта T1 всегда следует такт T2. Если в течении такта T2 не поступает подтверждение цикла и сигнал NA# пассивен, такт T2 повторяется. Если цикл завершается сигналом READY# в такте T2, то далее может следовать: такт T1 следующего цикла, если имеется внутренний запрос к интерфейсу магистрали МП 80386; Ti, если внутреннего запроса нет; Th, если активен входной сигнал HOLD. Эта диаграмма состояний учитывает и изменение разрядности шины данных. Если поступил активный сигнал BS16# и для полной передачи по 16-разрядной шине требуется дополнительный цикл, то он осуществляется в соответствии с диаграммой состояний на рис.2.8.

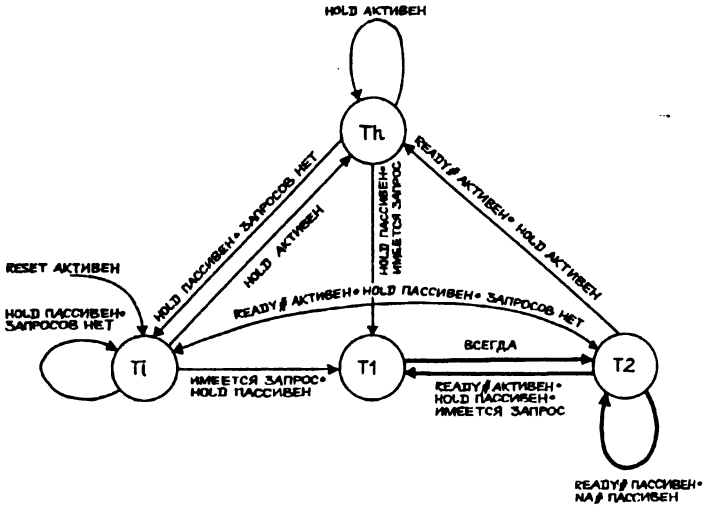


Рис.2.8. Диаграмма состояний без конвейера адресов

Во время циклов чтения или записи, выполняемых в соответствии с диаграммой на рис.2.8, управление шиной данных осуществляется следующим образом. Если выполняется цикл чтения, МП 80386 высвобождает шину, передавая управление внешнему адресованному устройству. Если выполняется цикл записи, управление шиной данных осуществляется процессором, начиная со второй половины такта T1 и до окончания первой фазы такта, следующего за сигналом подтверждения цикла. На рис. 2.6 показаны неконвейерные циклы без тактов ожидания, а на рис. 2.7 к циклам 2 и 3 добавлены по одному такту ожидания. В этих циклах в конце первого такта T2 при проверке значение READY# оказалось пассивным, поэтому такт T2 повторен. В конце второго такта T2 сигнал READY# оказался активным.

Когда конвейерное формирование адресов не применяется, адрес и код цикла сохраняют в течение всех тактов ожидания.

При этом необходимо обеспечить пассивное значение сигнала NA# во время всех тактов T2, за исключением последнего, как показано на рис. 2.7 (циклы 2 и 3). Если во время какого-либо такта, кроме последнего такта T2, сигнал NA# будет иметь активное значение, то следующим будет такт T2I или T2P для конвейерных адресов.

Размер шины данных для неконвейерных циклов записи и чтения может составлять 32 или 16 разрядов. В начале цикла шина данных 32-разрядная. Когда цикл подтверждается активным сигналом READY# в конце такта T2, результат проверки значения BS16# определяет разрядность шины данных для текущего цикла. Если BS16# пассивен, размер шины данных определяется как 32-разрядный; при активном BS16# устанавливается 16-разрядное значение. Во втором случае для полной передачи требуется два цикла 16-разрядной шины и BS16# должен сохранять это же зна-чение во время второго цикла, т.е. 16-разрядный размер шины не предполагается, а должен быть определенно указан. Второй 16-разрядный цикл должен быть подтвержден сигналом READY#. Значения адресных разрядов A31-A2 во втором цикле остаются теми же, что и в первом, а сигналы BE1# и BE0# всегда пассивны (высокий уровень).

Рисунки 2.9 и 2.10 иллюстрируют случаи, когда для полной передачи 32-разрядного операнда требуются два цикла 16-разрядной шины. На рис. 2.9 показаны циклы без тактов ожидания, а на рис. 2.10 циклы с одним тактом ожидания. На рис. 2.10 в циклах 1 и 1A сигнал NA# должен быть пассивным в тактах T2 перед последним. Это необходимо для правильности результатов проверки значения BS16# в последнем такте T2.

2.4.2. Чтение/запись с конвейером адресов

Конвейерное формирование адресов - это установка адреса и определение типа следующего цикла до того, как текущий цикл подтвержден активным значением сигнала READY#.

Когда готов следующий адрес, процессор активизирует сигнал на выводе ADS#. Опция конвейерной организации адресов управляется входным сигналом NA# от цикла к циклу.

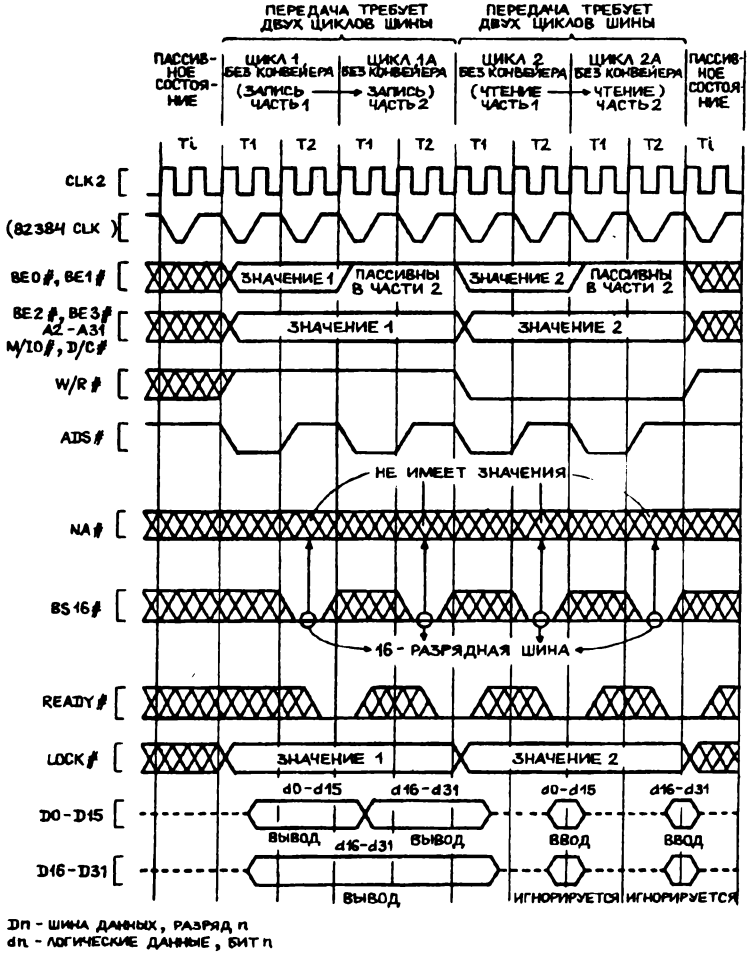


Рис.2.9. Обмен по 16-разрядной шине (без тактов ожидания)

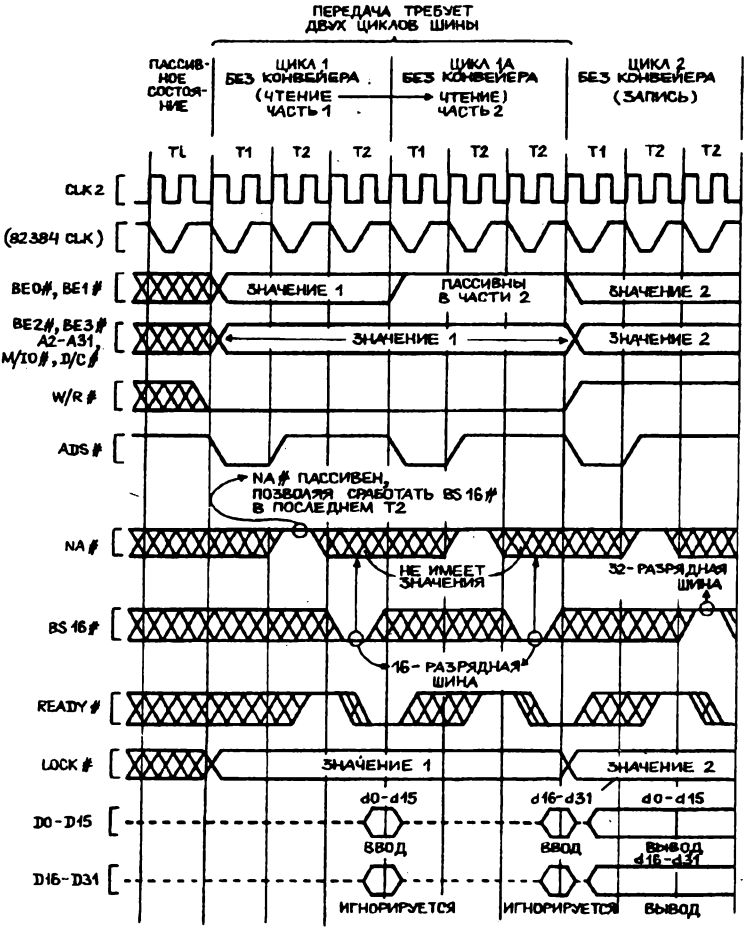


Рис.2.10. Обмен по 16-разрядной шине (с тактами ожидания)

Если адрес в течение по крайней мере одного такта магистрали был истинным, вход NA# проверяется в конце каждой фазы Ф1 до подтверждения цикла. Следовательно, во время неконвейерных циклов вход NA# проверяется в конце первой фазы каждого такта T2. Примером служит цикл 2 на рис. 2.11, в котором сигнал NA# имел активное значение во время первого такта T2.

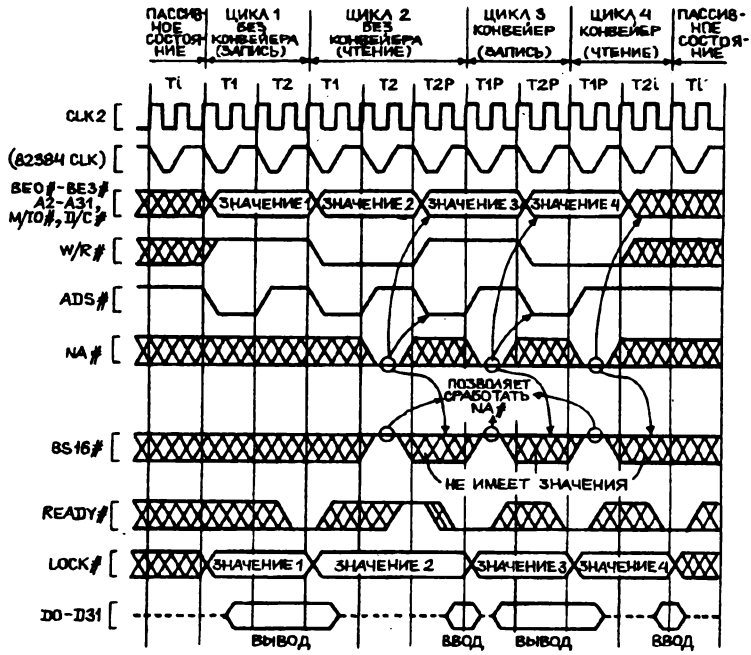


Рис.2.11. Переход к конвейеру адресов

МП 80386 имеет следующие особенности конвейера адресов:

1. При анализе состояния NA#, сигнал BS16# должен быть пассивным в окне проверки (рис. 2.11 циклы 2 - 4, рис. 2.12 циклы 1 - 4). Если при проверке во время последнего периода T2 цикла магистрали активны и NA# и BS16#, то активность BS16# имеет приоритет. В этом случае текущий размер шины

устанавливается 16-разрядным и следующий адрес является неконвейерным.

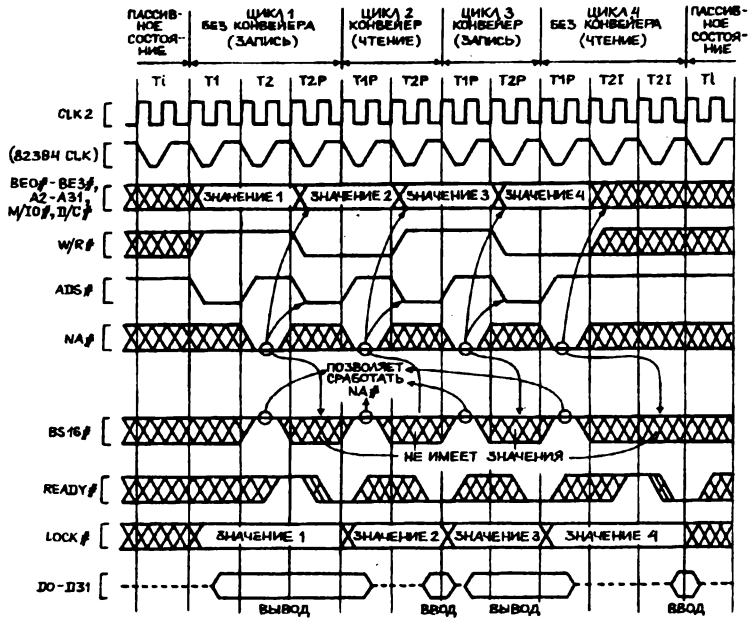


Рис 2.12. Самый короткий переход к конвейеризации адресов

2. Следующий адрес может появиться сразу после состояния магистрали, в котором при проверке NA# оказался активным (рис. 2.11, 2.12). В этом случае сразу вводится состояние T2P. Однако, если у интерфейса магистрали МП 80386 нет внутреннего запроса, следующий адрес не будет доступен сразу после того, как NA# окажется активным, и вместо T2P вводится T2I (рис. 2.13, цикл 3). Если текущий цикл еще не подтвержден установкой READY#, такт T2P появится сразу, как только микропроцессор выдаст следующий адрес. Внешняя аппаратура должна анализировать выход ADS#, чтобы подтвердить появление следующего адреса на шине.

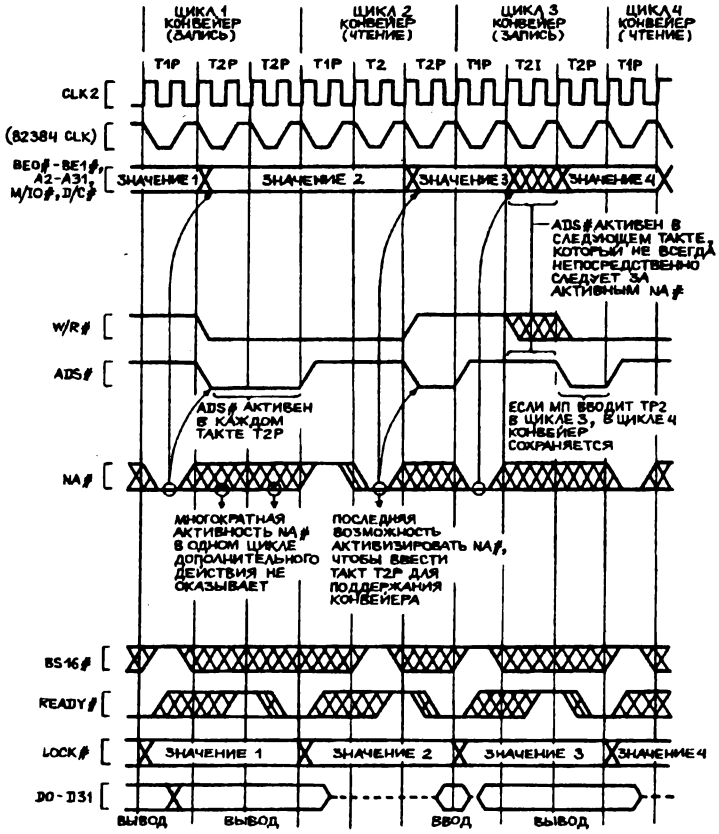


Рис. 2.13. Конвейер адресов с тактами ожидания

3. После выявления активного сигнала NA# МП 80386 переходит к внутреннему запросу шины, имеющему самый высокий приоритет. Он не может произвести еще один 16-разрядный цикл по тому же адресу, если BS16# установлен извне, поэтому с этого момента подразумевается 32-разрядный размер шины. Таким образом, если NA# при проверке внутри цикла окажется установленным, с этого момента в этом шинном цикле сигнал BS16# игнорируется (рис. 2.11, 2.12, 2,13). Поэтому, не устанавливайте в активное состояние NA# во время циклов, в которых уменьшается размер шины данных.

4. Любой адрес, правильность которого подтверждена сигналом на выходе ADS#, будет оставаться стабильным в течение, по крайней мере, двух периодов синхронизации микропроцессора (рис. 2.11, 2.12, 2.13). МП 80386 не может формировать новый адрес чаще, чем каждые два периода тактовой частоты процессора.

5. Тип цикла и адрес определяются только для следующего цикла, механизм конвейеризации работает только на один цикл вперед.

Полная диаграмма состояний магистрали и переходов из одного состояния в другое приведена на рис. 2.14. Видно, что она представляет собой диаграмму для неконвейерных адресов, к которой добавлены три состояния, характеризующих конвейер адресов.

Самый быстрый цикл магистрали с конвейерным адресом состоит из двух состояний шины T1P и T2P (напомним, что для неконвейерного адреса это T1 и T2). T1P всегда является первым тактом конвейерного цикла.

Рассмотрим переходы из пассивного состояния Ti к началу конвейерного цикла T1P (рис. 2.14). Из состояния Ti возможен переход только в T1, а это неконвейерный цикл. Однако следующий цикл может быть конвейерным, если NA# активен и первый цикл заканчивается состоянием T2P (адрес для следующего цикла приходит во время T2P). Самый быстрый путь от пассивного состояния к циклу с конвейерным адресом следующий:

Ti, Ti, Ti	T1 - T2 - T2P	T1P - T2P
пассивное	неконвейерный	конвейерный
состояние	цикл	цикл

T1-T2-T2P являются тактами цикла, который устанавливает конвейер адресов для следующего цикла, начинающегося с T1P. Тоже самое верно после состояния захвата:

Th, Th, Th	T1 - T2 - T2P	T1P - T2P
состояние подтверждения захвата	неконвейерный цикл	конвейерный цикл

На рис. 2.12 цикл 1 используется для перехода к конвейерному формированию адресов в циклах 2, 3, 4. В соответствующее время активизируется вход NA#.

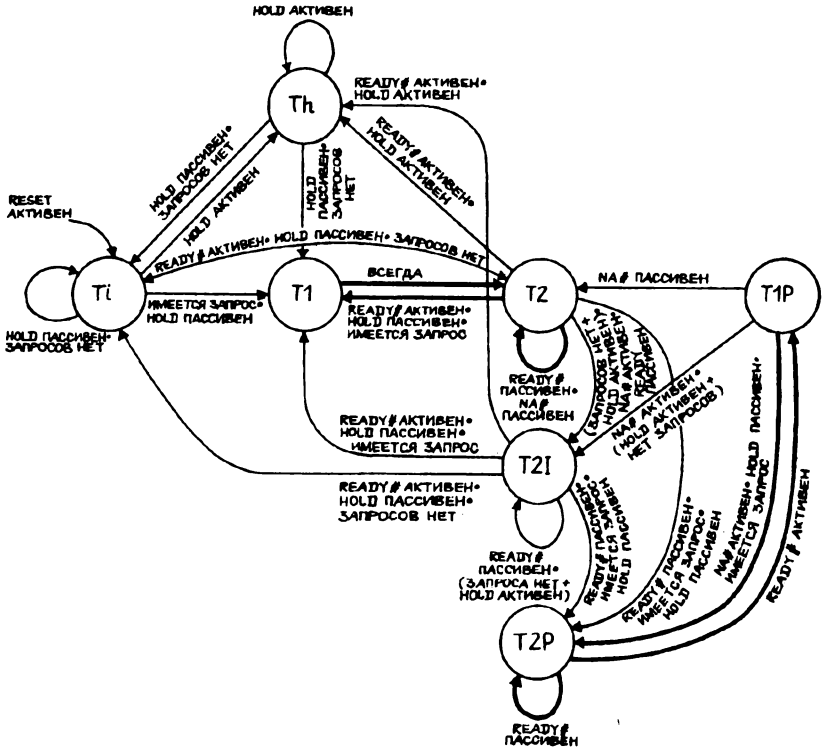


Рис 2.14. Полная диаграмма состояний МП 80386

При выполнении цикла магистрали, если текущий адрес подтвержден, вход NA# проверяется в конце фазы Ф1 начала такта до момента подтверждения цикла. Поэтому во время цикла 1 на рис. 2.12 проверка начинается в такте Т2. Если NA# при проверке в текущем цикле активен, МП 80386 может выводить на шину адрес и тип нового цикла, начиная со следующего состояния шины. Например, в цикле 1 на рис. 2.12 следующий адрес выводится во время состояния Т2Р. Таким образом, цикл 1 производит переход к конвейеризации, поскольку начинается с Т1, а заканчивается Т2Р. Поскольку адрес для цикла 2 становится доступным до начала самого цикла, цикл 2 называется конвейерным и начинается с Т1Р. Цикл 2 начинается в тот момент, когда установка READY# завершает цикл 1.

Примерами переходных циклов являются цикл 2 на рис.2.11 и цикл 1 на рис.2.12. Рисунок 2.12 показывает переход во время первого цикла после пассивного состояния шины, это самый быстрый переход к конвейеру адресов. Цикл 2 на рис. 2.11 иллюстрирует переход, происходящий в течение последовательности циклов без конвейера. В любом случае цикл перехода всегда одинаков: он состоит по меньшей мере из Т1, Т2 (в этот момент устанавливаеся NA#) и Т2Р (при условии, что у МП 80386 имеется внутренний ждущий запрос шины, а он имеется почти всегда). Если к циклу добавляются состояния ожидания, то повторяются такты Т2Р. Видно, что три состояния (Т1, Т2 и Т2Р) требуются только для циклов, осуществляющих переход от неконвейерных адресов к конвейерным, например, цикл 1 на рис.2.12. Циклы 2, 3, 4 на рис. 2.12 показывают, что поддержка конвейера адресов может быть выполнена циклами, содержащими только 2 такта - Т1Р и Т2Р.

Если выполняется конвейерный цикл, то конвейерзация для следующего цикла поддерживается установкой активного значения NA# и выявлением того, что во время текущего цикла МП 80386 вводит такт типа Т2Р. Последнее идентифицируется по установке активного значения ADS#. Рисунки 2.11, 2.12 демонстрируют конец конвейера после цикла 4, т.к. этот цикл заканчивается тактом Т2I. Это говорит о том, что до поступления подтверждения цикла 4, МП 80386 не имел внутреннего запроса шины. Если цикл заканчивается тактом Т2 или Т2I, то следующий цикл не будет конвейерным.

Практически конвейерная организация адресов поддерживается в течение длинных последовательностей циклов, если шина доступна и в каждом цикле сигнал NA# активен. Это происходит потому, что пока очередь не заполнена, в отсутствие других запросов имеется запрос выборки кода из памяти.

Если при проверке сигнал на входе NA# оказался активным, МП 80386 начинает реагировать на внутренний ждущий запрос и выводит соответствующий адрес на шину. Таким образом, установка NA# делает невозможным доступ для следующего шинного цикла по текущему адресу на A2-A31, который может потребоваться, если внешним устройством активизирован сигнал BS16#.

Во избежание конфликта при одновременной активизации сигналов NA# и BS16# предусмотрено следующее:

1. МП 80386 игнорирует BS16# в текущем цикле, если ранее в этом цикле при проверке обнаружен активный NA#. Следовательно,при активном NA# шина данных подразумевается 32-разрядной.

2. Если BS16# и NA# активны в одном окне проверки, BS16# имеет приоритет и МП 80386 ведет себя так, как если бы NA# в этот момент был пассивным.

Определенные типы операндов не требуют согласования для правильной передачи по 16-разрядной шине. Таковыми являются операнды чтения или записи, использующие только младшую половину шины данных и операнды записи, использующие только старшую половину шины, так как МП 80386 дублирует данные записи на младшей половине шины данных. Для таких сочетаний сигналов разрешения байтов и R/W# нет необходимости в установке BS16# и активность NA# допускается в течении всего цикла.

2.5. Цикл подтверждения прерывания (INTA)

В ответ на запрос прерывания по входу INTR (когда прерывания разрешены) МП 80386 производит два цикла подтверждения прерывания (рис. 2.15). Эти циклы схожи с циклами чтения.

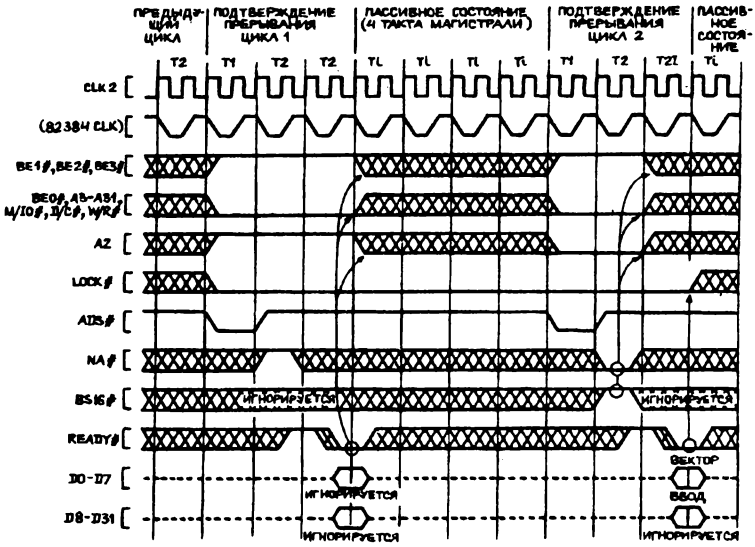


Рис. 2.15. Цикл подтверждения прерывания

Первый и второй циклы подтверждения прерывания различаются по состоянию A2. Адрес байта, адресуемого во время первого цикла подтверждения прерывания, равен 4 (на A31-A3 логический "0", A2=1, на BE3#-BE1# логическая "1", BE0#=0). Адрес байта, адресуемого во время второго цикла подтверждения прерывания, равен 0 (A31-A2 логический "0", на BE3#-BE1# логическая "1", BE0=0).

Сигнал на выход LOCK# активен с начала первого цикла подтверждения прерывания и до конца второго цикла. Между двумя циклами подтверждения прерывания МП 80386 вставляет четыре такта Ti, что необходимо для сопряжения с контроллером прерываний 8259A.

Данные в конце первого цикла не читаются, в конце второго цикла МП 80386 вводит вектор прерывания с линий D0-

D7 шины данных. Этот вектор указывает номер прерывания (0-255), которое необходимо обработать.

2.6. Циклы индикации останова и выключения

МП 80386 производит останов в результате выполнения команды HALT, а аварийное выключение - в порядке защиты от двойной ошибки. Чтобы указать на эти состояния, выполняются циклы индикации останова и выключения (рис. 2.16, 2.17). Они идентифицируется по комбинации сигналов, указанных в табл. 2.3, и отличаются значениями сигналов на выводах BE0#, BE2#. Во время циклов индикации останова и выключения линии D0-D31 пассивны, циклы должны быть подтверждены сигналом READY#.

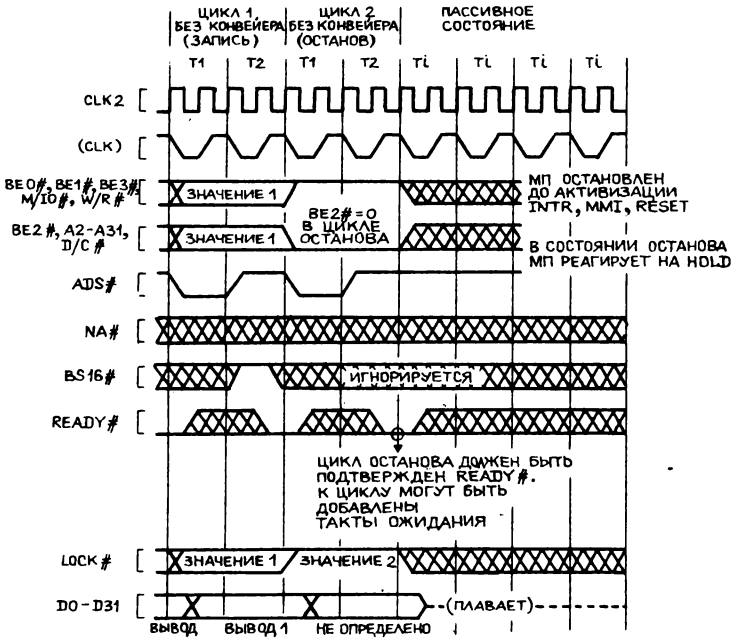


Рис. 2.16. Цикл индикации останова

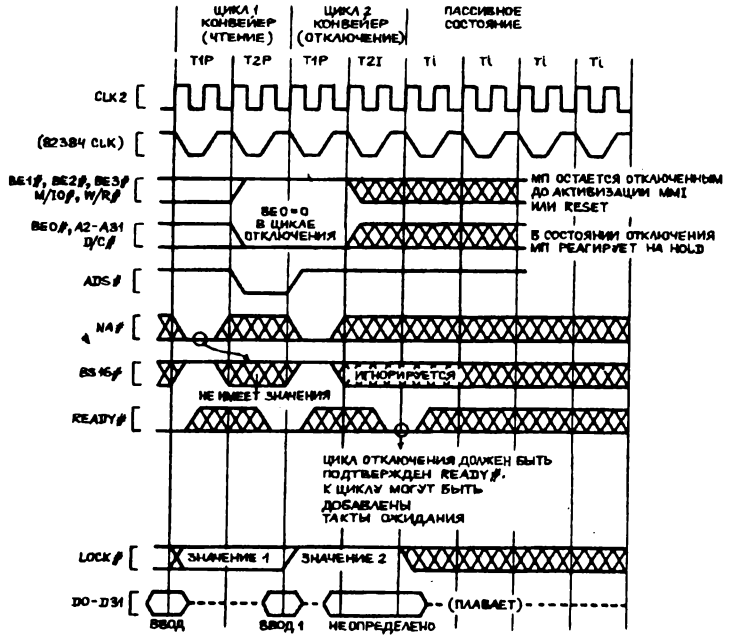


Рис. 2.17. Цикл индикации выключения

После останова МП 80386 возобновляет действия по актививности сигналов на входах INTR (если прерывания разрешены), NMI или RESET. Аварийно выключенный микропроцессор возобнавляет действия по активным сигналам NMI или RESET.

Глава 3. Формат команд и способы адресации

Набор команд МП 80386 обеспечивает выполнение операций над операндами, которые находятся в регистре, памяти или непосредственно в команде. В набор входят безадресные, одно-, двух- и трехадресные команды. Микропроцессор 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. Форматы и коды всех команд МП 80386 даны в гл. 4.

3.1. Общий формат команд

В обобщенном виде формат команды МП 80386 показан на рис. 3.1. Код команды содержит следующие поля: код операции (1 или 2 байта); байты адресации (0, 1 или 2 байта); байты смещения (0, 1, 2 или 4 байта); байты непосредственных данных-операндов (0, 1, 2 или 4 байта).

КОП	MOD R/M	SIB	СМЕЩЕНИЕ	ОПЕРАНД
1 или 2 байта	0 или 1 байта	0 или 1 байт	0,1,2 или 4 байта	0,1,2 или 4 байта

Рис.3.1. Общий формат команд

Команды содержат от 1 до 11 байт. Проведенные оценки показывают, что в среднем длина команды составляет 4-5 байт, поэтому очередь команд в МП 80386 (16 байт) обычно содержит около 3-4 команды. Перед кодом команды в ряде случаев вводятся один или несколько префиксных байтов, модифицирующих выполняемую операцию.

Рассмотрим назначение основных полей кода команды (рис.3.1). Код операции (КОП) занимает 1 или 2 байта. Во многих командах пересылок, а также в логических и арифметических командах значение бита w в первом байте КОП 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 микропроцессора и устанавливается битом D в дескрипторе сегмента кода. При выполнении отдельных команд разрядность операндов может меняться соответствующим префиксом. В ряде команд первой байт КОП содержит поля reg или sreg, определяющие выбор используемых регистров. Трехбитовое поле reg задает выбираемый регистр в соответствии с разрядностью обрабатываемых операндов (табл. 3.1).

Поле reg определяет выбор сегментных регистров и может быть двух или трехбитовым (табл. 3.2). При этом трехбитовый код sreg позволяет выбирать дополнительные сегментные регистры FS и GS МП 80386. Коды sreg=110, 111 не используются.

Выполнение ряда пересылок, арифметических и логических операций зависит от значения битов d, s в первом байте КОП:

d- определяет выбор источника и приемника операндов для двухоперандных команд;

s- осуществляет расширение знака для 8 разрядных непосредственных данных.

Влияние битов d, s на выполнение команд поясняется ниже.

Таблица 3.1. Кодировка регистров общего назначения

Поле reg	Разрядность операндов		
	8	16	32
000	AL	AX	EAX
001	CL	CX	ECX
010	DL	DX	EDX
011	BL	BX	EBX
100	AH	SP	ESP
101	CH	BP	EBP
111	BH	DI	EDI

Таблица 3.2. Кодировка сегментных регистров

sreg 2 бита	sreg 3 бита	Сегментный регистр
00	000	ES
01	001	CS
10	010	SS
11	011	DS
	100	FS
	101	GS

Байт адресации MODR/M содержит три поля (рис. 3.2.). 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 (разд. 3.2.).

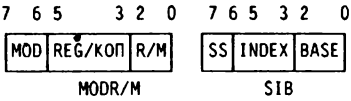


Рис. 3.2. Форматы байтов MODR/M и SIB

В одноадресных командах поле REG/КОП 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 КОП. При этом бит d в КОП задает выбор регистров, используемых в качестве источника и приемника информации при выполнении ряда двухадресных арифметических и логических операциий типа регистр-регистр:

d=0 - код источника в поле REG/КОП, код приемника в поле R/M;

d=1 - код источника в поле R/M, код приемника в поле REG/КОП.

Кодировка регистров указана в табл. 3.1, 3.2.

Для реализации некоторых 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. 3.2). Правила формирования адреса при использовании байта SIB изложены в разд. 3.2.

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.3.4, 3.5). Это смещение задается соответствующими байтами в коде команды, которые располагаются после байтов адресации.

При выполнении операций с непосредственной адресацией один из операндов задается в последних байтах команды (рис.3.1). В этом случае КОП ряда команд содержит бит s, определяющий способ использования непосредсвенно задаваемых данных. Если операция выполняется над байтами (КОП команды содержит бит w=0), то в качестве операнда задается один байт

непосредственных данных im8. Если операция выполняется над 16- или 32-разрядными словами (в КОП команды 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. Такой способ формирования операнда называется расширением знака.

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, 2 или 3 больше определяемых байтами адресации.

3.2. Способы адресации

Микропроцессор 80386 реализует 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 микропроцессора 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. 3.3.). Для некоторых способов обращения возможны варианты выбора сегментных регистров, которые указаны в табл. 3.3. 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(DI) или эффективный адрес (EA), который формируется в соответствии с заданным способом адресации.

Таблица 3.3. 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(SP)
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(DI)
6. Адресация операнда с использованием в качестве базового регистра EBP(BP) или ESP (SP)	SS(CS,DS,ES,FS,GS)	EA

* В скобках указаны регистры, содержащие относительный адрес при работе МП 80386 в режиме обработки 16 разрядных операндов.

Эффективный адрес операнда (EA) 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 EA образуется путем арифметического сложения трех компонент:

- содержимого базового регистра EBP(BP) или EBX(BX);
- 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 команды (рис. 3.1).

В зависимости от значения полей MOD и R/M для формирования EA используются все или часть этих слагаемых в соответствии с табл. 3.4, 3.5. В этих таблицах указаны также сегментные регистры (DS: и SS:), используемые для определения базовых адресов сегмента.

Если при формировании 32-разрядного адреса R/M=100, то команда содержит дополнительный байт адресации SIB и правила образования EA определяются табл. 3.6, где (IR*F) - масштабированный индекс; IR - содержимое индексного регистра,

определяемого кодом INDEX (табл. 3.7); F - масштабный коэффициент, определяемый кодом SS (табл. 3.8).

Таблица 3.4. Формирование 16- разрядного EA

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 3.5. Формирование 32- разрядного EA (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.3.6.)	(см.табл.3.6.)	(см.табл.3.6.)
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]

Таблица 3.6. Формирование 32- разрядного EA (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]

Следует напомнить, что установленная для выполняемой задачи разрядность относительного адреса операнда может быть

изменена для текущей команды с помощью соответствующего префикса (табл. 1.1).

Таблица 3.7. Кодировка индексных и базовых регистров в байте SIB

Код INDEX	Тип индексного регистра	Код BASE	Тип базового регистра
000	EAX	000	EAX
001	ECX	001	ECX
010	EDX	010	EDX
011	EBX	011	EBX
100	Нет индексного регистра	100	ESP
101	EBP	101	EBP(d32)
110	ESI	110	ESI
111	EDI	111	EDI

Таблица 3.8. Кодировка множителя F

ss	Масштабный множитель F
00	1
01	2
10	0
11	8

Микропроцессор реализует ряд способов адресации операнда, набор которых обеспечивает эффективную работу с такими языками высокого уровня, как Си, Фортран и др.

Непосредственная адресация. В качестве операнда 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 КОП, содержащегося в первом байте тих команд, или поля REG/КОП байта MODR/M. Разрядность используемых непосредственных данных (8, 16 или 32 разряда) зависит от режима работы микропроцессора и может изменяться соответствующим префиксом.

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

Косвенно-регистровая адресация. Относительный адрес содержится в индексном (SI, DI, ESI, EDI) или базовом (BX, EBX) регистрах, или регистрах общего назначения EAX, ECX, EDX. Данный способ реализуется при MOD=00 (табл. 3.4, 3.5).

Прямая адресация. Относительный адрес операнда 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. 3.4) или R/M=101 (табл. 3.5).

Базовая адресация. Относительный адрес формируется путем сложения содержимого базового регистра (BX, BP) и смещения d8 или d16. Базовая адресация реализуется при MOD=01 или 10 и значениях R/M=110 или 111 (табл.3.4).

Индексная адресация. Относительный адрес 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. 3.4).

При формировании 32-разрядных адресов в качестве базового или индексного, может использоваться любой из регистров EAX, ECX, EDX, EBX, EBP, ESI, EDI, (табл. 3.5).

Базово-индексная адресация. Относительный адрес образуется путем сложения содержимых базового (BX, BP) и сегментного регистров (SI, DI). Такой способ адресации операнда реализуется при значениях поля R/M=000, 001, 010, 011, если задано MOD=00 (табл. 3.4).

Базово-индексная адресация со смещением. Это вариант базово-индексной адресации, при котором к относительному адресу прибавляется смещение d8 или d16. Такая адресация осуществляется при значениях MOD=01 или 10 (табл. 3.4).

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 используется в качестве индекса и базы для формирования адреса операнда (табл. 3.7).

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 EBP 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. 3.7). Содержимое этого регистра умножается на масштабный коэффициент F (табл.3.8), т.е. 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.

Индексная адресация с масштабированием. Относительный адрес образуется сложением масштабированного индекса (содержимого индексного регистра) 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: EAX, EBX, ECX, EDX, ESI или EDI. Выбор регистра определяется значением поля BASE при MOD=00 (табл. 3.7).

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, а поле MJD=01 или 10 определяет разрядность смещения (табл. 3.7).

При описанных способах задается относительный адрес операнда в сегменте данных DS или стека SS (табл. 3.6). Для адресации операндов в других сегментах перед командой необходимо ввести префиксный байт SEG.

Относительная адресация используется в микропроцессоре 80386 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 IP, соответствующего текущей команде, и смещения d8, d16 или d32, определяющего положение следующей команды относительно текущей.

Глава 4. Система команд

Микропроцессор 80386 реализует набор из 146 команд, причем большинство из них аналогичны командам микропроцессора 8086 (К1810ВМ86). Основное различие в системах команд этих микропроцессоров связано с расширением разрядности (до 32) и введением команд, обеспечивающих обработку битовой информации, защиту данных, хранящихся в памяти, и поддержку реализации языков высокого уровня.

4.1. Общий перечень команд

Набор команд МП 80386 делится на девять групп в зависимости от типа выполняемых операций. В данном разделе дается общий перечень команд, разбитых на функциональные группы.

1. Пересылка данных и адресов

Пересылка данных без преобразования:

MOV	Пересылка операнда
PUSH	Запись операнда в стек
PUSHA	Запись в стек содержимого всех регистров
POP	Чтение операнда из стека
POPA	Чтение из стека содержимого всех регистров
XCHG	Обмен между регистрами или памятью и регистром
XLAT	Преобразование кодов

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 регистр
OUT	Вывод операнда из регистра в порт

Загрузка адреса селекторов

LEA	Загрузка адреса EA в регистр
LDS	Загрузка селектора в регистр DS

LES	Загрузка селектора в регистр ES
LFS	Загрузка селектора в регистр FS
LGS	Загрузка селектора в регистр GS
LSS	Загрузка селектора в регистр SS
2. Арифметические операции	
Сложение:	
ADD	Сложение операндов
ADC	Сложение операндов с признаком CF (перенос)
INC	Инкремент операнда
AAA	ASCII коррекция результата сложения
DAA	Десятичная коррекция результата сложения
Вычитание:	
SUB	Вычитание операндов
SBB	Вычитание операндов и признака CF (заем)
DEC	Декремент операнда
CMP	Сравнение операндов
NEG	Изменение знака операнда (с переводом в дополнительный код)
AAS	ASCII коррекция результата вычитания
DAS	Десятичная коррекция результата вычитания
Умножение:	
MUL	Беззнаковое умножение
IMUL	Знаковое (целочисленное) умножение
AAM	ASCII коррекция результата умножения
Деление:	
DIV	Беззнаковое деление
IDIV	Знаковое (целочисленное) деление
AAD	ASCII коррекция результата деления
Преобразование	байтов, слов, двойных слов:
CBW	Преобразование байта (AL) в слово (AX)
CWDE	Преобразование слова (AX) в двойное слово (EAX)
CWD	Преобразование слова (AX) в двойное слово (DX, AX)

CDQ	Преобразование двойного слова (EAX) в учетверенное слово (EDX, EAX)

3. Логические операции и сдвиги

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)

4. Операции с битами и строками битов

BT	Проверка бита
BTS	Проверка и установка бита
BTR	Проверка и сброс бита
BTC	Проверка и инверсия бита
BSF	Прямое сканирование битов
BSR	Обратное сканирование битов

5. 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

6. Управление программой

Безусловная передача управления:	
JMP	Безусловный переход
CALL	Вызов подпрограммы
RET	Возврат из подпрограммы
Условные переходы:	
JA/JNBE	Переход, если выше (не ниже или равно)
JAE/JNB/JNC	Переход, если (CF)=0: выше или равно (не ниже)
JB/JNAE/JC	Переход, если (CF)=1: ниже (не выше или равно)
JBE/JNA	Переход, если ниже или равно (не выше)
JE/JZ	Переход, если равно (нуль): (ZF)=1
JNE/JNZ	Переход, если не равно (не нуль): (ZF)=0
JP/JPE	Переход, если четность : (PF)=1
JNP/JPO	Переход, если нечетность : (PF)=0
Условные переходы	с учетом знака:
JG/JNLE	Переход, если больше (не меньше или равно)
JGE/JNL	Переход, если больше или равно (не меньше)
JL/JNGE	Переход, если меньше (не больше или равно)
JLE/JNG	Переход, если меньше или равно (не больше)
JS	Переход, если отрицательно : (SF)=1
JNS	Переход, если положительно : (SF)=0
JO	Переход, если переполнение : (OF)=1
JNO	Переход, если нет переполнения: (OF)=0
Условная установка	байта:
SETA/SETNBE	Установка, если выше (не ниже или равно)
SETAE/SETNB	Установка, если выше или равно (не ниже)
SETB/SETNAE	Установка, если ниже (не выше или равно)

SETBE/SETNA	Установка, если ниже или равно (не выше)
SETE/SETZ	Установка, если равно (нуль): (ZF)=1
SETNE/SETNZ	Установка, если не равно (не нуль): (ZF)=0
SETP/SETPE	Установка, если четность :(PF)=1
SETNP/SETPO	Установка, если нечетность :(PF)=0
SETG/SETNLE	Установка, если не равно (не нуль)
SETGE/SETNL	Установка, если больше или равно (не меньше)
SETL/SETNGE	Установка, если меньше (не больше или равно)
SETLE/SETNG	Установка, если меньше или равно (не больше)
SETS	Установка, если отрицательно: (SF)=1
SETNS	Установка, если положительно: (SF)=0
SETO	Установка, если переполнение: (OF)=1
SETNO	Установка, если нет переполнения: (OF)=0
Прерывания:	
INT	Прерывание
INT0	Прерывание по переполнению: (OF)=1
INT3	Прерывание в контрольной точке
IRET	Возврат из подпрограммы прерывания
CLI	Запрещение прерываний
STI	Разрешение прерываний
Организация циклов:	
LOOP	Реализация циклов, пока (CX)=0
LOOPE/LOOPZ	Реализация циклов, пока (CX)=0 или (ZF)=1
LOOPNE/LOOPNZ	Реализация циклов, пока (CX)=0 или (ZF)=0
JCXZ (JECXZ)	Реализация циклов, пока (CX)=0 или (ECX)=0
Операции над признаками:	
LAHF	Загрузка признаков в регистр AH
SAHF	Запись содержимого AH в регистр признаков
PUSHF	Запись содержимого регистра признаков в стек

POPF	Чтение содержимого регистра признаков из стека
CLC	Сброс признака переноса : (CF)=0
STC	Установка признака переноса : (CF)=1
CMC	Инвертирование признака переноса
CLD	Сброс признака направления : (DF)=0
CTD	Установка признака направления: (DF)=1

7. Поддержка языка высокого уровня

BOUND	Проверка границ массива
ENTER	Обращение к процедуре
LEAVE	Выход из процедуры

8. Организация защиты памяти

SGDT	Запись содержимого регистра таблицы глобальных дескрипторов
SLDT	Запись содержимого регистра таблицы локальных дескрипторов
SIDT	Запись содержимого регистра таблицы дескрипторов прерываний
STR	Запись содержимого регистра задачи
LGD	Загрузка регистра таблицы глобальных дескрипторов
LLDT	Загрузка регистра таблицы локальных дескрипторов
LIDT	Загрузка регистра таблицы дескрипторов прерываний
LTR	Загрузка регистра задачи
CLTS	Сброс признака переключения задачи : (TS)=0
ARPL	Коррекция запрошенного уровня привилегий
LAR	Загрузка прав доступа
LSL	Загрузка границы сегмента
VERR	Проверка доступности сегмента при чтении
VERW	Проверка доступности сегмента при записи
LMSW	Загрузка слова состояния машины (MSW)
SMSW	Запись слова состояния машины (MSW)

9. Управление процессором

ESC	Подключение сопроцессора
MOV	Пересылка содержимого управляющих, отладочных, тестирующих регистров
WAIT	Ожидание до поступления сигнала BUSY=1
NOP	Отсутствие операции
HLT	Останов

Выполнение команд модифицируется при введении перед ними префиксных байтов:

LOCK	Блокировка магистрали
SEG	Замена сегмента

Префикс изменения разрядности адреса

Префикс изменения разрядности операнда

REP/REPE/REPZ	Повторение операций со строками
REPNE/REPNZ	Повторение операций со строками

В последующих разделах главы указаны форматы и коды всех команд, а также число тактов, требуемых для их выполнения в режимах реальных адресов, виртуального МП 8086 (РР, V86) и в защищенном режиме (ЗР). Приняты следующие обозначения:

R - регистр, определяемый полем reg в КОП или байте MODR/M;

M - ячейка памяти, адресуемая байтами MODR/M, SIB;

SR - сегментный регистр, определяемый полем sreg в КОП;

A - аккумулятор (AL, AX или EAX);

S - стек.

Число тактов определяется для случая, когда команда предварительно выбрана из памяти, декодирована и подготовлена для выполнения блоками предвыборки и преддешифрации. Предполагается также отсутствие состояний ожидания, запросов, блокирующих доступ к магистрали и других ситуаций, задерживающих выполнение команды. Если в качестве операнда в команде можно использовать как содержимое регистра, так и содержимое адресуемой ячейки памяти (R/M), то меньшее число тактов соответствует адресации к регистру.

4.2. Команды пересылки

Форматы и коды команд этой группы приведены в табл. 4.1. Значения признаков при выполнении команд не изменяются. В данной и последующих таблицах описания команд (табл. 4.1-4.14) в первой половине колонки "Число тактов" приводится информация о времени выполнения команды в режимах реальных адресов и виртуального МП 8086, а во второй половине колонки - в защищенном режиме.

Таблица 4.1. Команды пересылок

Команда	Формат	Число тактов	
MOV - пересылка;			
(R/M)<- (R)	1000100W mod reg r/m	2/2	2/2
(R)<- (R/M)	1000101w mod reg r/m	2/4	2/4
(R/M)<- (im)	1100011w mod 000 r/m	2/2	2/2
(R)<- (im)	1011w reg im8,16,32	2	2
(A)<- (M)	1010000w d8,16,32	4	4
(M)<- (A)	1010001w d8,16,32	2	2
(SR)<- (R/M)	10001110 mod sreg r/m	2/5	18/19
(R/M)<- (SR)	10001100 mod sreg r/m	2/2	2/2
PUSH-запись в стек			
S<- (M)	11111111 mod 110 r/m	5	5
S<- (R)	01010 reg	2	2
S<-(ES,CS,SS,DS)	000sreg110	2	2
S<- (FS или GS)	00001111 10 sreg 000	2	2
S<- (im)	011010s0 im8,16,32	2	2
PUSHA-запись в стек всех РОН	01100000	18	18
POP-чтение из стека			
(M)<- S	10001111 mod 000 r/m	5	5
(E)<- S	01011 reg	4	4
(ES,CS,SS,DS)<- S	000sreg111	7	21
(FS или GS)<- S	00001111 10 sreg 001	7	21
POPA- чтение из стека всех РОН	01100001	24	24
XCHG - обмен			
R/M<-> (R)	1000011w mod reg r/m	3/5	3/5
(A)<-> (R)	10010 reg	3	3
XLAT-преобразование кодов	11010111	5	5
MOVSX- пересылка с расширением знака	00001111 1011111w modregr/m	3/6	3/6
MOVZX-пересылка с расширением нулям	00001111 1011011w modregr/m	3/6	3/6
IN - ввод из			
Фиксированного порта	1110010w p8	5	5
Изменяемого порта	1110110w	6	6
OUT - вывод из			
Фиксированного порта	1110011w p8	3	3
	1110011w p8	3	3
Изменяемого порта	1110111w	4	4

Таблица 4.1. (продолжение)

Команда	Формат	Число тактов	
LEA- загрузка адреса EA в регистр	10001101 mod reg r/m	2	2
LDS-загрузка селектора в регистр DS	11000101 mod reg r/m	7	22
LES-загрузка селектора в регистр ES	11000100 mod reg r/m	7	22
LFS-загрузка селектора в регистр FS	00001111 10110100 modregr/m	7	25
LGS - загрузка селектора в в регистр GS	00001111 10110101 modregr/m	7	25
LSS - загрузка селектора в регистр SS	00001111 10110010 modregr/m	7	22
		7	22

Команда MOV имеет восемь модификаций, шесть из которых осуществляют пересылку данных, а две - пересылку содержимого сегментных регистров. При этом в байте MODR/M используется двухбитное поле sreg, адресующее сегментные регистры в соответствии с табл. 3.2. Необходимо отметить, что данная команда может оперировать только с регистрами CS, DS, SS или ES.

Адресация памяти при обращении к стеку производится с помощью регистра ESP (или SP), который содержит относительный адрес последней заполненной ячейки памяти в сегменте стека SS. При выполнении команд PUSH, PUSHA операнды, начиная с младшего байта, заносятся в ячейки сегмента стека с относительными адресами (SP)-1, (SP)-2 и т.д. При выполнении команд POP, POPA операнды, начиная со старших байтов, выбираются из ячеек сегмента стека с относительными адресами (SP), (SP)+1 и т.д. Команды PUSHA, POPA выполняют запись/чтение всех восьми регистров общего назначения. При этом в зависимости от заданной разрядности операндов производится загрузка/выборка содержимого младших 8, 16 или всех 32 разрядов регистров.

При операциях со стеком выбор сегментных регистров CS, DS, SS или ES производится с помощью двухбитового поля sreg в первом байте КОП, выбор дополнительных регистров FS, GS с

помощью трехбитового поля sreg во втором байте КОП (табл. 3.2).

Две модификации команды XCHG обеспечивают обмен содержимым между регистром, заданным полем reg, и регистром/памятью, адресуемым полями mod, r/m или аккумулятором.

Команда XLAT заменяет содержимое младшего байта аккумулятора AL на байт из 256-байтовой таблицы, начальный (базовый) адрес которой содержится в регистре EBX. При этом содержимое AL используется как относительный адрес (индекс) выбираемого в таблице байта Данная команда используется для табличного преобразования одного кода в другой.

Команды MOVSX, MOVZX осуществляют пересылку (R) <- (R/M) с одновременным расширением разрядности операнда. При w=0 производится преобразование байта в слово (режимы РР, V86) или двойное слово (ЗР). При w=1 слово преобразуется в двойное слово (режим ЗР). При выполнении команды MOVSX расширение производится заполнением старших разрядов значением знака, при выполнении команды MOVZX - заполнением нулями.

При выполнении команд ввода-вывода IN, OUT производится пересылка данных между аккумулятором и адресуемым портом. Адрес порта может задаваться вторым байтом команды p8 (фиксированный порт) или содержимым регистра DX (изменяемый порт). Таким образом возможно обращение к 256 портам, адреса которых фиксированы в программе, или к 65536 портам, адреса которых могут изменяться путем перезагрузки содержимого регистра DX. Отметим, что в защищенном режиме эти команды выполняются только, если уровень привилегии текущей программы CPL $\leqslant$ IOPL, где значение IOPL задается соответствующим полем содержимого регистра EFLAGS. При нарушении этого условия происходит прерывание типа 13 (нарушение защиты). Особенности выполнения этих и ряда других команд в защищенном режиме рассматриваются в разд. 5.2.

При выполнении команд LEA, LDS, LES, LFS, LGS, LSS один из операндов размещается в памяти, в байте MODR/M

задается код поля MOD=11. Команда LEA производит вычисление эффективного адреса EA этого операнда и загружает его в регистр, определяемый полем reg. Команды LDS, LES, LFS, LGS, LSS выбирают 16- или 32-разрядное слово из памяти и заносят его в регистр, определяемый полем reg. Затем следующее слово из памяти загружается в соответствующий сегментный регистр: DS, ES, FS, GS или SS. Таким образом производится перезагрузка селекторов, хранящихся в этих регистрах.

4.3. Команды арифметических операций

Форматы и коды данной группы команд даны в табл. 4.2. При выполнении команд производится установка кода признаков в соответствии с табл. 4.3.

Таблица 4.2. Команды арифметических операций

Команда	Формат	Число тактов	
ADD -сложение			
(R)-> (R)+(R)	000000dw mod reg r/m	2	2
(M)<- (R)+(M)	0000000w mod reg r/m	7	7
(R)<- (R)+(M)	0000001w mod reg r/m	6	6
(R/M)<-(R/M)+(im)	100000sw mod 000r/mim8,16,32	2/7	2/7
(A)<- (A)+(im)	0000010w im 8,16.32	2	2
(A)<- (A)+(im)	0000010w im 8,16.32	2	2
ADC -сложение с переносом (CF)			
(R)<- (R)+(R)+(CF)	000100dw mod reg r/m	2	2
(M)<- (R)+(M)+(CF)	0001000w mod reg r/m	7	7
(M)<- (R)+(M)+(CF)	0001001w mod reg r/m	6	6
(R/M)<-(R/M)+(im)+(CF)	100000sw mod010 r/mim8,16,32	2/7	2/7
(A)<-(A)+(im)+(CF)	0001010w im 8,16,32	2	2
INC -инкремент			
(R/M)<- (R/M)+1	1111111w mod 000 r/m	2/6	2/6
(R)<-(R)+1	01000reg	2	2
AAA-ASCIIкоррекция сложения	00111111	4	4
	00111111	4	4
DAA -десятичная коррекция сложения	00100111	4	4
	00100111	4	4

Таблица 4.2 (продолжение)

Команда	Формат	Число тактов	
SUB -вычитание			
(R)<- (R)-(R)	001010dw mod reg r/m	2	2
(M)<- (M)-(R)	0010100w mod reg r/m	7	7
(R)<- (R)-(M)	0010101w mod reg r/m	6	6
(R/M)<- (R/M)-(im)	100000sw mod 101r/mim8,16,32	2/7	2/7
(A)<- (A)-(im)	0010110w im8,16, 32	2	2
SBB -вычитание с заемом (CF)			
(R)<- (R)-(R)-(CF)	000110dw mod reg r/m	2	2
(M)<- (M)-(R)-(CF)	0001100w mod reg r/m	7	7
(R)<- (R)-(M)-(CF)	0001101w mod reg r/m	6	6
(R/M)<- (R/M)-(im)-(CF)	100000sw mod011r/m im8,16,32	2/7	2/7
(A)<-(A)-(im)-(CF)	0001110w im8,16,32	2	2
DEC -декремент			
(R/M<- (R/M)-1	1111111w reg001 r/m	2/6	2/6
(R)<- (R)-1	01001reg	2	2
CMP -сравнение			
(R)-(R)	001110dw mod reg r/m	2	2
(M)-(R)	0011100w mod reg r/m	5	5
(R)-(M)	0011101w mod reg r/m	6	6
(R/M)-(im)	100000sw mod111r/m im8,16,32	2/5	2/5
(A)-(im)	0011110w im8,16,32	2	2
NEG-изменение знака	1111011w mod 011 r/m	2/6	2/6
AAS-ASCII коррекция вычитания	00111111	4	4
DAS -десятичная коррекция	00101111	4	4
MUL-умножение беззнаковое			
(A)<-(A)*(R/M),	1111011w mod 100 r/m		
- 8-разрядн.		9-14/ 12-17	9-14/ 12-17
- 16-разрядн.		9-22/ 12-25	9-22/ 12-25
- 32-разрядн.		9-38/ 12-41	9-38/ 12-41
IMUL-умножение целое знаковое			
(A)<- (A)*(R/M)	1111011w mod 101 r/m		
- 8-разрядн.		9-14/ 12-17	9-14/ 12-17
- 16-разрядн.		9-22/ 12-25	9-22/ 12-25
- 32-разрядн.		9-38/ 12-41	9-38/ 12-41

Таблица 4.2 (продолжение)

Команда	Формат	Число тактов	
(R)<-(R)*(R/M), -8-разрядн.	00001111 10101111 modregr/m	9-14/ 12-17	9-14/ 12-17
- 16-разрядн.		9-22/ 12-25	9-22/ 12-25
- 32-разрядн.		9-38/ 12-41	9-38/ 12-41
(R)<-(R/M)*im - 8-разрядн.	011010s1 modregr/mim8,16,32	9-14/ 12-17	9-14/ 12-17
- 16-разрядн.		9-22/ 12-25	9-22/ 12-25
- 32-разрядн.		9-38/ 12-41	9-38/ 12-41
AAM -ASCII коррекция	11010100 00001010	17	17
DIV -деление беззнаковое			
(A)<- (A)/(R/M)	1111011w mod 110 r/m		
- 8-разрядн.		14/17	14/17
- 16-разрядн.		22/25	22/25
- 32-разрядн.		38/41	38/41
IDIV-деление целое (знаковое)			
(A)<- (A)/(R/M)	1111011w mod 111 r/m		
- 8-разрядн.		19/22	19/22
- 16-разрядн.		27/30	27/30
- 32-разрядн.		43/46	43/46
AAD -ASCII коррекция	11010101 00001010	19	19
CBW/CWDE - преобразование из байта в слово/слово в двойное слово	10011000	3	3
CWD/CDQ- удвоение слова/двойного слова	10011001	2	2

Команды сложения, вычитания (ADD, SUB) и сложения, вычитания с учетом переноса/заема (CF) имеют по пять модификаций с различными способами адресации операндов. При выполнении команды сравнения CMP содержимое адресуемых регистров и ячеек памяти не изменяется, производится только установка признаков. При равенстве операндов устанавливается значение (ZF)=1. При сравнении операндов без знака уменьшаемое *выше* вычитаемого, если устанавливается (CF)=0, уменьшаемое *ниже* вычитаемого, если (CF)=1. При сравнении

операндов со знаком уменьшаемое *больше* вычитаемого, если (SF)=(OF)=0 или (SF)=(OF)=1; *меньше*, если (SF)=0, (OF)=1 или (SF)=1, (OF)=0.

Таблица 4.3. Установка признаков при выполнении команд арифметических, логических операций и сдвигов

Команды	Признаки					
	OF	CF	AF	SF	ZF	PF
ADD,ADC,SUB,SBB,CMP,NEG	+	+	+	+	+	+
INC, DEC	+	-	+	+	+	+
MUL, IMUL	+	+	н	н	н	н
DIV, IDIV	н	н	н	н	н	н
DAA, AAS	н	+	+	+	+	+
AAA, AAS	н	+	+	н	н	н
AAM, AAD	н	н	н	+	+	+
AND, OR, XOR, TEST	0	0	н	+	+	+
SHL, SHR, (один разряд)	+	+	н	+	+	+
SHL,SHLD,SHR,SHRD(перем.разряд)	н	+	н	+	+	+
SAR	0	+	н	+	+	+
ROL,ROR,RCL,RCR (один разряд)	+	+	-	-	-	-
ROL,ROR,RCL,RCR (перем. разряд)	н	+	-	-	-	-

\+ установка признаков по результату операции
\- сохранение ранее установленного признака
н неопределенное значение признака
0 установка нулевого значения признака

Команда NEG преобразует адресуемый операнд в дополнительный код, меняя таким образом его знак.

Для умножения 8-, 16- или 32-разрядных операндов используются одноадресная команда MUL и команда IMUL, имеющая одно-, двух- или трехадресную форму. В одноадресных командах MUL и IMUL один из сомножителей и произведение размещаются в аккумуляторе: AL, AX или EAX в соответствии с разрядностью операндов. Так как разрядность произведения может вдвое превышать разрядность операндов, то для его размещения производится расширение аккумулятора AL -> AX, AX -> DX, AX, EAX -> EDX, EAX. При этом 16-разрядное произведение 8-разрядных операндов заносится в AL (младшие разряды), AH (старшие разряды), 32-разрядное произведение - в AX (младшие разряды), DX (старшие разряды), 64-разрядное произведение - в EAX (младшие разряды), EDX (старшие разряды). При двух- и трехадресной форме команды IMUL произведение размещается в адресуемом регистре или ячейке памяти. В этом случае старшие 16 или 32 разряда произведения при умножении 16- или 32-разрядных операндов теряются.

При выполнении команд MUL, IMUL признаки OF, CF устанавливаются в единицу, если разрядность результата превышает разрядность операндов (8, 16 или 32), определяемую режимом работы микропроцессора, префиксным байтом или значением бита w в КОП. Команды MUL, IMUL различаются условиями установки этих признаков. При выполнении беззнакового умножения (MUL) устанавливается (OF)=(CF)=1, если старшая половина произведения не равна 0, (OF)=(CF)=0, если все разряды старшей половины равны нулю. При выполнении знакового умножения (IMUL) устанавливаются (OF)=(CF)=1, если старшая половина произведения не является расширением знака младшей половины, (OF)=(CF)=0, если во всех разрядах старшей половины повторяются значения старшего бита (знака) младшей половины.

Команды MUL и IMUL используют алгоритм быстрого окончания, который досрочно завершает выполнение команды, если старшие разряды множителя равны нулю. Таким образом время выполнения команды зависит от значения множителя. В табл. 4.2 указаны минимальное и максимальное число тактов умножения.

Команды деления DIV и IDIV имеют только одноадресную форму, причем разрядность делимого, которое размещается в аккумуляторе с расширением, должна вдвое превышать разрядность делителя (8, 16 или 32 разряда). Используемое при этом размещение делимого и результата указано в табл. 4.4.

Таблица 4.4. Размещение делимого и результатов деления

Разрядность делителя	Делимое		Частное	Остаток
	Старшие разряды	Младшие разряды		
8	AH	AL	AL	AH
16	DX	AX	AX	DX
32	EDX	EAX	EAX	EDX

Знак остатка при выполнении команды IDIV устанавливается равным знаку делимого. Дробное частное округляется до целого значения отбрасыванием его дробной части. Если делитель равен нулю или разрядность частного

превышает разрядность аккумулятора (AL, AX или EAX),то выполняется прерывание типа 0 (ошибка деления).

Команды CBW/CWDE и CWD/CDQ осуществляют преобразование хранящегося в аккумуляторе байта (AL), слова (AX) или двойного слова (EAX) соответственно в одинарное, двойное или четверное слово (16, 32 или 64 разряда) путем расширения знакового разряда. При выполнении команды CBW/CWDE расширенный операнд остается в аккумуляторе: (AL) в AX или (AX) в EAX. При выполнении команды CWD/CDQ расширение (AX), (EAX) производится в регистры DX или EDX, куда заносится старшая половина (расширенный знак) результата. Команды CBW, CWD реализуются, когда МП 8036 работает в режимах реальных адресов или виртуального 8086 (обработка 16-разрядных слов). Команды CWDE, CDQ выполняются в защищенном режиме (обработка 32-разрядных слов). Эти операции обычно производятся перед выполнением команды IDIV для образования из 8-, 16- или 32-разрядных операндов делимого удвоенной разрядности.

Команды AAA, AAS, AAM, AAD обеспечивают получение правильного результата арифметических операций над неупакованными двоично-десятичными числами от (0 до 9), значение которых представляется одним байтом. Такое представление чисел используется в коде ASCII (КОИ-7), который в младшей тетраде содержит двоичный код соответствующего десятичного числа (0...9), а в старшей тетраде - код 0011. Для получения правильного результата перед обработкой таких чисел в старшей тетраде устанавливается код 0000. Результат арифметических операций над двоично-десятичными числами размещается в регистре AX. После сложения, вычитания, умножения выполняются команды коррекции AAA, AAS, AAM, после чего в младшей тетраде регистра AL образуется двоичный код младшего десятичного разряда результата, а в младшей тетраде регистра AH - двоичный код старшего разряда результата. Например, после умножения чисел 9 и 7, которые в неупакованной двоично-десятичной форме задаются кодами 0000 1001 и 0000 0111, в регистре AL будет содержаться произведение 001111111=63, которое после коррекции с помощью команды AAM будет представлено в регистре AX в следующем виде: (AH)=0000 0110, (AL)=0000 0111 (соответственно 6 и 3 в двоично-десятичном

коде). Команда AAD, в отличие от AAA, AAS, AAM, выполняется перед командой DIV. При этом двоично-десятичное число, старший десятичный разряд которого расположен в регистре AH, а младший - в регистре AL; преобразуется таким образом, чтобы после команды DIV содержимое AL, AH представляло младший и старший десятичные разряды частного в неупакованной двоично-десятичной форме.

Команды DAA и DAS осуществляют коррекцию результата сложения и вычитания чисел в упакованной двоично-десятичной форме. При этом старшая и младшая тетрады байта содержат двоичные коды старшего и младшего разряда двухразрядного десятичного числа. Например, число 63 в данной форме будет представлено в регистре AL кодом 0110 0011. После сложения упакованных двоично-десятичных чисел 63=0110 0011 и 18=0001 1000 в регистре AL будет получен код 0111 1011, который после выполнения команды DAA преобразуется в двоично-десятичную форму 1000 0001=91. При выполнении команд DAA и DAS учитывается значение признака переноса между тетрадами (AF). После их выполнения устанавливается (AF)=0. Следует отметить, что при использовании двоично-десятичного представления чисел соответствующие команды коррекции должны выполняться и после команд INC и DEC.

4.4. Команды логических операций и сдвигов

Команды логических операций и сдвигов даны в табл. 4.5.

Таблица 4.5. Команды логических операций и сдвигов

Команда	Формат	Число тактов	
NOT-инверсия (логическое НЕ)	1111011w mod 010 r/m	2/6	2/6
AND-конъюнкция (логическое И)			
(R)<-(R)^(R)	001000dw mod reg r/m	2	2
(M)<-(R)^(M)	0010000w mod reg r/m	7	7
(R)<-(R)^(M)	0010001w mod 100 r/m	6	6
(R/M)<-(R/M)^(im8)	1000000sw mod 100 r/m im8,16,32	2/7	2/7
(A)<-(A)^(im)	0010010w im 8,16,32		
OR-дизъюнкция (логическое ИЛИ)			

Таблица 4.5. (продолжение)

Команда	Формат	Число тактов	
(R)<-(R)∨(R)	000010dw mod reg r/m	2	2
(M)<-(R)∨(M)	0000100w mod reg r/m	7	7
(R)<-(R)∨(R)	0000101w mod 100 r/m	6	6
(R/M)<-(R/M)∨(im)	100000sw mod 100 r/m im8,16,32	2/7	2/7
(A)<-(A)∨(im)	0000110w im 8,16,32	2	2
XOR-неравнозначность (исключающее ИЛИ)			
(R)<-(R)+(R)	001100dw mod reg r/m	2	2
(M)<-(R)+(M)	0011000w mod reg r/m	7	7
(R)<-(M)+(R)	0011001w mod 100 r/m	6	6
(R/M)<-(R/M)+(im)	100000sw mod 110 r/m im8,16,32	2/7	2/7
(A)<-(A)+(im)	0011010w im 8,16,32	2	2
TEST-логическое сравнение (установка признаков ZF,SF,PF)			
(R/M) ∧ (R)	1000010w mod 100 r/m	2/5	2/5
(R/M) ∧ (im)	1111011w mod 011 r/mim 8,16,32	2/7	2/7
(A) ∧ (im)	1010100w im 8,16,32		
SHL/SAL,SHR, SAR,ROL,ROR-циклические сдвиги			
(R/M) на 1 разряд	1101000w mod TTT r/m	3/7	3/7
(R/M) на (CL) разрядов	1101001w mod TTT r/m	3/7	3/7
(R/M) на (im) разрядов	1100000w mod TTT r/m	3/7	3/7
RCL и RCR циклический сдвиг через CF			
(R/M) на 1 разряд	1101000w mod TTT r/m	9/10	9/10
(R/M) на (CL) разрядов	1101001w mod TTT r/m	9/10	9/10
(R/M) на (im) разрядов	1100000w mod TTT r/m	9/10	9/10
SHLD -сдвиг влево двойной			
(R/M) на (im) разрядов	00001111 10100100 modregr/mim8	3/7	3/7
(R/M) на (CL) разрядов	00001111 10100101 mod reg r/m	3/7	3/7
SHRD -сдвиг вправо двойной			
(R/M) на (im) разрядов	00001111 10101100 modregr/mim8	3/7	3/7
(R/M) на (CL) разрядов	00001111 10101101 mod reg r/m	3/7	3/7

Команда NOT производит инвертирование содержимого адресуемого регистра или ячейки памяти. При этом значения признаков не изменяются.

Логические операции конъюнкции И (AND), дизъюнкции ИЛИ (OR), Исключающее ИЛИ (XOR) выполняются над значением каждого из разрядов адресуемых операндов. Команда проверки TEST реализует конъюнкцию операндов без изменения их значений и устанавливает соответствующие признаки (аналогично арифметической команде CMP).

Микропроцессор выполняет несколько типов команд сдвига, реализующих одноразрядные сдвиги или сдвиги на заданное число разрядов от 0 до 31, которое определяется содержимым регистра (CL) или значением байта (im 8) в команде (переменная разрядность).

Однсоперандные команды сдвига содержат в байте MODR/M трехбитовое поле TTT, определяющее выбор выполняемой операции (табл. 4.6). Команда SHL/SAL осуществляет сдвиг адресуемого операнда, размещенного в регистре или памяти, влево на заданное число разрядов; освобождающиеся младшие разряды заполняются нулями. Команды SHR, SAR сдвигают адресуемый операнд вправо. При выполнении команды SHR освобождающиеся старшие разряды заполняются нулями. Команда SAR сохраняет знак операнда; при этом освобождающиеся слева разряды заполняются значениями старшего (знакового) разряда.

Таблица 4.6. Коды операций сдвига

TTT (КОП)	Команда
000	ROL
001	ROR
010	RCL
011	RCR
100	SHL/SAL
101	SHR
111	SAR

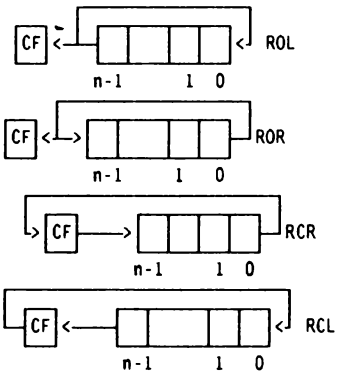


Рис.4.1. Выполнение команд циклических сдвигов

Выполнение команд циклических сдвигов ROL, ROR, RCR, RCL иллюстрируется на рис. 4.1.

Команды SHLD, SHRD используют два операнда, адресуемых байтом MODR/M. Первый из них может храниться в регистре или памяти, второй в регистре. При выполнении команды SHLD первый операнд сдвигается влево на заданное число разрядов, а в его освободившиеся младшие разряды вводится соответствующее число старших разрядов второго операнда. При выполнении команды SHRD первый операнд сдвигается вправо, а в освобождающиеся старшие разряды последовательно вводятся младшие разряды второго операнда. При этом копия второго операнда сохраняется в соответствующем регистре.

Во всех видах сдвигов признак (CF) принимает значение последнего выдвигаемого бита (в командах SHLD, SHRD - последнего бита, выдвигаемого из *первого* операнда). При одноразрядных сдвигах устанавливается признак переполнения (OF)=1, если изменилось значение знака (старший разряд). При многоразрядных сдвигах признак (OF) остается неопределенным. Так как при выполнении команды SAR значение знакового разряда не меняется, то всегда устанавливается значение (OF)=0.

4.5. Команды битовых операций

Данная группа команд отсутствует в микропроцессорах младших моделей: 8086, 80186, 80286. Форматы и коды этих команд приведены в табл. 4.7.

Команды BT, BTS, BTR и BTC выбирают из адресуемого регистра или памяти значение определенного бита и копируют его в признак CF. Номер бита Nb определяется значением байта (im8) команды или задается содержимым регистра, который в свою очередь задается полем reg в байте MODR/M. При выполнении команды BT значение бита в исходном 16- или 32-разрядном слове остается неизменным; при выполнении BTS значение бита устанавливается в 1; при выполнении BTR - сбрасывается в 0; при выполнении BTC - инвертируется.

Таблица 4.7. Команды битовых операций

Команда	Формат	Число тактов	
BT -проверка бита			
NB <- (im)	00001111 10111010 mod 100r/mim8	3/6	3/6
NB <- (R)	00001111 10100011 mod reg r/m	3/12	3/1
BTS-проверка и установка бита			
NB <- (im)	00001111 10111010 mod 010r/mim8	6/8	6/8
NB <- (R)	00001111 10111011 mod reg r/m	6/13	6/13
BTR-проверка и сброс бита			
NB <- (im8)	00001111 10111010 mod110 r/mim8	6/8	6/8
NB <- (R)	00001111 10111011 mod reg r/m	6/13	6/13
BTC-проверка и инверсия бита			
NB <-(im8)	00001111 10111010 mod 111r/mim8	6/8	6/8
NB<- (R)	00001111 10111011 mod reg r/m	6/13	6/13
BSF-прямое сканирование битов	00001111 10111100 mod reg r/m	10+3n	10+3
BSR-обратное сканирование битов	00001111 10111100 mod reg r/m	10+3n	10+3

Команды BSF, BSR производят сканирование содержащегося в адресуемом регистре или ячейке памяти операнда и заносят в регистр-приемник, заданный полем reg в байте MODR/M, номер первого встреченного бита, установленного в 1. При выполнении команды BSF сканирование начинается с младшего разряда операнда, при выполнении BSR - со старшего разряда. Если операнд равен 0 (единичные биты отсутствуют), то устанавливается признак (ZF)=1. При этом содержимое регистра-приемника будет неопределенным. Если единичный бит найден, то устанавливается (ZF)=0. Число тактов, требуемое для выполнения этих команд, зависит от числа n=0...31, определяющего количество нулевых битов, предшествующих при сканировании первому единичному биту.

4.6. Команды операций со строками символов

При выполнении данной группы команд символом называется 8-, 16- или 32-разрядный операнд, являющийся

элементом строки, которая представляет последовательность символов, расположенных в смежных ячейках памяти.

Различаются два типа строк символов. *Строка - источник* размещается обычно в сегменте DS. Размещение этой строки в других сегментах (табл. 3.3) производится при наличии префикса SEG перед соответствующей командой. Относительный адрес символа в строке задается содержимым регистра SI. *Строка - приемник* всегда размещается в сегменте ES (табл. 3.3). Относительный адрес символа в этой строке задается содержимым регистра DI.

После выполнения очередной команды со строкой-источником содержимое (SI) изменяется на +1 или -1, +2 или -2, +4 или -4 в зависимости от разрядности символа (8, 16 или 32). Направление изменения адресов определяется значением признака (DF). Если (DF)=0, осуществляется увеличение адреса (автоинкремент) на 1, 2 или 4; если (DF)=1, осуществляется его уменьшение (автодекремент).

Если перед командами отсутствует префикс повторения, то соответствующие операции выполняются только для одного символа в строке, адресуемого содержимым регистров (SI) или (DI). Команда LODS осуществляет загрузку символа из строки - источника в аккумулятор (AL, AX или EAX). Команда STOS заносит содержимое аккумулятора в качестве символа в строку - приемник.

Команды INS и OUTS выполняют ввод символа в строку - приемник или вывод символа из строки - источника. Номер адресуемого порта задается при этом содержимым регистра DX. В защищенном режиме выполнение этих команд, также как команд IN, OUT (разд. 4.2) производится только, если уровень привилегий текущей программы CPL≤IOPL. Команда MOVS пересылает символ из строки - источника в строку - приемник. При выполнении команды CMPS происходит вычитание символа строки - приемника из символа строки - источника с установкой соответствующих признаков.

Команда SCAS производит аналогичное вычитание символа строки - приемника из содержимого аккумулятора. Установка признаков для команд CMPS, SCAS выполняется так же, как для команды CMP (табл. 4.3). Результат вычитания в командах CMPS, SCAS не сохраняется, поэтому эти команды используются для сравнения значений символов двух строк между собой или

сравнения символа строки - приемника и эталонного символа, размещенного в аккумуляторе.

Таблица 4.8. Команды операций со строками символов

Команда	Формат	Число тактов	
LODS-загрузка символа в аккумулятор	1010110w	5	5
STOS-запись символа из аккумулятора	1010101w	4	4
MOVS-пересылка символа	1010010w	7	7
INS - ввод символа	0110110w	8	8
OUTS - вывод символа	0110111w	7	7
SCAS - сканирование символов	1010111w	7	7
CMPS-сравнение символов	1010011w	10	10
REP LODS-Загрузить в аккумулятор	11110010 1010110w	5+6n	5+6n
REP STOS - запомнить из аккумулятора	11110010 1010101w	5+5n	5+5n
REPMOVS - Пересылка строки	11110010 1010010w	7+4n	7+4n
REPOUTS-вывести строку в DX-порт	11110010 0110111w	5+5n	5+5n
REPINS-ввести строку из DX-порта	11110010 0110110w	6+6n	6+6n
REPE CMPS - Сравнение строк (Поиск совпадений символов)	11110010 1010011w	5+9n	5+9n
RENPE CMPS - Сравнение строк (Поиск несовпадений символов)	11110011 1010011w	5+9n	5+9n
REPE SCAS - сканирование строки (Поиск совпадения с аккумулятором)	11110011 1010111w	5+8n	5+8n
REPNE SCAS - сканирование строки (Поиск несовпадения с аккумулятором)	11110010 1010111w	5+8n	5+8n

Признаки повторения REP//REPE/REPZ и RERNE/REPNZ обеспечивают последовательное выполнение команд над символами строки. Количество повторений при этом определяется содержимым регистра (CX) или выполнением определенного условия. Так как регистр CX имеет 16 разрядов, то максимальная длина обрабатываемых строк составляет 65535 символов.

Префикс REP используется для последовательного выполнения команд LODS, STOS, INS, OUTS, MOVS. При этом

команды повторяются n раз, где n=(CX). В результате после выполнения команды REP LDOS в аккумулятор будет помещен n-ый символ строки - источника, а после REP STOS содержимое аккумулятора будет введено n раз в качестве n символов строки - приемника. Команды REP INS и REP OUTS осуществляют ввод n символов в строку - приемник или вывод n символов из строки - источника. Команда REP MOVS осуществляет пересылку n символов из строки - источника в строку - приемник.

Префиксы REPE/REPZ, REPNE/REPNZ используются перед командами CMPS и SCAS. Наличие префикса REPE/REPNZ останавливает повторение операций при обнаружении одинаковых символов в строках или в строке - источнике и аккумуляторе. Если одинаковые символы отсутствуют, то выполнение команды продолжается до конца строки. Префикс REPNE/REPNZ останавливает выполнение команд CMPS и SCAS при обнаружении неодинаковых символов.Если все символы одинаковы, то команды выполняются n=(CX) раз.

При повторении команд время их выполнения, естественно, возрастает. Поэтому в табл. 4.8 для повторяющихся команд указано число тактов с учетом n - числа повторений. При больших значениях n выполнение таких команд требует значительного количества времени. Чтобы при этом сократить время реакции на внешние прерывания микропроцессор воспринимает запросы прерывания INTR после обработки каждого символа строки.

4.7. Команды управления программой

К этой группе относится большое число команд, которые для удобства описания разобьем на несколько подгрупп.

Команды безусловной передачи управления (табл. 4.9) осуществляют безусловный переход (JMP), вызов подпрограммы (CALL) и возврат из подпрограммы (RET).

Если передача управления происходит внутри сегмента, то содержимое (CS) не меняется и реализуется переход к команде, расположенной в текущем сегменте кодов. Команды JMP и CALL с относительной адресацией производят увеличение значения (IP)

или (EIP) на указанную в команде величину d8, 16, 32, которая воспринимается как число со знаком. При вычислении нового относительного адреса команды (IP, EIP)+(d8, 16, 32) возникающий перенос игнорируется, поэтому полученный адрес остается в пределах текущего сегмента кодов. Команды JMP и CALL с косвенной адресацией загружают в IP или EIP содержимое регистра или ячейки памяти, адресуемое байтом MODR/M. Команда CALL перед этим заносит в стек текущее значение (IP) или (EIP), которое является относительным адресом для следующей команды программы. При поступлении команды RET производится восстановление из стека содержимого (IP), (EIP), необходимого для вычисления адреса возврата.

При передаче управления между сегментами меняется не только содержимое (EIP), (IP), но и базовый адрес сегмента кодов CS. При выполнении команд JMP и CALL с прямой адресацией новое содержимое (IP) или (EIP) задается двумя или четырьмя байтами (ip 16) или (eip 32) команды, базовый адрес определяется заданным значением 16-разрядного селектора sel 16. При косвенной адресации с помощью байта MODR/M адресуются ячейки памяти, в которых последовательно располагается новое содержимое (IP) или (EIP) и 16-разрядный селектор (sel 16), загружаемый в (CS). Отметим, что в этом случае не производится адресация к регистрам, т.е. в байте MODR/M поле mod=11.

Для возврата из подпрограммы может использоваться вариант команды RET с увеличением содержимого (SP). После восстановления из стека содержимого (IP) или (EIP) и, в случае необходимости, (CS), данная команда увеличивает содержимое (SP) или (ESP) на величину (d16), заданную в команде. Таким образом из стека исключается число ячеек памяти, равное (d16). Эта команда производит очистку сегмента стека от использованных в программе параметров, если они не требуются для продолжения выполнения основной программы.

Для команд безусловной и условной передачи управления (табл. 4.9 и 4.10) количество тактов, необходимых для их выполнения, зависит от m - числа байтов следующей команды. Эта зависимость связана с тем, что команды передачи нарушают установившуюся очередь команд, требуя дополнительное число тактов для загрузки в очередь следующей команды. Число тактов также значительно возрастает при выполнении этих команд в

защищенном режиме. Если при переходах сохраняется установленный уровень привилегий, то для выполнения команд JMP и CALL требуется nt=40-60 тактов. При переходах с изменением уровня привилегий и межзадачных переходах число тактов составляет 110-280 для команд JMP, CALL и около 80 для команды RET.

Таблица 4.9. Команды безусловной передачи управления

Команда	Формат	Число тактов	
JMP -безусловный переход			
внутри сегмента			
короткий	11101011 d8	7+m	7+m
относительный			
длинный	11101001 d16,32	7+m	7+m
относительный			
косвенный	11111111 mod 100 r/m	7+m/ 10+m	7+m/ 10+m
JMP -безусловный переход между сегментами			
прямой	11101010 ip16,eip32 sel16	12+m	23
косвенный	11111111 mod 101 r/m	17+m	28
внутри сегмента			
CALL -вызов подпрограммы			
внутри сегмента			
относительный	11101000 d16,32	7+m	7+m
косвенный	11111111 mod 010 r/m	7+m/ 10+m	7+m/ 10+m
CALL -вызов подпрограммы между сегментами			
прямой	10011010 ip16,eip32 sel16	17+m	35
косвенный	11111111 mod 011 r/m	22+m	40
RET -возврат из подпрограммы			
внутри сегмента			
без увеличения (SP)	11000011	10+m	10+m
с увеличением (SP)	11000010 d16	10+m	10+m
RET -возврат из подпрограммы			
между сегментами			
без увеличения (SP)	11001011	18+m	35
с увеличением (SP)	11001010 d16	18+m	35

Команды условных переходов $J_{усл.}$ (табл. 4.10) выполняют переходы только в пределах текущего сегмента с использованием относительной адресации. При выполнении заданного условия (табл. 4.11) производится переход к команде, относительный адрес которой в сегменте кодов равен (IP)+(d8, 16) или

(EIP)+(d32), где d8, 16, 32 представляет 8-, 16- или 32-разрядное смещение со знаком. Таким образом допускаются переходы к ячейкам памяти, размещенным в текущем сегменте кодов перед или после команды Jусл. Возникающие при вычислении этих относительных адресов переносы не учитываются. Имеется две модификации команд Jусл. (табл. 4.10): короткие и длинные переходы.Первая модификация использует 8-разрядное смещение (d8), как в микропроцессорах 8086, 80186, 80286.

Вторая модификация использует 16- или 32-разрядное смещение (d16, 32) в зависимости от разрядности адреса, устанавливаемой режимом работы микропроцессора или соответствующим префиксом перед командой Jусл. подгруппу входят 16 команд, различающихся условием выполнения перехода, которое задается четырехбитовым полем cond в первом или втором байте КОП. Кодировка условий приведена в (табл. 4.11). Соответствующие мнемокоды условий добавляются к букве J (JUMP), образуя мнемокод команды, например: J+O=JO - мнемокод команды условного перехода по переполнению. Условие определяется значением отдельных признаков или их комбинацией.

Таблица 4.10. Команды условных переходов, условной установки байтов и организации циклов

Команда	Формат	Число тактов	
Jусл.-усповный переход			
короткий	0111cond d8	7+m или 3	7+m или 3
длинный	00001111 1000cond d16,32	7+m или 3	7+m или 3
SETусл.-установка байтов	00001111 1001cond mod000r/m	4/5	4/5
JCXZ,JECXZ-повторение цикла,если CX, ECX = 0	11100011 d8	9+m или 5	9+m или 5
LOOP -цикл пока (CX)<>0; (CX)<-(CX)-1	11100010 d8	11+m	11+m
LOOPZ/LOOPE - цикл пока (CX)<>0 и ZF=1; (CX)<-(CX)-1	11100001 d8	11+m	11+m
LOOPNZ/LOOPNE - цикл пока (CX)<>0 и ZF=0; CX)<-(CX)-1	11100000 d8	11+m	11+m

В качестве условия часто используется относительное значение двух операндов, определяемое с помощью команды сравнения CMP. Равенство операндов определяется значением признака нуля (ZF) после команды CMP: равно E/Z (результат равен 0); не равно NE/NZ (результат не равен 0). Если сравниваются *беззнаковые операнды*, то соотношение *выше- ниже* между ними определяют условия:

$<$ B/NAE; $\geqslant$ NB/AE;
$\leqslant$ BE/NA; $>$ NBE/A.

Отметим, что команды JB/JNAE и JNB/JAE эквивалентны командам переходов по наличию переноса JC или отсутствию переноса JNC. Если сравниваются *операнды со знаком*, то соотношения *больше- меньше* определяют условия:

$<$ L/NGE; $\geqslant$ NL/GE;
$\leqslant$ LE/NG; $>$ NLE/G.

Таблица 4.11. Коды условий

Мнемокод	Условие	Код	Мнемокод	Условие	Код
O O	Переполнение: (OF)=1	0000	NO	нет переполнения: (OF)=0	0001
B/NAE	ниже/ /не выше или равно:(CF)=1	0010	NB/AE	не ниже/выше или равно: (CF)=0	0011
E/Z	равно / нуль:(ZF)=1	0100	NE/NZ	не равно/не ноль	
BE/NA	ниже или равно/не выше: (CF)V(ZF)=1	0110	NBE/A	не ниже или равно/ выше:(CF)V(ZF)=0	0111
S	отрицательный знак:(SF)=1	1000	NS	положительный знак:(SF)	1001
P/PG	четность: (PF)=1	0110	NP/PO	нечетность:(PF)=0	1011 1011
L/NGE	меньше/не больше или равно: (SF)+(OF)=1	1100	NL/GE	не меньше/больше или равно: (SF)+(OF)=0	1101
LE/NG	меньше или равно/не больше:((SF)+ (OF))V(ZF)=1	1110	LNE/G	не меньше или равно/больше: ((SF)+(OF))V(ZF)=0	1111

Команды условной установки байтов $SET_{усл}$. (табл. 4.10) производят установку байта, размещенного в регистре или ячейке пямяти, в зависимости от выполнения условия, заданного четырехбитовым полем cond во втором байте КОП. При выполнении условия все разряды устанавливаются в 1, в противном случае - в 0. Адресация байта осуществляется в соответствии с полями mod, r/m байта MODR/M. Команды данной группы не выполняются микропроцессорами 8086, 80186, 80286.

Команды организации циклов (табл. 4.10) осуществляют условный переход в зависимости от состояния содержимого регистра (CX) или (ECX), которое автоматически уменьшается на 1 (декремент) при поступлении команд этой подгруппы: JCXZ (JECXZ), LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ. Таким образом, регистр CX (ECX) используется в качестве счетчика циклов. В зависимости от режима работы МП 80386 и наличия префиксов изменения разрядности адреса при выполнении команд организации циклов используются регистры IP или EIP, CX или ECX.

Команда LOOP производит декремент (CX)-1 и проверяет полученное значение (CX). Если (CX)≠0, то выполняется переход к команде, имеющей относительный адрес (IP)+d8 или (EIP)+d8 в текущем сегменте кодов. Если (CX)=0, то выполняется следующая команда программы. Команда LOOP обычно ставится в конце цикла, обеспечивая его выполнение (CX) раз. Команда JCXZ (JECXZ) также выполняет декремент (CX)-1, но осуществляет переход при получении (CX)=0. Поэтому данная команда ставится в начале цикла.

Команды LOOPE/LOOPZ, LOOPNE/LOOPNZ аналогичны команде LOOP, но используют дополнительные условия для повторения цикла. Команда LOOPE/LOOPZ обеспечивает выход из циклов при получении ненулевого результата, т.е. при установке (ZF)=0, до выполнения заданного числа циклов (CX). Если в каждом цикле реализуется нулевой результат, то выполняется (CX) циклов, как в команде LOOP. Команда LOOPNE/LOOPNZ прекращает выполнение циклов при получении нулевого результата, т.е. установке (ZF)=1.

Команды прерываний (табл. 4.12) обеспечивают переход к одной из 256 программ обработки исключений и прерываний.

При этом текущее содержимое регистров IP (EIP), CS и FLAGS (EFLAGS) заносится в стек. Каждая из программ обработки соответствует определенному типу исключения или прерывания который определяется 8-разрядным вектором (int 8). Этот вектор задается вторым байтом команды INT. Первые 32 вектора прерываний зарезервированы для обслуживания нештатных ситуаций (ошибки, отказы и т.п.), возникающих при выполнении программы (гл. 6). Для двух из этих ситуаций предусмотрены отдельные однобайтные команды INT3 и INTO. Команда INT3 имеет вектор 3 и используется для установки контрольных точек в процессе отладки программы. Программа, вызываемая этой командой, обычно обеспечивает останов микропроцессора и выдачу пользователю необходимой контрольной информации о его состоянии.

Таблица 4.12. Команды прерываний и управления признаками

Команда	Формат	Число тактов	
INT - прерывание с задаваемым вектором	11001101 int8	37	59-287
INT3 - прерывание в контрольной точке	11001100	33	59-283
INTO-прерывание по переполнению:(OF)=1	11001110	35	59-285
IRET - возврат из подпрограммы прерывания	11001111	22	38-271
CLI - запрещение прерываний	11111010	3	3
STI - разрешение прерываний	11111011	3	3
LAHF - загрузка признаков из регистра EFLAGS в регистр AH	10011111	2	2
SAHF - запись содержимого AH в в регистр EFLAGS	10011110	5	5
PUSHF-запись содержимого регистра признаков в стек	10011100	4	4
POPF-чтение содержимого регистра признаков из стека	10011110	3	3
CLC-сброс признака переноса: (CF)=0	11111000	2	2
CLD-сброс признака направления: (DF)=0	11111100	2	2
CMC - инвертирование признака переноса	11110101	2	2
STC-установка признака переноса: (CF)=1	11111001	2	2
STD-установка признака направления (DF)=1	11111101	2	2

Команда INTO проверяет значение признака OF и вызывает соответствующую программу, если (OF)=1, т.е. результат предыдущей операции (число со знаком) вышел за пределы разрядной сетки. Эта команда имеет вектор 4.

При выполнении команд INT, INT3, INTO выбирается соответствующий 8-байтовый дескриптор из таблицы прерываний, адресуемой с помощью регистра IDTR. Содержимое (IDTR) указывает базовый адрес и границы таблицы. Значение вектора прерываний, умноженное на 8, дает относительный адрес первого байта дескриптора в таблице IDT. Содержимое дескриптора используется для формирования адресов ячеек памяти, из которых выбираются новые значения (IP) или (EIP) и (CS). Для обеспечения защиты памяти при выполнении команд прерываний учитываются уровни привилегий программ. Для этого требуется значительное время, поэтому при работе в защищенном режиме число тактов выполнения команд существенно возрастает, особенно при различных уровнях привилегий программ или переключении задач (разд. 5.2).

Команды управления признаками (табл. 4.12) обеспечивают запись - чтение содержимого регистра признаков и их изменение. Команды LAHF и SAHF выполняют загрузку младшего байта из регистра EFLAG в регистр AH и заполнение младшего байта в EFLAGS из регистра AH. Команды POPF, PUSHF производят загрузку в стек или выборку из стека содержимого (FLAGS) при обработке 16-разрядных операндов или (EFLAGS) при обработке 32-разрядных операндов. Команды CLC, STC осуществляют установку значений признака переноса (CF)=0, (CF)=1, команда CMC его инвертирование. Команды CLD, STD устанавливают значения признака направления (DF)=0, (DF)=1. Команды CLI, STI выполняют, соответственно, установку значений признака разрешения прерываний (IF)=0 (запрещение прерываний), (IF)=1 (разрешение прерываний). В защищенном режиме изменение значения (IF) с помощью этих команд производится при выполнении условия CPL $\leqslant$ IOPL, как и для команд IN, OUT, INS, OUTS (разд. 4.2, 4.6). При нарушении этого условия реализуется прерывание типа 13 (нарушение защиты).

Необходимо отметить, что команды IRET, POPF в защищенном режиме могут изменить значение признака (IF), в регистре EFLAGS только при выполнении такого же условия:

CPL<IOPL. Содержимое поля IOPL в регистре EFLAGS эти команды могут изменять только при наиболее высоком уровне привилегий текущей программы: CPL=0 (разд. 5.3).

4.8. Команды поддержки языков высокого уровня

Команды этой группы (табл. 4.13) реализованы в микропроцессорах 80286, 80386 для упрощения программирования на языках высокого уровня. Команда BOUND проверяет, находится ли содержимое регистра, адресуемое полем reg байта MODR/M в заданных пределах. Если содержимое регистра меньше нижнего или больше верхнего предела, то микропроцессор переходит к выполнению программы обработки прерывания с вектором 5 - "превышение границы" (гл.6) Содержимое регистра рассматривается как число со знаком. Значения верхнего и нижнего предела размещаются в двух последовательно расположенных 16- или 32-разрядных ячейках памяти, адресуемых с помощью байта MODR/M. При этом не допускается использование регистровой адресации, т.е. в байте MODR/M, должно быть указано значение поля mod=11. Если команда будет иметь поле mod=11, то выполняется прерывание с вектором 6 - "неправильный код операции" (гл.6). Данная команда удобна для проверки нахождения индекса в заданных границах. Время ее выполнения составляет 10 или 44 такта (в зависимости от нахождения или не нахождения числа в заданных пределах).

Команда ENTER создает стековый кадр заданного размера для размещения параметров процедуры. Перед этим кадром в стек помещается список указателей на стековые кадры, созданные ранее выполняемыми вложенными процедурами. Таким образом обеспечивается возможность из текущей процедуры обращаться к параметрам других процедур. Размер стекового кадра задается содержимым байтов (d16), команды. Последний байт (I8) может иметь значения от 0 до 31 и указывает уровень вложенности текущей процедуры. Этот уровень определяет количество указателей (относительных адресов) предыдущих стековых кадров, помещаемых в стек перед образуемым стековым кадром. При поступлении команды ENTER содержимое регистра BP (или EBP) посылается в стек, а последнее значение

указателя стека SP (или ESP) заносится в BP (или EBP), после чего содержимое SP (или ESP) уменьшается на величину (d16), образуя стековый кадр заданного размера. Однако, если (I8) > 0, то после содержимого BP (или EBP) в стек заносится (I8) слов из последовательно расположенных 16- или 32-разрядных ячеек памяти, адрес которых задается содержимым BP (или EBP), последовательно уменьшаемым на 2 или 4. В этом случае число циклов выполнения команды зависит от количества заносимых в стек слов n = (I8).

При выполнении команды выхода из процедуры LEAVE содержимое регистра BP (или EBP) заносится в регистр SP (или ESP), восстанавливая старое значение указателя стека. Затем из стека извлекается старое значение BP (или EBP). Таким образом производится подготовка к возврату в предыдущую процедуру. Следующая за LEAVE команда RET продолжает выполнение предыдущей процедуры.

4.9. Команды управления защитой

В эту группу (табл. 4.13) объединены команды, выполняющие пересылку содержимого регистров таблиц глобальных и локальных дескрипторов (GDT и LDT), дескрипторов прерываний (IDT), младших 16 разрядов регистра управления CR0, содержащих слово состояния машины MSW (machine status word); команды, управляющие переключением задач; команды, изменяющие уровень привилегий, права доступа и границы сегментов, и проверяющие допустимость их записи и считывания.

Команда LDTR загружает из памяти в регистр GDT 32-разрядный линейный адрес начала и 16-разрядную границу (размер) таблицы глобальных дескрипторов. Байт MODRM команды определяет адрес младшего из загружаемых в регистр байтов. Команда SDTR посылает содержимое этого регистра в память, размещая его в ячейках памяти, начиная с адресуемой байтом MODRM. Команды LIDT и SIDT выполняют аналогичные процедуры с содержимым регистра таблицы прерываний, определяющим 32-разрядный начальный адрес и 16-разрядную границу таблицы дескрипторов прерываний (разд. 5.1.1).

Команда LLDT загружает в регистр LDT 16-разрядный селектор, определяющий выбор локального дескриптора из таблицы GDT. Загрузка производится из регистра или памяти, адресуемых байтом MODRM команды. Команда SLDT посылает содержимое регистра LDT в регистр или память, адресуемые байтом MODRM команды.

Команда LTR загружает в регистр задачи TR из памяти или регистра 16-разрядный селектор, с помощью которого в таблице GDT выбирается дескриптор, определяющий сегмент состояния выполняемой задачи TSS. Команда STR посылает селектор из регистра TR в память или регистр, адресуемый байтом MODRM.

CLTS сбрасывает в нуль бит 3 в регистре управления CR0, который служит флагом переключения задачи. Этот бит устанавливается в состояние (TS)=1 при каждом переключении задачи.

Таблица 4.13. Команды поддержки языков высокого уровня и управления защитой памяти

Команда	Формат	Число тактов	
BOUND-проверка границ массива	01100010 mod reg r/m	10или 44	10или 44
ENTER обращение к процедуре	11001000 d16, 18		
(18) = 0		10	10
(18) =1		12	12
(18) > 1		15+ 4(n-1)	15+ 4(n-1)
LEAVE-выход из процедуры	11001001	4	4
LGDT-загрузка регистра таблицы глобальных дескрипторов	00001111 00000001 mod010r/m	11	11
SGDT-запись содержимого регистра таблицы глобальных дескрипторов	00001111 00000001 mod000r/m	9	9
LIDT-загрузка регистра таблицы дескрипторов прерываний	00001111 00000001 mod011r/m	11	11
SIDT-запись содержимого регистра таблицы дескрипторов	00001111 00000001mod 001r/m	9	9

Таблица 4.13. (продолжение)

Команда	Формат	Число тактов	
прерываний			
LLDT-загрузка регистра таблицы покальных дескрипторов	00001111 00000000 mod010r/m	-	20/24
SLDT-запись содержимого регистра таблицы покальных дескрипторов	00001111 00000000 mod000r/m	-	2/2
LTR-загрузка регистра задачи	00001111 0000000 mod001r/m	-	2/2
STR-запись содержимого регистра задачи	00001111 00000000 mod001r/m	-	2/2
CLTS-сброс признака переключения задачи	00001111 00000110	2	2
LAR-загрузка байта доступа	00001111 00000010 modregr/m	-	5/16
LSL-загрузка границы сегмента	00001111 00000011 modregr/m	-	21 (25)/ 27 (26)
ARPL-коррекция запрошенного уровня привилегий	01100011 mod reg r/m	-	20/21
VERR-проверка доступности сегмента при чтении	00001111 00000000 mod100r/m	-	10/11
VERW-проверка доступности сегмента при записи	00001111 00000000 mod101r/m	-	15/16
LMSW-загрузка слова состояния машины (MSW)	00001111 00000001 modregr/m	10/13	10/13
SMSW-запись состояния машины (MSW)	00001111 00000001 mod100r/m	10/13	10/13

Команды LAR и LSL загружают в регистр, адресуемый полем reg байта MODRM, определенные фрагменты дескрипторов. Дескриптор выбирается с помощью селектора, содержащегося в памяти или регистре, адресуемых полями mod и r/m байта MODRM команды. По команде LAR из выбранного дескриптора

выделяется и загружается в регистр байт 5 дескриптора, называемый *байтом доступа* (разд. 5.1.2). Этот байт содержит информацию о наличии сегмента, его типе, уровне его привилегий DPL. Байт доступа загружается в разряды 8-15 адресованного регистра, остальные разряды которого принимают нулевое значение.

Команда LSL загружает в регистр значение границы (размер сегмента), определяемое двадцатью разрядами выбранного дескриптора. Если в дескрипторе бит дробности G=0, то граница определяется в байтах, и в регистр загружается 20-разрядное число, указывающее количество байтов в сегменте. Если в дескрипторе бит G=1, то соответствующие двадцать разрядов дескриптора определяют число страниц в сегменте. В этом случае микропроцессор преобразует число страниц в число байтов (1 страница=4096 байт) и полученное 32-разрядное число загружает в выбранный регистр. Выполнение команды при этом занимает несколько большее число тактов (указано в табл. 4.13).

После загрузки командами LAR и LSL байта доступа или границы сегмента в адресованный регистр устанавливается признак нуля (ZF)=1. Если выбранный этими командами дескриптор недоступен из-за нарушения уровня привилегий или границы таблиц GDT, LDT, то устанавливается (ZF)=0. Прерывание типа 13 (нарушение защиты) при этом не реализуется.

Команда ARPL выполняет сравнение значений двух младших разрядов (1-0) операндов, первый из которых хранится в памяти или регистре, второй в регистре. В качестве операндов используются селекторы, разряды (1-0) которых содержат запрашиваемый уровень привилегий RPL, имеющий значение от 00 (высший уровень) до 11 (низший уровень). Вторым операндом обычно служит селектор текущей программы, который предварительно перегружается из регистра CS в регистр, адресуемый командой ARPL. При этом младшие разряды регистра будут содержать уровень привилегий текущей программы CPL. Если RPL первого операнда-селектора меньше CPL, то значение RPL в этом операнде устанавливается равным CPL, т.е. уровень его привилегий снижается. При этом устанавливается признак нуля (ZF)=1, указывающий на возможную попытку нарушения защиты. Если (RPL)≥(CPL), то значение RPL сохраняется и признак нуля сбрасывается: (ZF)=0.

Команды VERR и VERW проверяют доступность выбранного сегмента при текущем уровне привилегий и допустимость чтения или записи в этом сегменте. Байт MODRM адресует память или регистр, где хранится селектор проверяемого сегмента. С помощью этого селектора выбирается дескриптор сегмента, байт доступа которого содержит бит1, определяющий право чтения или записи (рис. 5.4б,в). Для сегментов кода (CS) этот бит разрешения считывания R, для сегментов данных (DS,ES,FS,GS)- бит разрешения записи W. Таким образом команда VERR используется для проверки доступности сегментов кодов, команда VERW сегментов данных.Если выбранный сегмент при текущем уровне привилегий недоступен, или чтение/запись в них запрещены, то признак нуля сбрасывается: (ZF)=0. При этом не возникает исключения по нарушению защиты. Если сегмент доступен, то устанавливается (ZF) =1.

Команда LMSW загружает из памяти или регистра *слово состояния машины* MSW в младшие 16 разрядов регистра управления CRO. Так как младший бит MSW управляет включением защиты, то данная команда используется для перехода в защищенный режим. Команда SMSW пересылает содержимое 16 младших разрядов CRO в память или регистр, адресуемые байтом MODRM.

Отметим, что команды LLDT, SLDT, LTR, STR, LAR, LSL, VERR, VERW выполняются *только в защищенном режиме.* Попытка их выполнения в реальном режиме вызовет прерывание типа 6 (неразрешенный код команды).

Команды LGDT, LLDT, LIDT, LTR, CLTS, LMSW при работе в защищенном режиме выполняются только при наиболее высоком уровне привилегий текущей программы: CPL = 0 (разд. 5.2). Невыполнение этого уровня вызывает прерывание типа 13 (нарушение защиты).

4.10. Команды управления процессором

Помимо традиционных команд останова HLT, отсутствия операции NOP в эту группу входят команды ожидания WAIT, передачи управления сопроцессору ESC и пересылка информации с участием управляющих, отладочных и тестирующих регистров MOV (табл. 4.15).

Команда ESC включает в работу арифметический сопроцессор 80387, который предназначен для быстрого выполнения операций над числами с плавающей точкой. Разряды TTT и LLL команды ESC определяют вид операции. При выполнении этих операций МП 80386 и сопроцессор работают совместно, обмениваясь необходимой информацией по шине D, а так же используя сигналы BUSY, PEREQ, ERROR (разд. 2.1). Команда ожидания WAIT останавливает работу МП 80386 до поступления от сопроцессора низкого уровня на вход BUSY#. Она используется для синхронизации работы МП 80386 и сопроцессора.

Команды MOV этой группы обеспечивают пересылку информации между регистрами общего назначения и специальными регистрами: управления CR0, CR2, CR3, отладки DR0-DR3, DR6, DR7 и тестирования TR6, TR7. Регистры общего назначения адресуются полем reg (табл. 3.1), специальные регистры - полем eee в соответствии с табл. 4.14.

Команды HALT и MOV этой группы выполняются только при наиболее высоком уровне привилегий программы: CPL=00. При всех других значениях CPL эти команды вызывают прерывание типа 13 нарушение защиты (гл. 6).

Таблица 4.14. Кодировка специальных регистров

Поле eee	Специальные регистры		
000	CR0	DR0	-
001	-	DR1	-
010	CR2	DR2	-
011	CR3	DR3	-
110	-	DR6	TR6
111	-	DR7	TR7

4.11. Префиксные байты

Команды могут иметь несколько различных префиксных байтов. Назначение ряда префиксных байтов было определено выше. Префиксы изменения разрядности адресов и данных устанавливают для следующей команды их разрядность в соответствии с табл. 1.1.

Префикс SEG определяет выбор сегмента CS, DS, ES, FS, GS или SS для адресации операнда в последующей команде в

соответствии с заданным кодом (табл. 4.15). Этот префикс действует, если команда обращена к операнду в памяти; в противном случае он не учитывается.

Префикс блокировки LOCK вызывает установку на выходе LOCK микропроцессора низкого потенциала, запрещающего другим устройствам системы обращаться к общей шине в течение времени выполнения последующей команды. Префикс LOCK может предшествовать только командам ADD, ADC, SVB, SBB, AND, OR, XOR, DEC, INC, NEG, NOT, BT, BTC, BTR, BTS и XCHG. Использование префикса LOCK перед другими командами вызовет прерывание типа 6 - неразрешенный код операции. Следует отметить, что при выполнении операции XCHG сигнал блокировки LOCK вырабатывается и в случае отсутствия префикса LOCK.

Префиксы повторения REP/REPE и REPNE применяются совместно с командами операций над строками. Префикс REP используется с командами INS, OUTS,MOVS и STOS и вызывает их повторение до тех пор (ECX) $\neq$ 0 или (CX) $\neq$ 0. С командами CMPS и SCAS используются префиксы REPE или REPNE. Префикс REPE обеспечивает повторение команды до установки признака при (ZF)=0, т.е. до поступления различных операндов. Если одинаковые операнды в строках отсутствуют, то команды выполняются до получения (ECX)=0 или (CX)=0, т.е. до конца строки. Префикс REPNE вызывает повторение команды до установки признака (ZF)=1, т.е. до поступления одинаковых операндов. Выполнение команд с префиксами повторения может прерываться после обработки очередного элемента строки при поступлении сигнала INTR=1.

Таблица 4.15. Команды управления процессором и префиксные байты

Команда	Формат	Число тактов	
HLT - останов	11110100	5	5
NOP-отсутствие операции	00001111	3	3
WAIT-ожидание до поступления сигнала BUSY=1	10011011	7	7
ESC-подключение сопроцессора	11011TTT mod LLL r/m		
MOV-пересылка содержимого управляющих отладочных, тестирующих регистров			
CR0/CR2/CR3 из РГ	00001111 00100010 11eeereg	11/4/5	11/4/5
РГ из CR0-CR3	00001111 00100000 11eeereg	6	6
DR0-DR3 из РГ	00001111 00100011 11eeereg	22	22
DR6-DR7 из РГ	00001111 00100011 11eeereg	16	16
РГ из DR-DR7	00001111 00100001 11eeereg	14	14
РГ из DR0-DR3	00001111 00100001 11eeereg	22	22
TR6-TR7 из РГ	00001111 00100110 11eeereg	12	12
РГ из TR6-TR7	00001111 00100100 11eeereg	12	12
префикс изменения разрядности:			
данных	01100110	0	0
адреса	01100111	0	0
SEG : Выбор сегмента CS	00101110	0	0
SEG : Выбор сегмента DS	00111110	0	0
SEG : Выбор сегмента ES	00100100	0	0
SEG : Выбор сегмента FS	01100100	0	0
SEG : Выбор сегмента GS	01100101	0	0
SEG : Выбор сегмента SS	01100101	0	0
LOCK-блокировка магистрали	00110110	0	0
REP/REPE-повторение операций, пока (CX)≠0, (ZF)≠0	11110000	0	0
REPNE-повторение операций, пока (ZF)≠1	11110011	0	0
	11110010	0	0

Глава 5. МП 80386 в защищенном режиме

Возможности МП 80386 наиболее полно реализуются при работе в защищенном режиме. При этом обеспечивается пространство линейных адресов до 2^{32} 4Гбайт и доступ к виртуальной памяти до 2^{46} = 64Гбайт. Помимо сегментации памяти в защищенном режиме может быть реализована ее страничная организация. Данный режим позволяет использовать дополнительные команды, введенные для эффективной поддержки многозадачных операционных систем (разд. 4.8). Кроме того, обеспечивается защита пользовательских программ друг от друга и от операционной системы, предотвращающая несанкционированное вмешательство в их работу.

5.1. Формирование адреса

В защищенном режиме логический адрес задается двумя компонентами: 16-разрядным селектором, определяющим базовый адрес сегмента и 32- или 16-разрядным относительным адресом, указывающим положение выбираемой ячейки памяти в сегменте.

Формирование физического адреса ячейки памяти производится с использованием двух механизмов: *сегментации* (segmentation) и *страничной организации* (paging).

5.1.1. Сегментация

Сегментацией называется один из способов организации памяти, при котором отдельные фрагменты программного обеспечения и массивов данных хранятся в изолированных областях - *сегментах* памяти. Каждый из сегментов характеризуется соответствующими *атрибутами*, которые определяют локализацию данного сегмента в общем пространстве адресуемой памяти и правила обращения к нему. В МП 80386 атрибуты сегмента представлены в виде 8-байтной структуры данных, называемой *дескриптором*.

Дескрипторы сегментов хранятся в памяти в составе массивов данных, сформированных в виде таблиц. МП 80386 использует три типа таблиц дескрипторов:

GDT - глобальная таблица дескрипторов;
LDT - локальная таблица дескрипторов;
IDT - таблица дескрипторов прерываний.

Таблица GDT содержит любые типы дескрипторов, кроме используемых при обработке прерываний (гл. 6). В GDT заносятся дескрипторы сегментов, которые могут использоваться системой при выполнении различных задач. Таблицы LDT содержат дескрипторы сегментов, используемых при решении данной задачи. Количество создаваемых таблиц LDT определяется операционной системой и зависит от числа реализуемых задач. В принципе, каждая задача может иметь отдельную LDT, которая включает дескрипторы сегментов, используемых при ее решении. Кроме того, LDT могут содержать специальные дескрипторы *шлюзов* (gates) - точек входа в программы или задачи, которые используются при вызовах программ (подпрограмм) и переключениях задач (разд. 5.3). В случае совместного использования ряда сегментов таблицы LDT могут полностью или частично перекрывать друг друга. Таблица IDT используется для реализации прерываний (гл. 6). Таблицы могут иметь размеры от 8 байт до 64 Кбайт, т.е. содержать от 1 до 8192 дескрипторов.

Обращение к дескриптору осуществляется с помощью *селектора*, содержащегося в соответствующем сегментном регистре: CS, DS, SS, ES, FS, GS. Селектор представляет собой 16-разрядный указатель, который имеет три поля (рис. 5.1).

RPL поле (биты 0-1) определяет *уровень привилегий* запроса. Это двухразрядный код, указывающий допустимый уровень защиты сегмента, который может быть выбран с помощью данного селектора. Используемый в МП 80386 механизм защиты памяти подробно описан в разд. 5.2.

TI поле (бит 2) служит *индикатором таблицы*. Его значение указывает на таблицу: GDT при TI=0, LDT при TI=1.

INDEX поле (биты 3-15) служит *индексом* для выбора одного из 8192 дескрипторов, содержащихся в таблице.

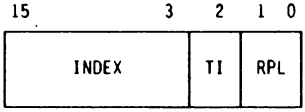


Рис.5.1. Формат селектора

Селектор с нулевым значением разрядов 2-15 называется *нуль-индикатором*. Он указывает на первый дескриптор в таблице GDT. Этот дескриптор не используется МП 80386, поэтому при его выборке выполняется прерывание.

С каждой из таблиц связан соответствующий регистр: GDTR, LDTR, IDTR (разд. 1.2.6). Регистр GDTR 48-разрядный; 32 разряда задают базовый адрес таблицы дескрипторов, а 16 указывают ее обьем в байтах (границу таблицы). Если селектор обращается к таблице GDT (TI=0), то его индекс, сдвинутый на три разряда влево (т.е. умноженный на 8 - число байтов в дескрипторе), служит в качестве *смещения* для формирования адреса дескриптора. Это смещение сравнивается с хранящейся в GDTR границей таблицы и, если смещение превышает границу, вырабатывается соответствующее прерывание (гл. 6). Если нарушения границы нет, то смещение прибавляется к содержащемуся в GDTR базовому адресу, в результате образуется логический адрес младшего байта выбираемого дескриптора. Регистр LDTR содержит 16-разрядный указатель, определяющий размещение в GDT дескриптора используемой таблицы LDT. Этот указатель используется как смещение при формировании адреса дескриптора LDT, выбираемого из GDT. Дескриптор LDT содержит 32-разрядный базовый адрес используемой LDT и ее 16-разрядный размер (границу), которые при загрузке регистра LDTR выбираются из GDT и хранятся во внутренних программно недоступных регистрах МП 80386. Если селектор обращается к LDT (TI=1), то его индекс, сдвинутый на три разряда влево, используется в качестве смещения при образовании адреса дескриптора выбираемого сегмента. Величина этого смещения сравнивается с границей используемой LDT. Если смещение не превышает границу, то адрес дескриптора формируется путем сложения базового адреса LDT и смещения. При превышении границы вырабатывается прерывание типа 13.

Загрузка регистров GDTR, LDTR из памяти, а также сохранение в памяти их содержимого реализуется с помощью команд LGDT, LLDT и SGDT, SLDT. Загрузка селекторов в регистры сегментов данных DS, ES, FS, GS, SS производится командами LDS, LES,LFS,LGS, LSS. Регистр CS является программно недоступным, поэтому прямая загрузка в него (или выгрузка) селекторов для выбора сегментов программ невозможна. Начальное содержимое CS устанавливается при

инициализации системы и затем изменяется при выполнении команд межсегментных вызовов и переходов CALL, JUMP, и при переключении задач (разд. 5.3).

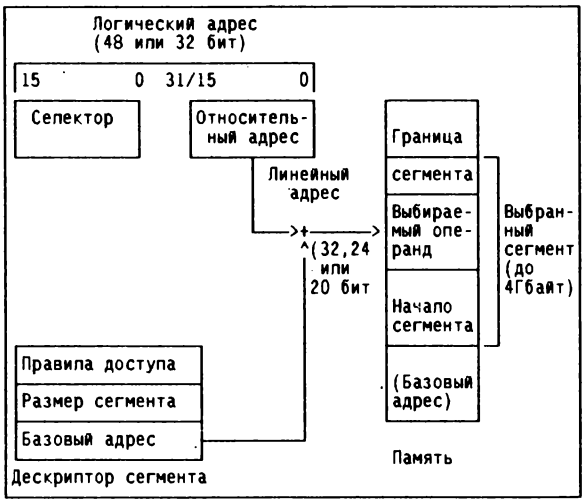


Рис. 5.2. Формирование линейного адреса

После выбора соответствующего дескриптора формирование линейного адреса осуществляется путем сложения базового адреса, содержащегося в дескрипторе и относительного адреса, образуемого в соответствии с используемым способом адресации (рис. 5.2). Способы адресации описаны в разд. 4.2.

При выполнении программ, написанных для МП 80386, базовый и относительный адреса содержат по 32 разряда. При выполнении программ, написанных для МП 80286, используется 24-разрядный базовый адрес, поэтому дескрипторы соответствующих программных сегментов должны содержать нули в старшем байте базового адреса. При этом формируется 16-разрядный относительный адрес. При работе в режиме VM 86 (виртуальный 8086) в качестве базового адреса используется селектор, сдвинутый влево на 4 разряда. Таким образом базовый адрес содержит 20 разрядов. Формируемый в этом режиме относительный адрес имеет 16 разрядов.

Если при работе МП 80386 не используется страничная организация памяти (бит 32 в регистре управления CR0 сброшен: PG=0), то полученный линейный адрес служит в качестве физического для выбора требуемой ячейки памяти.

Иногда при работе микропроцессорной системы сегментации памяти не требуется. В этом случае необходимо загрузить все регистры сегментов селекторами дескрипторов, имеющих нулевые базовые адреса и задающие размеры сегментов по 4 Гбайт. В результате каждый из сегментов использует полное адресное пространство, т.е. сегментация подавляется. При этом может быть реализована страничная организация памяти (разд. 5.1.3).

5.1.2. Дескрипторы

Помимо базового адреса и размера сегмента дескрипторы определяют ряд других важных его атрибутов. В общем виде формат дескриптора представлен на рис. 5.3, где 32-разрядный базовый адрес сегмента (база) и 20-разрядный размер сегмента (границы) размещены по частям в различных байтах дескриптора. Отдельные биты байта 6 дескриптора определяют следующие атрибуты сегмента.

G (бит дробности, бит 7) указывает, в каких единицах задан размер сегмента: в байтах при G=0 или страницах обьемом по 4 Кбайт при G=1. Таким образом сегмент может иметь размер до 2^{20}=1 Мбайт при G=0 или до 2^{32} =4 Гбайта при G=1.

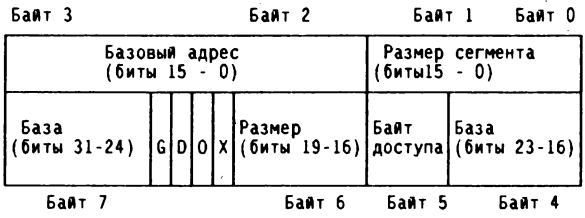


Рис. 5.3. Общий формат дескриптора

D (бит разрядности по умолчанию, бит 6) определяет разрядность формируемого относительного адреса или выбираемого операнда: 16 разрядов при D=0, 32 разряда при

D=1 (разд. 1.3). Если производится обращение к стеку (т.е. при выполнении команд типа PUSH, POP, CALL), то при D=0 используется 16-разрядный регистр SP, а при D=1, 32-разрядный регистр ESP.

Бит 5 в байте 6 всегда имеет нулевое значение, а бит 4 может иметь произвольное значение, устанавливаемое пользователем или операционной системой (обозначено x на рис. 5.3).

Байт 5 дескриптора определяет права доступа к выбираемому сегменту. В зависимости от содержимого сегмента байт доступа имеет различные форматы (рис. 5.4), хотя назначение ряда полей (битов) остается одинаковым. Одинаковое назначение имеют следующие биты и поля.

P (бит присутствия, бит 7) определяет наличие соответствующего сегмента в памяти. Если P=0 (сегмент отсутствует), то данный дескриптор не используется для формирования адресов, т.е. соответствующие байты дескриптора не загружаются в регистры, хранящие базовый адрес и размер сегмента. Поэтому содержимое этих байтов может быть установлено произвольно. Если в регистр сегмента поступает селектор дескриптора, имеющего P=0, то микропроцессор переходит к обработке соответствующего прерывания (гл. 6).

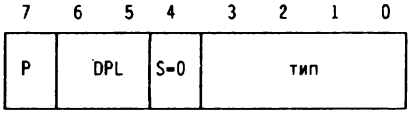
DPL (двухбитовое поле, биты 6-5) указывает *уровень привилегий дескриптора.* В зависимости от соотношения значений DPL и RPL, задаваемого в младших битах селектора (рис. 5.1), разрешается или запрещается обращение к данному сегменту. Таким образом обеспечивается требуемый уровень защиты сегмента (разд. 5.2).

S (системный бит, бит 4) определяет роль выбираемого дескриптора в системе. При S=1 дескриптор обеспечивает обращение к сегментам программ (кодов) или данных (включая стек). Дескрипторы с S=0, служат для обращения к таблицам LDT, сегментам состояния задачи TSS или шлюзам для входа в другие задачи или программы, включая программы обслуживания прерываний. Такие дескрипторы называются системными.

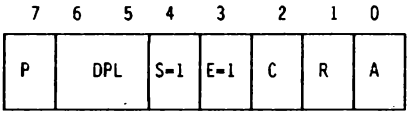
Формат дескриптора зависит от значения бита S.

Формат системных дескрипторов (S=0) приведен на рис.5.4а.

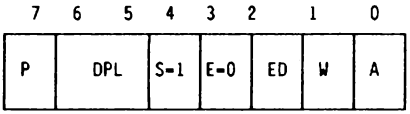
Поле типа системного дескриптора, биты 0-3 определяет его тип (табл. 5.1), указываемый в виде шестнадцатеричного числа 0-F.



а)



б)



в)

Рис.5.4. Формат байта доступа для системного дескриптора (а), дескрипторов программ (б) и данных (в)

Таблица 5.1. Тип системных дескрипторов

Тип	Определение	Тип	Определение
0	не используется	8	не используется
1	доступен TSS 80286	9	доступен TSS 80386
2	таблица LDT	A	зарезервирован
3	TSS 80286 занят	B	TSS 80386 занят
4	шлюз вызова 80286	C	шлюз вызова 80386
5	шлюз задачи (80286/386)	D	зарезервирован
6	шлюз прерываний 80286	E	шлюз прерываний 80386
7	шлюз ловушки 80286	F	шлюз ловушки 80386

Имеется три основных типа системных дескрипторов.

Дескрипторы таблиц LDT (тип 2) обеспечивают обращение к локальной таблице дескрипторов для выбора сегментов, используемых при выполнении текущей задачи.

Дескрипторы сегментов TSS (типы 1, 3, 9, B), используются при переключении задач в многозадачном режиме (разд. 5.2). При этом бит 3 байта доступа указывает, решается ли вызываемая задача средствами МП 80386 (типы 9, B) или МП 80286 (тип 1, 3). Бит 1 в байте доступа сегмента TSS называется *битом занятости* и указывает, является ли вызываемая задача занятой (уже находится в ряду решаемых задач) или доступной (еще не поступившей на выполнение). Этот бит дескриптора устанавливается в единицу, когда МП выполняет соответствующую задачу, и сбрасывается в нуль, когда МП переключается на другую задачу. Бит занятости используется в мультипроцессорных системах для предотвращения одновременного решения одной и той же задачи несколькими МП. Переключение на занятую задачу вызывает прерывание типа 13, указывающее на нарушение защиты.

Отметим, что дескрипторы LDT и TSS всегда должны иметь в байте 6 значение бита разрядности D=0 (рис. 5.3). Селекторы для выбора этих дескрипторов загружаются соответственно в регистры LDTR и TR. Для ускорения обращения к соответствующим сегментам дескрипторы LDT и TSS после их выбора хранятся в отдельных не доступных программисту регистрах.

Дескрипторы шлюзов (типы 4-7, C, E, F) используются для обращения к программам и задачам, а также при обработке прерываний и ловушек. Назначение шлюзов и форматы соответствующих дескрипторов рассмотрены в разд. 5.3 и гл. 6.

Дескрипторы программ и данных, формат байта доступа для которых показан на рис. 5.4б,в. имеют значения системного бита S=1. Они отличаются значениями *бита выполняемости E* (бит 3 в байте доступа), который равен единице (E=1) для сегментов программ и нулю (E=0) для сегментов данных.

Бит обращения A (бит 0 байта доступа) устанавливается в единицу при обращении к сегменту, т.е. при загрузке соответствующего селектора в сегментный регистр. Этот бит периодически проверяется операционной системой, реализующей виртуальную память, которая таким образом выявляет невостребованные сегменты, имеющие A=0. Сегменты, долгое время остающиеся невостребованными, выводятся из оперативной памяти на магнитный диск, освобождая место для других сегментов.

Назначение битов 1 и 2 байта доступа зависит от типа сегмента.

R (бит разрешения считывания, бит 1). Вводится для сегмента программ и разрешает при R=1 производить считывание его содержимого. При R=0 допускается только выборка содержимого этого сегмента для выполнения через регистр CS. Попытка считывания сегмента в этом случае вызовет прерывание типа 13. Отметим, что прерывание этого типа возникает также при попытке записи в сегмент программ независимо от значения бита R.

C (бит подчинения, бит 2) определяет дополнительные правила обращения, которые обеспечивают защиту сегментов программ (разд. 5.3).

W (бит разрешения записи, бит 1). Вводится для сегментов данных. Разрешает (при W=1) или запрещает (при W=0) изменение содержимого этих сегментов. При W=0 разрешается только считывание данных, а при попытке записи производится прерывание типа 13. Таким образом запись в сегмент программ запрещена. Если возникает необходимость внести изменение в этот сегмент, то можно создать сегмент данных с разрешением записи (W=1), занимающий то же адресное пространство, что и модифицируемый сегмент программ. После внесения изменений в созданный сегмент данных можно обратиться к нему как к сегменту программ.

ED (бит направления расширения, бит 2). Определяет размещение сегмента данных относительно базового адреса. При ED=0 (расширение вверх) данные в сегменте размещаются в направлении возрастания адресов от базового адреса до границы, определяемой суммой базового адреса и размера сегмента. При ED=1 (расширение вниз) данные в сегменте располагаются в направлении уменьшения адресов. Это реализуется в сегментах стека, где данные размещаются, начиная с ячейки, адрес которой равен базовому, увеличенному на максимальный размер сегмента: 64 Кбайт или 4 Гбайт в зависимости от значения бита дробности G в байте 6 дескриптора (рис. 5.3). Остальные ячейки стека имеют меньшие адреса, вплоть до нижней границы стека, адрес которой равен сумме значений базового адреса и размера сегмента, указанных в дескрипторе.

Таким образом, при расширении вверх (ED=0) относительный адрес ячейки сегмента должен быть меньше или

равен размеру сегмента, при расширении вниз (ED=1) относительный адрес должен быть больше размера сегмента.

5.1.3. Страничная организация памяти

МП 80386 реализует страничную организацию памяти, если в регистре управления CR3 бит 31 имеет значение PG=1. При этом сегмент разбивается на отдельные разделы, число которых может достигать $2^{10} = 1024$. Раздел может содержать до $2^{10} = 1024$ страниц объемом по 4 Кбайт каждая. Границы страниц жестко фиксированы, их начальные адреса имеют значения от 00000000H до FFFFF000H (шестнадцатеричное счисление).

Начальные адреса страниц данного раздела хранятся в соответствующей таблице страниц, содержащейся в памяти. Обращение к этой таблице производится с помощью каталога, в котором содержатся адреса таблиц страниц для всех разделов. Таким образом, страницы, содержащие определенный сегмент программ или данных, могут быть рассеяны по разным частям памяти, а их размещение определяется содержанием каталога разделов и таблиц страниц. При этом границы страниц и сегментов могут не совпадать.

Страничная организация обеспечивает более эффективное использование (заполнение) памяти по сравнению с сегментной, однако требует дополнительного времени и специальных аппаратных средств для преобразования адресов. Линейный 32-разрядный адрес при этом является исходной информацией для формирования физического адреса с помощью каталога разделов и таблиц страниц. Формирование физического адреса при страничной организации иллюстрируется рис. 5.5.

Линейный адрес при страничной организации рассматривается как совокупность трех полей (рис. 5.5). Поле TABLE (разряды A31-22 линейного адреса) указывает относительный адрес таблицы страниц выбираемого раздела в каталоге. Поле PAGE (разряды A21-12 линейного адреса) задает относительный адрес требуемой страницы раздела. Поле BYTE (разряды A11-0 линейного адреса) содержит относительный адрес выбираемого на странице байта.

Каталог занимает одну страницу памяти, где для каждого из 1024 возможных разрядов содержатся 32-разрядные указатели

входа в таблицу страниц этого раздела. Каждая из таблиц страниц также занимает одну страницу, где для каждой из 1024 страниц раздела даются 32-разрядные указатели входа в нее. Содержимое регистра управления CR3 задает старшие 20 разрядов адреса (A31-12) ячейки памяти, содержащей указатель входа в таблицу страниц раздела. Разряды A11-A2 адреса этой ячейки составляют относительный адрес, содержащегося в поле TABLE линейного адреса. Выбираемый при этом из каталога указатель имеет формат, представленный на рис. 5.5. Старшие 20 разрядов указателя задают базовый адрес, определяющий начало таблицы страниц. Поле PAGE определяет разряды A11-A2 адреса ячейки памяти, хранящей указатель входа в кадр выбираемой страницы. Этот указатель имеет такой же формат, как и указатель входа в таблицу (рис. 5.5). Его старшие 20 разрядов служат базовым адресом, определяющим адрес начала страницы (первого байта). Добавление к базовому адресу поля BYTE, задающего значения 12-ти младших разрядов адреса, позволяет получить физический адрес выбираемого байта. Таким образом, при страничной организации памяти 32-разрядный (физический) адрес формируется как сумма базового адреса, задаваемого указателем входа в кадр страницы, и относительного адреса, содержащегося в поле BYTE линейного адреса.

Помимо базового адреса указатели входа в таблицу страниц и кадр страницы содержат дополнительную информацию, определяющую порядок их использования. Эта информация задается значениями следующих битов.

P (бит присутствия, бит 0) разрешает использование таблицы страниц или кадра страницы при P=1. Если же P=1, то обращение к соответствующему разделу или странице запрещено, и попытка их использования вызовет прерывание типа 14 (отсутствие доступа к странице, гл. 6). Отметим, что при P=0 остальные биты указателя могут использоваться для представления какой-либо информации, используемой операционной системой.

R/W (бит чтения/записи, бит 1, U/S бит пользователя супервизора, бит 2) определяют права доступа к соответствующим разделу или странице для программ пользователя, имеющих минимальные привилегии (уровень 3, разд. 5.2.1). Если осуществляется запрос с уровнем привилегий 3, то при значении U/S=0 ему запрещается доступ к соответствующему разделу или

кадру страницы. Если U/S=1, то при значении R/W=0 разрешается только чтение раздела или страницы, а при R/W=1 и чтение, и запись. Отметим, что при запросах с большими привилегиями (уровни 0, 1, 2) допускается запись и чтение разделов и страниц при любых значениях U/S, R/W.

A (бит доступа, бит 5) автоматически устанавливается микропроцессором в состояние A=1 при обращении к данному разделу или странице для записи или чтения информации.

D (бит "мусора", бит 6) в указателе кадра страницы устанавливается в состояние D=1 при записи на данную страницу. Для указателей таблиц страниц значение бита D является неопределенным.

Биты A и D используются операционной системой, поддерживающей виртуальную память, для определения в оперативной памяти разделов и страниц, содержание которых подлежит замене из внешней памяти. Проверку и сброс этих битов выполняет операционная система

Биты 9-11 указателей зарезервированы для операционной системы (ОС), которая может использовать их для своих потребностей. Например, в этих битах может размещаться информация о времени последнего обращения к данному разделу или странице. Эта информация используется для определения разделов и страниц, подлежащих замене из внешней памяти.

Как указано выше, страничная организация памяти требует дополнительных затрат времени для преобразования линейного адреса в физический. Эти затраты будут весьма значительными, если в каждом цикле производить обращение к оперативной памяти для выбора указателей входа в таблицы и кадры страниц. Существенное сокращение времени преобразования адресов в МП 80386 достигается путем использования внутренней кэш-памяти, которая называется *буфером ассоциативной трансляции TLB*.

TLB представляет собой память с ассоциативной выборкой, которая содержит 20-разрядные базовые адреса 32 страниц, т.е. старшие 20 разрядов физического адреса. Каждый из базовых адресов имеет свой признак - *тег*. В качестве тега используются старшие 20 разрядов линейного адреса. Формирование физического адреса с использованием TLB производится следующим образом.

При поступлении в блок управления страницами линейного адреса его старшие 20 разрядов сравниваются с тегами базовых (физических) адресов, хранящихся в TLB. Если обнаруживается совпадение этих разрядов с каким-либо из тегов, то из TLB выбирается соответствующий этому тегу базовый адрес. Блок страничной организации формирует 32-разрядный физический адрес, в котором выбранный базовый адрес задает старшие 20 разрядов, а поле BYTE линейного адреса - младшие 12 разрядов.

Случай, когда базовый адрес страницы находится в TLB, называется *кэш-попаданием.* При этом не требуется обращаться к оперативной памяти для выбора указателей входа в таблицы и кадры страниц.

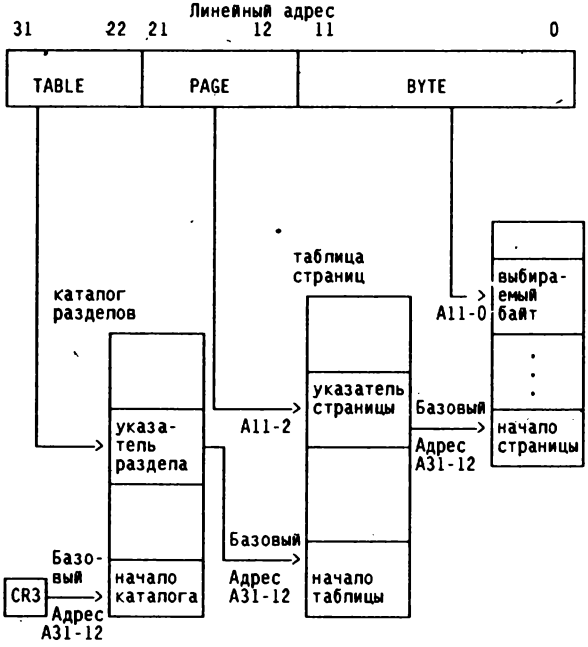


Рис.5.5. Формирование физического адреса при страничной организации памяти

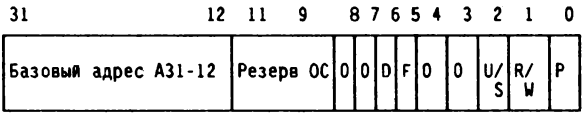


Рис. 5.6. Формат указателей входа в таблицу страниц или в кадр страницы

В результате формирование физического адреса не требует дополнительных машинных циклов. Если базовый адрес нужной страницы отсутствует в TLB, то такое обращение называется *кэш-промахом*. При этом микропроцессор выполняет описанную выше процедуру формирования физического адреса с помощью каталога разделов и таблицы страниц, затрачивая на это значительное время. Полученный из таблицы страниц 20-разрядный базовый адрес вместе с соответствующими 20-ю старшими разрядами линейного адреса *(тег)* заносятся в свободную ячейку TLB, или занимают ячейку, в которой хранится адрес, введенный в TLB ранее других. Таким образом обеспечивается непрерывное обновление содержимого TLB.

Помимо тега для каждого базового адреса страницы в TLB хранится дополнительная информация, определяющая правила доступа к ним. Эта информация задается следующими битами.

Бит достоверности V сбрасывается в состояние V=0 при записи нового содержимого в регистр управления CR3 (базовый адрес каталога разделов). Затем после преобразования очередного линейного адреса в физический для этого адреса в TLB будет установлено значение V=1, указывающее на допустимость использования данного базового адреса (ячейки TLB).

Бит мусора D принимает значение D=1 при выполнении записи на данную страницу.

Биты пользователя/супервизора U и записи/чтения W - определяют доступность данной страницы для программ пользователя (уровень привилегий 3) или системных программ (уровни привилегий 0, 1, 2), аналогично описанным выше битам U/S, R/W в указателе входа в страницу (рис. 5.6). Значения битов D, U, W устанавливаются в TLB одновременно с загрузкой соответствующего базового адреса из таблицы страниц при кэш-промахе. Таким образом TLB содержит всю

необходимую информацию для формирования физического адреса в случае кэш-попаданий.

Так как TLB хранит адреса 32 страниц объемом по 4 Кбайт, то МП 80386 может непосредственно формировать физические адреса для 128Кбайт памяти. Как показывает опыт применения микропроцессора, при этом доля кэш-попаданий составляет в среднем 98%. Только 2% обращений к памяти (кэш-промахи) требуют двухступенчатого преобразования адресов с использованием каталога и таблиц страниц.

При страничной организации МП 80386 контролирует возможность преобразования линейного адреса в физический и реализует прерывание типа 14, если преобразование не разрешено. Контроль осуществляется как при обращении к базовым адресам страниц, хранящимся в TLB (кэш-попадание), так и при формировании адресов страниц, отсутствующих в TLB (кэш-промах). Прерывание реализуется в следующих случаях:

1. Линейный адрес обращается к входу в каталог разделов или таблиц страниц, имеющих бит присутствия P=0. Соответствующий базовый адрес страницы при этом не может быть загружен в TLB.

2. Текущая процедура не имеет достаточного уровня привилегий для обращения к данной странице, что определяется битами U/S, R/W в каталоге разделов и таблице страниц, либо битами V, W в TLB.

В этих случаях в регистр управления CR2 заносится 32-разрядный линейный адрес, при преобразовании которого возникло прерывание. В стек программ обработки прерываний заносится 16-разрядный код ошибки страничного преобразования адреса, который имеет три младших значащих бита (рис. 5.7).

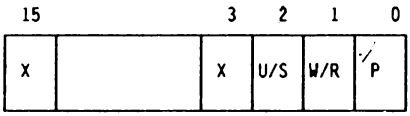


Рис.5.7. Формат кода ошибки страничного преобразования адреса

Бит P указывает на то, что ошибка вызвана либо обращением к отсутствующей странице (P=0), либо обнаружением защиты страницы (P=1). Бит W/R определяет,

что ошибка возникла при чтении (W/R=0) или записи (W/R=1) информации. Бит U/S показывает, что ошибка возникла при выполнении системных (U/S=0) или пользовательских (U/S=1) программ. Остальные биты кода ошибки (биты 15-3) имеют неопределенное значение.

5.2. Защита памяти

МП 80386 имеет специальные средства, обеспечивающие защиту памяти от несанкционированного доступа. Для защиты информации, хранящейся в памяти, используется ряд средств. Некоторые из них уже были упомянуты в предыдущих разделах, например, система *привилегий*, с помощью которой осуществляется защита сегментов.

5.2.1. Привилегии

Система привилегий регулирует доступ к тому или иному сегменту в зависимости от уровня его защищенности и от степени важности (привилегированности) запроса. В МП 80386, так же как и в 80286, установлены четыре *уровня привилегий PL*, которые задаются номерами от 0 до 3. Наиболее привилегированным является уровень с меньшим номером. Степень защищенности сегмента так же имеет четыре уровня, которые схематически представляются в виде вложенных колец защиты (рис. 5.8).

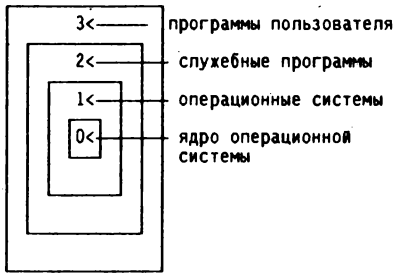


Рис.5.8. Уровни привилегий и защиты

Соответствующие уровни защищенности иллюстрируются на примере сегментов программ. Наименее защищенными (привилегированными) являются прикладные программы пользователя, для которых выделяется уровень с номером 3. Уровни с номерами 0, 1, 2 отводятся для системных программ (супервизора), которые можно разделить на три уровня в зависимости от требований к их защищенности. Наиболее защищенная часть - ядро операционной системы (ОС) имеет уровень 0. В ядро входит часть ОС, обеспечивающая инициализацию работы, управление доступом к памяти, защиту и ряд других жизненно важных функций, нарушение которых полностью выводит из строя микропроцессорную систему. Основная часть программ ОС должна иметь уровень 1. К уровню 2 обычно относят ряд служебных программ ОС, например, драйверы внешних устройств, системы управления базами данных, специализированные подсистемы программирования и др.

В соответствии с уровнями привилегий и защищенности установлены следующие правила доступа для сегментов программ и данных.

1. Данные из сегмента, имеющего уровень защиты PL могут быть выбраны программой, имеющей такой же или более высокий уровень привилегий.

2. Сегмент программ (прцедура), имеющий уровень защиты PL, может быть вызван программой, имеющей такой же или более низкий уровень привилегий.

Необходимо отметить, что правило 2 не выполняется при вызове сегментов программ, имеющих установленный бит подчиненности C=1 в байте доступа дескриптора (разд. 5.1.2, рис.5.4б.) Для таких подчиненных сегментов установлены правила вызова, изложенны в разд. 5.2.2.

Уровни защиты и привилегий определяются двумя битами, значение которых указывает номер кольца защиты или уровня. В зависимости от места размещения эти биты имеют различное назначение.

Уровень привилегий дескриптора DPL задается битами 5, 6 в байте доступа дескриптора, указывает уровень защищенности сегмента (номер кольца защиты).

Уровень привилегий запроса RPL задается битами 0 и 1 селектора. Значение RPL определяет уровень привилегий, инициатора запроса-обращения к сегменту. Инициатором является

программа или устройство, которое с помощью данного селектора обращается к памяти системы.

Текущий уровень привилегий CPL задается битами 0 и 1 селектора, размещенного в регистре CS. Определяет уровень привилегий выполняемого в настоящий момент сегмента программы.

Напомним, что меньшее значение DPL, RPL, CPL соответствует более высокому уровню привилегий (защиты). Наиболее защищенный сегмент имеет значение DPL=0, наименее защищенный - DPL=3. Для наименее привилегированных программ пользователя CPL=3, наиболее привилегированные программы ядра ОС имеют CPL=0.

Следует отметить, что МП 80386 имеет специальную команду ARPL, используемую для коррекции значения RPL селектора (разд. 4.9). При этом селектор загружается в регистр или ячейку памяти, адресуемую байтом MODRM команды ARPL. Селектор служит первым операндом команды. В качестве второго операнда в регистр, адресуемый байтом MODRM, загружается текущее значение CS. При выполнении команды сравниваются значения двух младших битов операндов, которые содержат RPL и CPL. Поле RPL селектора (первого операнда) принимает максимальное из значений RPL и CPL, т.е. устанавливается минимальный уровень привилегий запроса. Если значение RPL при этом изменялось, т.е. возросло до величины CPL, то устанавливается признак нуля ZF=1.

5.2.2. Обращение к сегментам данных, реализация передачи управления и ввода-вывода

Обращение к сегментам программ и данных в защищенном режиме производится с учетом описанной выше системы привилегий. При этом правила обращения зависят от типа сегмента.

Обращение к сегментам данных производится с помощью селекторов, загружаемых в регистры DS, ES, FS, GS. При обращении анализируются значения RPL селектора и CPL текущей программы. Эффективный уровень привилегий для запроса данных в этом случае определяется как максимальное из значений RPL и CPL. Обращение к запрашиваемому сегменту

данных разрешается, если его уровень защиты DPL≥max(RPL, CPL). Нарушение этого правила при обращении вызывает прерывание типа 13. Таким образом, задавая определенное значение RPL в селекторе, можно управлять доступом текущей программы к сегментам данных, имеющим различный уровень защиты. Например, при RPL=0 доступ к данным будет определяться значением CPL текущей программы. Если задать RPL=3, то при любом CPL программа может обратиться только к наименее защищенным сегментам данных, относящихся к кольцу 3.

Обращение к сегменту стека выполняется путем загрузки селектора в регистр SS. Этот селектор должен выбирать дескриптор сегмента с разрешенной записью, т.е. бит 1 в байте доступа должен иметь значение W=1 (разд. 5.1.2, рис. 5.4. в). Значения RPL и DPL должны быть равны CPL. Нарушение этих правил (обращение к сегменту, для которого W=0; использование селекторов и дескрипторов, имеющих значения RPL, DPL ≠ CPL) приводит к возникновению прерываний типа 13. При обращении к дескриптору с P=0 (отсутствующий сегмент) возникает прерывание типа 12.

Обращение к сегментам программ, т.е. передача управления реализуется при выполнении команд межсегментных переходов, вызовов, возвратов, прерываний и возвратов из прерываний. JMP, CALL, RET, INT, IRET. В разд. 4.7 даются правила передачи управления в защищенном режиме для межсегментных команд JMP, CALL, RET. Выполнение в защищенном режиме команд INT,IRET описывается в гл. 6. Правила обращения к сегментам программ зависят от значения бита подчиненности C, в байте доступа дескриптора (разд. 5.1.2, рис. 5.4б).

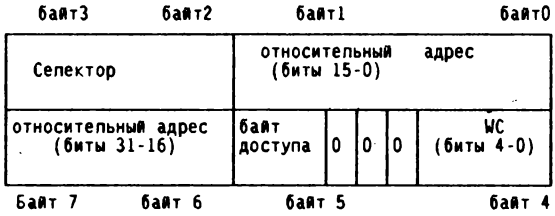
Обращение к подчиненным сегментам, для которых значение C=1, допускается только из программ, имеющих такой же или более низкий уровень привилегий. Таким образом, в программах с текущим уровнем привилегий CPL могут выполняться межсегментные команды JMP, CALL с передачей управления подчиненному сегменту, имеющему DPL≤CPL. При такой передаче значение CPL сохраняется, т.е. сохраняется уровень исходной (вызывающей) программы.

Обращение к неподчиненным сегментам, имеющим значение C=0, с помощью команд JMP и CALL допускается только в случае, если значение CPL текущей программы равно

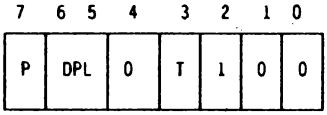
DPL сегмента. Кроме того, команда CALL может вызывать сегменты программ с более высоким, чем имеет текущая программа, уровнем привилегий, используя для этого специально установленные точки входа. Эти точки входа называются *шлюзами* (или вентилями) *вызова.*

Таким образом, менее привилегированная процедура может вызвать более привилегированную, обращаясь к ней через дескриптор шлюза вызова, определяющий доступную точку входа. Этот способ обращения, используемый также в МП 80286, позволяет программам пользователя обращаться за обслуживанием к операционной системе. При этом допускаются обращения только к определенным процедурам, которые санкционируются путем введения в систему соответствующего шлюза. Тем самым исключается возможность несанкционированного обращения менее привилегированных процедур к более привилегированным, что позволяет защитить их от возможных искажений.

Для вызова более привилегированной программы команда CALL должна обратиться к хранящемуся в LDT дескриптору шлюза вызова, формат которого показан на рис. 5.9а.



а)



б)

Рис.5.9. Формат дескриптора шлюза вызова (а) и его байта доступа. (б)

В байтах 2 и 3 дескриптора шлюза содержится селектор вызываемого сегмента программы, а байты 0, 1, 6, 7 задают относительный адрес шлюза- точки входа в эту программу. Байт доступа (рис. 5.9 б) содержит поле P и DPL, имеющие такое же значение, как и дескрипторах сегментов (разд. 5.1.2, рис. 5.4). Кроме того байт доступа шлюза вызова имеет бит типа процессора T (бит 3), который имеет значение T=0, если вызываемая программа написана для МП 80286, и T=1, если программа написана для МП 80386. Этот бит определяет механизм формирования линейного адреса, который зависит от типа микропроцессора (разд. 5.1.1).

Пятибитовое поле WC в байте 4 указывает количество параметров, которые переносятся из стека текущей программы в стек новой программы. Параметры представляют собой 16-разрядные (для программ МП 80286) или 32-разрядные (для программ 80386) слова. Число параметров, переносимых из старого стека в новый, может составлять от 0 до 31.

При вызове программ через шлюз должны выполняться следующие правила. Значения RPL селектора, вызывающего шлюз, и CPL текущей программы должны быть меньше или равны значению DPL в байте доступа шлюза. Таким образом дескриптор шлюза должен иметь меньший уровень привилегий, чем запрос и текущая программа. Вызванный сегмент программ, в котором находится шлюз вызова, должен иметь такой же или более высокий уровень привилегий, чем текущая программа, т.е. для этого сегмента DPL $\leqslant$ CPL.

Если эти правила выполняются, то после вызова командой CALL дескриптора шлюза в регистр CS загружается селектор - байты 2 и 3 этого дескриптора. Этот селектор выбирает из LDT дескриптор вызываемого сегмента программы. При этом младшие два бита (поле PL) селектора игнорируются, а вместо них в регистр CS в качестве CPL заносится значение DPL из дескриптора вызываемого сегмента. В регистр EIP из дескриптора шлюза загружается относительный адрес входа в программу - байты 0, 1, 6, 7 (для программ МП 80286 в IP загружаются байты 0, 1).

Если вызываемая программа имеет другой (более высокий) уровень привилегий, чем текущая, то при выполнении команды CALL для нее создается новый стек. При этом в регистры SS и ESP из сегмента состояния задачи TSS (разд. 5.3) загружается

новое содержимое, определяющее начальный адрес нового стека. В этот стек последовательно вводятся старые значения SS и ESP; параметры, переносимые из старого стека; старые значения CS и EIP. Число переносимых параметров определяется полем WC (рис. 5.9а), причем выбираются последние из загруженных в старый стек параметров. Последующие ячейки стека используются для хранения новых параметров.

По команде RET производится восстановление из стека старого содержимого регистров CS, EIP (IP) и SS, ESP (SP). Таким образом, одновременно с возвратом к исходной программе происходит и возвращение к старому стеку. Команда RET проверяет значение CPL и два младших разряда извлекаемого из стека старого содержимого CS, определяющие уровень привилегий программы, к которой осуществляется возврат. Команда выполняется, только если значение этих битов больше или равно CPL, т.е. если возврат осуществляется к программе с таким же или меньшим уровнем привилегий.

Нарушение командами JUMP, CALL, RET указанных выше правил обращения к сегментам программ вызывает прерывание типа 13 (нарушение защиты).

Реализация ввода - вывода с помощью команд IN, OUT, INS, OUTS в защищенном режиме производится с учетом CPL выполняемой программы. Величина CPL сравнивается с значением поля IOPL в регистре EFLAG. Ввод-вывод производится только при выполнении условия CPL $\leq$ IOPL, нарушение этого условия вызывает прерывание типа 13.

5.3. Многозадачность

Многозадачностью называется такой способ организации работы системы, при котором в ее памяти одновременно содержатся программы и данные для выполнения нескольких процессов обработки информации (задач). При этом должна обеспечиваться взаимная защита программ и данных, относящихся к различным задачам, а также возможность перехода от выполнения одной задачи к другой (переключение задач). МП 80386 имеет эффективные средства поддержки многозадачного режима, реализующие защиту и быстрое переключение задач. В качестве таких средств используется

специальная структура данных, организованная в виде сегмента, который называется *сегментом состояния задачи TSS*. Аппаратными средствами для поддержки многозадачности служит 16-разрядный *регистр задачи TR*, в который заносится селектор дескриптора TSS и связанный с TR программно недоступный 64-разрядный регистр, в который из GDT загружается 8-байтовый дескриптор TSS.

5.3.1. Сегмент состояния задачи

Каждая задача имеет свой сегмент TSS, структура которого (рис. 5.10) состоит из двух частей. Обязательная часть TSS объемом 104 байта содержит всю информацию, необходимую микропроцессору для решения данной задачи. Дополнительная часть может содержать какую-либо информацию о данной задаче, используемую операционной системой (имя задачи, комментарии и т.д.), и битовую карту, определяющую допустимые адреса устройств ввода-вывода. Рассмотрим содержание обязательной части TSS.

Первые два байта TSS используются для хранения селектора TSS предыдущей задачи, при выполнении которой произошел вызов данной задачи. В эти байты заносится содержимое регистра TR для предыдущей задачи, если предполагается возврат к ней. Отдельные поля TSS хранят содержимое всех регистров сегментов ES, CS, SS, DS, FS, GS, регистров общего назначения EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDS, регистра флагов EFLAGS и указателя команд EIP. При переключении задач содержимое указанных полей из вызванного TSS загружается в соответствующие регистры микропроцессора. При следующем переключении текущее содержимое регистров заносится в TSS данной задачи, после чего производится загрузка регистров из TSS новой задачи. Таким образом содержимое этих полей TSS обновляется при каждом переключении, фиксируя текущее состояние ее решения.

Содержимое ряда полей в обязательной части TSS не изменяется при решении задачи. Не изменяется поле, определяющее значение селектора LDT для данной задачи и содержимое регистра управления CR3, которое используется только при страничной организации памяти. Не изменяется

31	16	15	0	
0	0	Селектор возврата		0
ESP0				4
0		SS0		8
ESP1				C
0	0	SS1		10
ESP2				14
0	0	SS2		18
CR3				1C
EIR				20
EFLAGS				24
EAX				28
ECX				2C
EDX				30
EBX				34
ESP				38
EBP				3C
ESI				40
EDI				44
0	0	ES		48
0	0	CS		4C
0	0	SS		50
0	0	DS		54
0	0	FS		58
0	0	GS		5C
0	0	LDT		60
Относительный адрес БКВВ		0	0 T	64
Дополнительная информация для ОС				
Битовая карта ввода - вывода (БКВВ)				

Рис.5.10. Структура сегмента TSS

также содержимое полей SSO, ESPO, SSI, ESPI, SS2, ESP2, которые определяют начальный адрес стека при переключении к задачам с более высоким уровнем привилегий. Такое выделение отдельных стеков для задач с различными уровнями привилегий обеспечивает их более надежную защиту (разд. 5.2.2 и 5.3.2). Содержимое указанных полей TSS в процессе выполнения конкретной задачи загружается в соответствующие регистры.

Бит ловушки T в TSS вызывает при T=1 прерывание типа 1 при переключении на данную задачу.

Два последних байта в обязательной части TSS определяют относительный адрес начала битовой карты ввода-вывода (БКВВ) в TSS. Каждый бит БКВВ соответствует однобайтовому порту ввода-вывода. Так как МП 80386 обеспечивает обслуживание до 65536 портов, то полная БКВВ, определяющая возможность их обслуживания, будет представлять строку длиной 64 Кбит, занимающую 8 кбайт памяти. Поэтому относительный адрес БКВВ должен быть меньше или равен DFFF (шестнадцатеричная форма). При записи в бите БКВВ нуля разрешается обращение к соответствующему этому биту порту ввода-вывода. Если значение некоторого бита БКВВ равно 1, то при поступлении команд обращения к соответствующему порту микропроцессор реализует прерывание типа 13 (нарушение защиты).

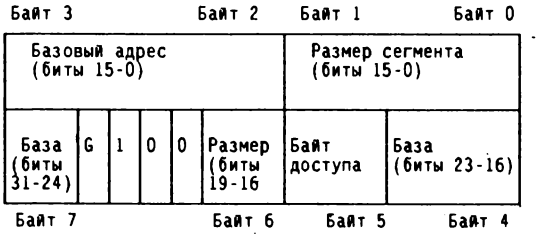
Таким образом БКВВ вводит для команд ввода-вывода дополнительный вид защиты (помимо выполнения условия CPL $\leqslant$ IOPL), который устанавливается для каждого порта индивидуально. Эту защиту можно обеспечить как для всех, так и для части портов. Если конец БКВВ выходит за границу сегмента, определенную его размером, задаваемым дескриптором TSS, то доступность соответствующих портов с высокими номерами не зависит от БКВВ. При этом БКВВ контролирует доступ только к портам с меньшими номерами, для которых биты БКВВ вошли в заданную дескриптором границу сегмента. Если значение размера сегмента, заданное дескриптором TSS, меньше указанного в TSS относительного адреса БКВВ, то эта карта не влияет на доступность портов, и ее можно полностью исключить.

За последним байтом БКВВ в TSS должен следовать заключительный байт, содержащий 1 во всех разрядах. Адрес этого байта должен соответствовать границе сегмента, определенной дескриптором TSS.

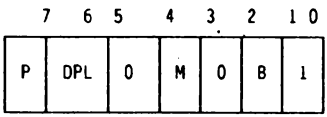
Объем дополнительной части TSS зависит от размеров применяемой БКВВ и количества служебной информации, используемой операционной системой. Этот объем определяется характером решаемой задачи. Во многих случаях дополнительная часть TSS вообще отсутствует.

Обращение к TSS осуществляется путем загрузки в регистр TR селектора, который адресует размещенный в GDT дескриптор TSS соответствующей задачи (рис. 5.11). Содержимое регистра TR можно загружать или заносить в память командами LTR, STR (разд. 4.9). Однако обычно команда LTR используется только при инициализации системы для установки начального содержимого TR. В процессе работы этот регистр загружается микропроцессором при выполнении команд, переключающих задачу (разд. 5.3.2).

Дескриптор TSS должен храниться только в таблице GDT. Поэтому обращение к дескриптору TSS с помощью селектора, имеющего бит TI=1 (индикатор таблицы LDT), вызовет прерывание типа 10. При загрузке селектора дескриптора TSS в какой-либо из регистров сегментов (CS, DS, SS, ES, FS, GS) также возникает прерывание этого типа.



а)



б)

Рис.5.11. Формат дескриптора TSS (а) и его байта доступа (б)

Содержимое сегмента TSS не может быть непосредственным образом считано или изменено. При необходимости считывания или изменения TSS необходимо сформировать дескриптор сегмента данных, имеющий те же аттрибуты (базовый адрес, размер и др.), что и TSS. Содержимое TSS в этом случае может быть считано или модифицировано путем обращения к нему как сегменту данных через сформированный дескриптор.

Формат дескриптора TSS и его байта доступа соответствуют приведенным на рис. 5.3 и 5.4а. Назначение полей P и G описаны в разд. 5.1.2. Бит M в байте доступа определяет тип задачи: при M=0 решаемая задача запрограммирована для решения на МП 80386. Бит занятости B устанавливается в состояние B=1 при переключении на данную задачу. Размер сегмента TSS, указанный в дескрипторе, должен быть не менее 104 байт (объем обязательной части). В противном случае при обращении к TSS происходит прерывание типа 10.

5.3.2. Переключение задач

Для переключения задач МП 80386 использует обычные команды межсегментного перехода JUMP, вызова CALL и возврата IRET, описанные в разд. 4.7. Если селектор (sel16) команды JUMP или CALL выбирает из таблицы GDT дескриптор, который имеет в поле типа 0001 (TSS для МП 80286) или 1001 (TSS для МП 80386), выполняется переключение задач. При этом селектор sel 16 заносится в регистр TR, а в связанный с ним программно-недоступный регистр загружается дескриптор TSS. В соответствии с содержимым обязательной части TSS производится загрузка регистров микропроцессора, и он начинает выполнение поступившей задачи. Следует отметить, что межсегментные команды JUMP и CALL содержат байты eip32, определяющие новое содержимое регистра EIP. Однако, если селектор выбирает дескриптор TSS, то эти байты игнорируются, а регистр EIP загружается содержимым соответствующего поля TSS. При переключении задачи устанавливается значение TS=1 бита 3 в регистре управления CR0, хранящем слово состояния машины MSW (разд. 1.2.5). Сброс этого бита в состояние TS=0 может

производиться командой CLTS или путем загрузки нового содержимого в CR0 командами LMSW или MOV.

Переключение задач производится, если бит занятости (бит1 в байте доступа) имеет значение B=0. При B=1 выполняется прерывание типа 13 (нарушение защиты). Установка B=1 в дескрипторе TSS производится командами JUMP или CALL, переключающими МП 80386 на выполнение данной задачи. При этом обращение других микропроцессоров в мультипроцессорной системе к этой задаче будет запрещено (вызовет прерывания типа 13).

При переключении задач с помощью команд JUMP или CALL должны выполняться правила привилегий, установленные для доступа к данным: DPL $\geqslant$ max (CPL, RPL). Таким образом допускается переключение на решение задач, чья степень защиты меньше или равна уровню привилегий текущей программы и запроса.

При использовании команды CALL возможно обращение к задачам с более высокой степенью защиты, чем уровень привилегий текущей программы, с помощью *шлюза задачи*. Использование шлюзов задачи аналогично, описанным в разд. 5.2 шлюзам вызова. Дескрипторы шлюза задачи размещаются в таблице LDT и имеют формат, приведенный на рис. 5.12.

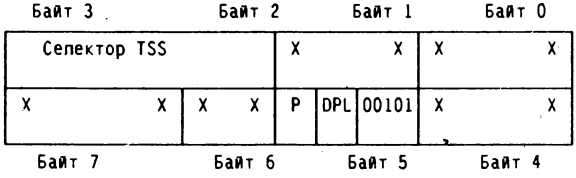


Рис.5.12. Формат дескриптора шлюза задачи

При использовании шлюза команда CALL должна задавать селектор с битом TI=0 для выбора в LDT необуходимого селектора TSS. Обращение к шлюзу разрешается, если для текущей программы значение CPL $\leqslant$ DPL для дескриптора шлюза. Селектор TSS из дескриптора шлюза должен иметь TI=1. Селектор загружается в регистр TR и выбирает из GDT дескриптор TSS, если этот дескриптор имеет значение поля DPL $\leqslant$ CPL. При нарушении указанных правил привилегий возникает прерывание типа 10.

При использовании шлюзов задачи с меньшим уровнем привилегий могут вызывать более привилегированные (защищенные) задачи, поэтому в данном случае встает задача обеспечения сохранности содержимого стеков. Эта проблема решается путем использования трех отдельных начальных адресов стеков, которые задаются тремя парами полей TSS: SSO и ESPO, SSI и ESPI, SS2 и ESP2. Если дескриптор вызываемого TSS имеет поле DPL=0, то при вызове задачи через шлюз в регистры SS и ESP микропроцессора загружаются значения SSO и ESPO. При DPL=1 загружается SSI и ESPI, при DPL=2 загружается SS2 и ESP2. Если DPL=CPL=3, то создание отдельного стека не требуется, и вызванная задача продолжает заполнение и использование стека, созданного в процессе выполнения старой (вызывающей) задачи. Таким образом при каждом уровне защищенности (привилегий) задачи создают отдельный стек, что исключает возможность нарушения его содержимого задачами с более низким уровнем привилегий.

При переключении задач с помощью команды CALL содержимое TR заносится в два младших байта сегмента TSS в качестве селектора, обеспечивающего возврат к выпонению текущей задачи. При этом команда CALL устанавливает значение бита вложенной задачи NT=1 в регистре EFLAG. Таким образом последовательным использованием этой команды можно реализовать многократное *вложение задач*. При этом возврат из задачи осуществляется с помощью команды IRET, которая анализирует значение NT и при NT =1 осуществляет переключение на задачу, задаваемую селектором возврата в сегменте TSS текущей задачи. При NT=0 команда IRET осуществляет обычную процедуру возврата из программы с восстановлением из стека содержимого CS, EIP, EFLAGS.

Команда JUMP при переключении задач не сохраняет в TSS селектор возврата и устанавливает значение NT=0. Кроме того, команды JUMP и CALL по разному влияют на значение бита занятости. Команда JUMP при переключении устанавливает бит B=0 в дескрипторе старой задачи и бит B=1 в дескрипторе новой задачи. Команда CALL также устанавливает B=1 для новой задачи, но сохраняет B=1 и для предыдущей задачи. Таким образом каждая задача в цепи вызовов оказывается занятой, что запрещает применение рекурсивных процедур и реентерабельных программ.

5.3.3. Особенности реализации режима виртуального МП 8086

Как указано в разд. 1.3, МП 80386 может выполнять в защищенном режиме все программы, написанные для МП 8086, обеспечивая при этом ряд средств защиты и возможность страничной организации памяти. Такой вариант защищенного режима называется режимом *виртуального микропроцессора 8086* (сокращенно V86) и реализуется при установке в регистре EFLAGS значения признака VM=1. Таким образом в режиме V86 микропроцессор 80386 работает как виртуальный процессор, состоящий из аппаратных средств МП 80386, прикладного программного обеспечения МП 8086 и системного программного обеспечения (монитор V86).

В разд. 1.3 описаны правила формирования линейного адреса в режиме V86. В данном разделе рассмотрены только особенности режима V86, связанные с обеспечением защиты и страничной организацией памяти, а также способы его инициализации.

При работе в режиме V86 микропроцессор использует страничную организацию памяти, если бит 31 в регистре управления CR3 имеет значение PG=1. При этом 20-разрядный линейный адрес, формируемый в режиме V86, делится на 256 страниц. С помощью таблицы страниц, они могут быть размещены в любом месте адресного пространства МП 80386, имеющего объем 4 Гбайта. Каждая задача в режиме V86 может использовать свои варианты размещения страниц, так как при переключении задач загружается заново содержимое регистра управления CR3, определяющее базовый адрес каталога разделов. При использовании страничной организации памяти в режиме V86 реализуются соответствующие способы защиты страниц, описанные в разд. 5.1.

Все программы, выполненные в режиме V86, имеют низший уровень привилегий: CPL=3. Этим режим V86 отличается от реального режима, при котором всем программам предоставляется высший уровень привилегий: CPL=0. При формировании линейного адреса в режиме V86 не используются дескрипторы (разд. 1.3), установленные в разд. 5.2 правила привилегий в этом режиме не выполняются. В режиме V86

используются только отдельные способы защиты, рассмотренные ниже.

1. В режиме V86 не осуществляется зашита сегментов. Если используется страничная организация памяти, то ее защита при нарушении правил доступа к странице реализуется путем прерывания типа 14 (отказ страницы).

2. Поступление команд, которые выполняются в защищенном режиме только при уровне CPL=0, в режиме V86 вызывает прерывание типа 13 (нарушение защиты). Это команды LIDT, LGDT, LMSW, CLTS, HLT, а также MOV для регистров управления, тестирования и отладки.

3. Поступление команд, выполняемых только в защищенном режиме, в режиме V86 или реальном режиме вызывает прерывание типа 6 (неразрешенный код команды). Это команды LLDT, SLDT, LTR, STR, LAR, LSL, ARPL, VERR, VERW.

4. Команды PUSHF, POPF, CLI, STI, INT и IRET в режиме V86 чувствительны к значению поля IOPL в EFLAGS. Они выполняются только при IOPL=3, в противном случае возникает прерывание типа 13. Отметим при этом, что команды INTO, INT3, ROUND в режиме V86 (и вообще в защищенном режиме) выполняются независимо от значения поля IOPL.

5. Выполнение команд ввода-вывода IN, OUT, INS, OUTS в режиме V86 не зависит от значения поля IOPL (отличие от защищенного режима). Однако при вызове в режиме V86 задачи, имеющей БКВВ в сегменте TSS, данные команды учитывают значение битов разрешения для соответствующих портов. При обращении к порту, для которого в БКВВ бит разрешения установлен в 1, реализуется прерывание типа 13.

Вход МП 80386 в режим V86 можно осуществить в защищенном режиме двумя способами:

1) путем переключения на задачу МП 80386, которая имеет в TSS поле EFLAGS с установленным битом VM=1. При этом новая задача будет выполнять команды МП 8086 и 80336 и формировать базовые адреса, как МП 8086. Отметим, что переключение на задачу МП 8028 не может вызвать переход в режим V86, так как при этом из TSS загружается только 16 младших разрядов поля EFLAGS, в которые не входит бит VM;

2) путем выполнения команды IRET, имеющей высший уровень привилегий CPL=0, если загружаемое при этом из стека

содержимое регистра EFLAGS имеет установленный бит VM=1. При других уровнях привилегий команда IRET не будет изменять значения VM, т.е. переход в режим V86 не реализуется.

Выход МП 80386 из режима V86 может произойти только при обработке прерываний. При этом возможны два варианта.

1. В результате прерывания происходит переход на процедуру с высшим уровнем привилегий, т.е. устанавливается CPL=0. При этом текущее состояние регистра EFLAGS заносится в стек, а бит VM сбрасывается в нуль. Таким образом, вызванная процедура будет выполняться как программа МП 80386 в защищенном режиме. Если вызванная процедура имеет более низкий уровень привилегий (CPL > 0), то происходит прерывание типа 13 (нарушение защиты).

2. Прерывание вызывает переключение задачи. При этом в TSS старой задачи, выполнявшейся в режиме V86, заносится текущее содержание регистров, в том числе EFLAGS с установленным битом VM=1. Таким образом имеется возможность вернуться к выполнению этой задачи в режиме V86. Если загружаемый при переключении задач новый TSS является сегментом МП 80386 со значением бита VM=0 в поле EFLAGS. или сегментом МП 80286, то признак VM в режиме EFLAGS сбрасывается в нуль, и процессор будет выполнять новую задачу МП 80386 или 80286 в защищенном режиме.

Команда POPF не изменяет значение VM, даже если она выполняется в защищенном режиме с наивысшим уровнем привилегий (CPL=0). Поэтому эта команда не обеспечивает переход МП 80386 в режим V86. При выполнении команды PUSHF в стек загружается содержимое EFLAGS со сброшенным битом VM=0, даже если процессор работает в режиме V86.

Глава 6. Прерывания

6.1. Прерывания и исключения

Система прерываний МП 80386 имеет особенности, которые связаны с введением в его архитектуру механизмов обработки виртуальных адресов и защиты задач. Следует, поэтому, уточнить некоторые понятия механизма прерываний, ранее известные разработчикам микропроцессорной техники, и определить ряд новых.

Обобщенно прерываниями МП 80386 называют реакцию на особые случаи, которая заключается в передаче управления от текущей программы специальной процедуре обслуживания. К таким требующим обработки случаям относится появление сигналов от аппаратной компоненты МП системы о важных внешних событиях и сообщения внутренних блоков МП об ошибках или затруднениях при выполнении команд. Под прерываниями в узком смысле слова обычно понимают процесс обработки запросов первого типа, от внешних устройств, в то время как исключения позволяют обрабатывать запросы второго типа, от внутренних блоков. В тексте книги при изложении общих положений, справедливых во всех ситуациях, используется термин прерывания/исключения, а при рассмотрении особенностей обработки аппаратных и программных событий - термины прерывание и исключение.

Аппаратные прерывания являются следствием внешних, асинхронных относительно тактовой частоты системы, событий и делятся на два вида - маскируемые и немаскируемые. Прерывания обслуживаются после завершения выполнения текущей команды; по окончании процедуры обслуживания микропроцессор продолжает выполнять программу с команды, непосредственно следующей за прерванной.

Исключения делятся на *отказы* (faults), *ловушки* (traps) и *выходы из процесса* (aborts), в зависимости от способа сообщения о них и возможности перезапуска микропроцессора с вызвавшей их команды. Отказы - это исключения, которые выявляются и обслуживаются перед выполнением команды; они могут иметь место в виртуальной системе памяти, когда

процессор обращается к несуществующим странице или сегменту. В процессе обработки такого исключения операционная система обратится к странице или сегменту на диске, а МП 80386 перезапустит команду. Ловушка - это исключение, которое возникает непосредственно после выполнения команды. Примером ловушки является прерывание/исключение, обрабатываемое пользователем. Выход из процесса является исключением, которое не позволяет точно локализовать вызвавшую его команду. Выходы из процесса используются для сообщения о крупных ошибках, таких как ошибки аппаратуры и ошибки в системных таблицах. Адресом возврата из процедуры обработки исключений всегда является команда, вызвавшая исключение, возможны префиксы.

6.2. Обработка прерываний

МП 80386 способен обрабатывать до 256 различных типов прерываний/исключений. В табл. 6.1 приведены типы прерываний/исключений и указана возможность возврата. Из 256 возможных прерываний 32 зарезервированы, остальные предоставлены разработчику системы.

Для связи внешней аппаратуры подсистемы прерываний, вырабатывающей запросы, с процедурами обслуживания должна быть определена таблица переходов. Для МП 80386 она включает максимум 256 элементов, которые представляют собой указатели на соответствующие процедуры обслуживания. В реальном режиме элементы таблицы 4-байтные - величина, загружаемая в регистр CS, плюс 16-разрядный относительный адрес, загружаемый в регистр IP. В защищенном режиме элементами таблицы являются 8-байтные дескрипторы, а сама таблица называется таблицей дескрипторов прерываний IDT (разд. 5.1.1). Доступ к таблице осуществляется через регистр IDTR (рис. 1.7) с помощью команд LIDT и SIDT, которые загружают в него новое значение и сохраняют текущее.

С точки зрения формата элементы таблицы IDT являются системными дескрипторами типа шлюз. Шлюз осуществляет связь между программой-источником и программой-приемником при передаче управления и позволяет процессору автоматически проверять условия защиты задач. Одновременно использование

шлюзов дает возможность проектировщику контролировать точки входа в операционную систему.

Таблица 6.1. Перечень прерываний/исключений

Номер	Описание	Причина	Наличие адреса возврата	Тип исключения
0	Ошибка деления	Команды DIV,IDIV	Да	Отказ
1	Исключение для отладки	Любая команда	Да	Ловушка
2	Немаскируемое прерывание	Сигнал NMI, команда INT 2	Нет	
3	Однобайтная команда останова	INT 3	Нет	Ловушка
4	Переполнение	INT 0	Нет	Ловушка
5	Превышение границы массива	BOUND	Да	Отказ
6	Неразрешенный код команды	Любая команда	Да	Отказ
7	Отсутствует сопроцессор	ESC,WAIT	Да	Отказ
8	Двойная ошибка	Любая команда, вызывающая исключение		Выход
9	Выход за сегмент сопроцессора	ESC	Нет	Выход
10	Неразрешенный TSS	JMP,CALL,IRET,INT	Да	Отказ
11	Отсутствут сегмент	Команды, использующие сегментные регистры	Да	Отказ
12	Ошибка обращения к стеку	Команды, использующие стек	Да	Отказ
13	Общая защита	Любое обращение к памяти	Да	Отказ
14	Отказ страницы	Любое обращение к памяти или выборка команды	Да	Отказ
15	Зарезервировано			
16	Ошибка сопроцессора	ESC,WAIT	Да	Отказ
17-31	Зарезервировано			
32-255	Предоставлены пользователю	Внешние маскируемые прерывания	Нет	
0-255	Двухбайтные команды прерываний	INT n	Нет	Ловушка

В таблице IDT могут размещаться три сорта шлюзов: шлюз задачи, шлюз прерывания и шлюз ловушки. Шлюз задачи используется для переключение задач, а шлюзы прерывания и ловушки - для вызова соответствующих процедур обслуживания. Таким образом, запрос прерывания может обрабатываться задачей прерывания или процедурой прерывания. Обслуживание запроса с помощью задачи является стандартным способом для

внешних прерываний. Это связано с тем, что такие запросы, как правило, предназначаются операционной системе и не связаны с текущей прикладной программой. Преимуществом такого подхода является возможность реализации механизма вложенных прерываний, если прерывания в задаче разрешены. Вызов задачи прерывания осуществляется и при обработке исключений, например исключения 10 "Неразрешенный сегмент TSS", когда поврежденная задача не может вызвать процедуру прерывания. Недостатком этого приема является то, что переключение задач выполняется медленнее вызова процедуры, поэтому если приоритет запроса высок, а программа обслуживания короткая, ее оформляют в виде процедуры. При вызове процедуры обслуживания через шлюз прерывания сбрасывается флажок IF, что запрещает дальнейшие прерывания. Это полностью соответствует вызову процедуры в реальном режиме, поэтому шлюзы прерываний часто используются для обеспечения совместимости. При вызове процедуры через шлюз ловушки флажок IF не сбрасывается. Такой прием передачи управления чаще реализуют при обработке исключений, поскольку на период обслуживания нежелательно выключать механизм разделения времени, использующий прерывания таймера.

Если запрос прерывания/исключения принят на обслуживание, выполняются следующие действия. Во-первых, текущий адрес программы и флаги сохраняются в стеке, что позволяет возвратиться в прерванную программу. Далее в МП 80386 вводится 8-разрядный код (вектор), который указывает точку входа в таблицу прерываний. Затем выполняется подпрограмма обслуживания, и по ее завершении, когда выполняется команда IRET, старое состояние процессора восстанавливается, программа продолжает выполняться с соответствующей команды.

Восьмиразрядный код прерывания вводится в МП 80386 различными способами: исключение вводит его изнутри, команды INT в программе содержат или предполагают этот вектор, аппаратное маскируемое прерывание вводит его посредством цикла магистрали типа подтверждение прерывания, немаскируемому аппаратному прерыванию назначен код 2.

6.3. Прерывания с точки зрения пользователя

Разработчик системы на основе МП 80386 имеет возможность использовать прерывания трех видов: аппаратные маскируемые, аппаратные немаскируемые и программные.

Маскируемые прерывания (INTR) являются наиболее общим методом обработки асинхронных внешних событий в аппаратуре. Прерывание этого типа имеет место, когда сигнал на входе INTR становится активным (высокий уровень), а бит флага прерывания (IF) имеет значение разрешения. Микропроцессор откликается на запрос прерывания только между командами (строковые команды REPeat имеют "окно прерывания", которое позволяет обслуживать прерывания в ходе передачи длинной строки). Если прерывания разрешены, процессор выполняет магистральные циклы подтверждения прерывания и во время второго из них читает 8-разрядный код, формируемый на шине данных аппаратурой подсистемы прерываний (обычно контроллером 8259A). Этот код указывает один из 224 источников прерываний, определенных разработчиком при формировании таблицы прерываний и процедур обслуживания. Бит IF в регистре EFLAG при принятии запроса сбрасывается, что запрещает обслуживание других запросов в ходе выполнения процедуры обслуживания. Однако этот бит может быть установлен принудительно в процедуре обслуживания, чтобы обеспечить вложенные прерывания. При выполнении в конце процедуры команды IRET первоначальное состояние флага IF восстанавливается.

Немаскируемые прерывания (NMI) используются для обслуживания запросов с очень высоким уровнем приоритета. Типичным примером такого запроса является вызов процедуры обработки сбоя по питанию. При высоком уровне сигнала на входе NMI выполняется прерывание с внутренним кодом 2. В отличие от маскируемых прерываний для NMI циклы подтверждения не выполняются. В ходе процедуры обслуживания NMI МП 80386 не воспринимает другие запросы немаскируемых и маскируемых прерываний, пока не будет выполнена команда возврата IRET или процессор не будет сброшен. Если сигнал NMI возникнет при обработки другого запроса NMI, его присутствие будет зафиксировано для обслуживания после первой

же команды IRET. Бит IF сбрасывается в начале прерывания NMI для блокировки последующих маскируемых прерываний.

Третьим видом прерываний МП 80386 являются программные прерывания. Команда INT n вызывает выполнение процедуры обслуживания, адресуемой через таблицу переходов вектором n. МП 80386 обрабатывает программные прерывания как исключения.

Особым случаем двухбайтных команд прерывания INT n является однобайтная INT 3, прерывание в точке останова. Включением этой однобайтной команды в программу разработчик может установить точку останова как средство отладки.

6.4. Приоритеты прерываний и исключений

Прерывания являются реакцией на внешние события, поэтому сигналы запросов по входам NMI и INTR проверяются и обрабатываются между выполнением соседних команд программы. Немаскируемое прерывание NMI имеет более высокий приоритет, т.е. при одновременном выявлении запросов по обоим этим входам процедура его обслуживания выполняется первой. По ее завершении, если маскируемое прерывание остается разрешенным, МП 80386 вызывает процедуру его обработки.

Запросы на обработку исключений генерируются внутренними блоками МП 80386, когда при выполнении команды возникают какие-либо проблемы. Архитектура МП 80386 позволяет перезапустить любую команду, вызвавшую исключение, после того как это исключение будет обработано соответствующей процедурой. Одна и таже команда может служить причиной нескольких исключений, однако при каждой попытке выполнить эту команду будет возникать только одно из них. Процедура обработки каждого исключения исправляет свою ситуацию и осуществляет перезапуск, таким образом исключения обрабатываются до тех пор, пока не произойдет успешное завершение команды.

При дешифрации и выполнении каждой команды МП 80386 параллельно осуществляет последовательность проверок на возникновение прерываний/исключений. Эта последовательность с

момента завершения текущей команды и до завершения следующей такова:

1) проверяются ловушки исключения 1 только что завершенной команды (пошагового режима - через установленный флаг ловушки, прерывания по данным - установкой бита в регистрах отладки);

2) проверяются отказы исключения 1 следующей команды (установка бита прерывания выполнения команды в регистрах отладки);

3) проверяются внешние сигналы на входах NMI и INTR

4) проверяются отказы сегментации, мешающие выборке следующей команды (исключения 11 и 13)

5) проверяются отказы страничной организации, мешающие выборке следующей команды (исключение 14);

6) проверяются отказы дешифрации следующей команды (исключение 6 при недопустимом коде операции; исключение 6, если в реальном режиме или виртуальном режиме 8086 делается попытка выполнить команду, относящуюся исключительно к защищенному режиму; исключение 13, если команда длинее 15байт или имеет место нарушение уровня привилегий в защищенном режиме);

7) если идентифицирован код команды WAIT, проверяются условия TS=1 и MP=1 (исключение 7, если оба бита равны 1);

8) если идентифицирован код ESCAPE команды арифметического сопроцессора, проверяются условия EM=1 и TS=1 (исключение 7, если один из бит равен 1);

9) если идентифицирован код команды WAIT или код ESCAPE, проверяется уровень сигнала на входе ERROR# (исключение 16, если этот сигнал активен);

10) при каждом обращении к памяти, затребованном командой, проверяются:

- -ошибки сегментации, мешающие передаче полной единицы памяти (исключения 11,12,13);

-ошибки страничной организации, мешающие передаче полной единицы памяти (исключение 14);

Видно, что эта последовательность поддерживает концепцию "сегменты над страницами".

6.5. Перезапуск команд

МП 80386 поддерживает перезапуск всех команд после отказа. Когда исключение вызвано выполнением команды (исключения, перечисленные в пп.4 - 10 предыдущего раздела), МП 80386 вызывает соответствующую процедуру обслуживания, которая исправляет ситуацию и передает управление на адрес возврата. Перезапуск команд гарантируется в любых ситуациях, за исключением двух случаев:

-команда вызывает переключение задач и в новой задаче TSS частично "отсутствует" (при полностью "отсутствующем" сегменте TSS перезапуск возможен). Этого можно избежать, если сохранять копию TSS такой задачи, либо выравнивать сегменты TSS, чтобы они полность помещались в страницу размером 4Кбайта (для сегментов TSS 4Кбайта и менее)

-операнд сопроцессора располагается у верхушки сегмента размером 64Кбайта или 4Гбайта и находится в трех страницах, причем средняя из них "отсутствует". Этого можно избежать, стартуя с границы страниц в сегментах, содержащих операнды сопроцессора (если сегменты имеют размер 64-200Кбайт и более).

Очевидно, что трудностей перезапуска можно избежать при соответствующем проектировании операционной системы.

6.6. Двойные ошибки

Двойная ошибка (исключение 8) имеет место, когда при вызове процедур обслуживания иключений, связанных с ошибками сегментов (разд. 6.2 исключения 10 - 13), МП 80386 выявляет новое исключение, отличное от исключения 14 ("Ошибка страницы"). Двойная ошибка также имеет место, когда при вызове процедуры обработки исключения 14 выявляется новое исключение, отличное по типу от повторной ошибки страницы. Повторная ошибка страницы не является двойной ошибкой, МП 80386 повторно вызовет процедуру обработки исключения 14, однако дальнейшие подобные ошибки вызовут отключение МП.

Глава 7. Сброс и инициализа́ция, тестирование, средства поддержки отладки

7.1. Сброс и инициализация

Сброс МП 80386 может быть выполнен посредством активизации сигнала на входе RESET, причем длительность импульса не должна быть менее 15 периодов CLK2. Если по спадающему фронту этого сигнала будет запрошено самотестирование, его длительность должна составлять по меньшей мере 78 периодов CLK2, в противном случае может быть зафиксирована неполадка, которой на самом деле нет. После спадающего фронта сигнала RESET (и после самотестирования, если оно было затребовано) МП 80386 в течение приблизительно 350-450 периодов CLK2 выполняет последовательность внутренней инициализации. В это время между 20-ым периодом CLK2 и первым циклом магистрали он проверяет вход ERROR# для того, чтобы определить наличие сопроцессора 80387.

Сигнал на входе RESET имеет наивысший приоритет среди сигналов управления процессором, при активизации он прерывает любые внутренние действия процессора и выполняемый цикл магистрали на любой стадии. После сброса и инициализации выводы микропроцессора устанавливаются в следующее состояние:

D0-D31	Высокоомное (третье) состояние
A2-A31	Высокий уровень
B0#-B3#	Низкий уровень
W/R#, M/IO#, HLDA	Низкий уровень
ADS#,DC#,LOCK#	Высокий уровень

В регистры заносится информация, указанная ниже.

EFLAGS - старшие 14 бит (31-18) и биты 15,5,3 неопределены. Значащие биты сбрасываются в 0 (кроме бита 1, который всегда установлен в 1).

CR0 - биты 30-5 неопределены. Значащие биты 31, 4-0 сбрасываются в 0.

EIP - содержит 0000FFF0H.

CS - содержит F000H (в регистре дескриптора базовый адрес равен FFFF0000H, а граница сегмента равна 0FFFFH).

DS - 0000H.

SS - 0000H.

ES - 0000H.

FS - 0000H.

GS - 0000H.

DX - содержит идентификатор изделия в поле DH (код 3, обозначающий МП 80386) и номер модификации в поле DL.

EAX - содержит результаты самотестирования, если оно было затребовано.

Содержимое остальных регистров микропроцессора неопределено.

После сброса и инициализации МП 80386 начинает работать в режиме реальных адресов при запрещенных прерываниях. Выполнение команд начинается с верха физической памяти - ячейки FFFFFFF0H. Такой подход позволяет выполнить инициализацию системы на основе МП 80386 с помощью подпрограммы, располагающейся в старших адресах младшего мегабайта физической памяти и зашитой в ПЗУ. Переход на другие адреса старших 64Кбайт может быть осуществлен с помощью команд внутрисегментной передачи управления JMP или CALL. Первая же команда межсегментной передачи управления снимает высокий уровень с выводов адреса A20-31.

Имея наивысший приоритет, сигнал RESET переводит МП 80386 в состояние сброса даже из состояния подтверждения захвата Th при активизированном сигнале HOLD. Если сигнал HOLD остается активным после завершения микропроцессором последовательности инициализации, то МП 80386, не начиная первого цикла магистрали, переходит в состояние подтверждения захвата. Если сигнал HOLD оставался установленным при сбросе сигнала RESET, то имеет место обычная проверка активности сигналов на входах BUSY# и ERROR#. Первый проверяется на предмет необходимости самотестирования, второй - для идентификации сопроцессора 80387.

7.2. Возможности тестирования

7.2.1. Самотестирование

МП 80386 обладает возможностью самотестирования. В этом режиме проверяются недоступные при внешнем обмене фрагменты структуры БИС (прежде всего управляющее ПЗУ), определяются идентификатор изделия и номер модификации. Результаты самотестирования заносятся в регистры EAX и EDX. Самотестирование инициируется, когда на входе RESET после сброса осуществляется переход из единицы в нуль, а на входе BUSY# сигнал имеет активный низкий уровень. Процесс продолжается примерно 2^{19} периодов CLK2, что составляет около 26мс при тактовой частоте 20МГц. При нормальном завершении процесса в регистре EAX находятся нули, в противном случае микросхема некондиционна. В регистр EDX заносятся идентификатор изделия и номер модификации.

7.2.2. Тестирование TLB

МП 80386 имеет механизм тестирования буфера ассоциативной трансляции TLB. Это свойство полезно прежде всего для тех, кто намерен самостоятельно создавать тестирующие программы для МП 80386. Проверка TLB требует наличия тестера и ассемблерной программы управления тестовыми последовательностями. Управление страницами при проверке TLB должно быть отключено. Метод тестирования TLB уникален для МП 80386 и не может применяться для других микропроцессоров.

Тестирование TLB осуществляется при помощи двух операций: запись в ячейки TLB и просмотр содержимого TLB (поиск). Эти операции поддерживаются двумя регистрами - TR6 является регистром команд тестирования, а TR7 регистром данных (рис.). Функции битовых полей и отдельных битов регистров тестирования следующие:

C: бит управления. При значении C=0 осуществляется запись в TLB, а при C=1 поиск.

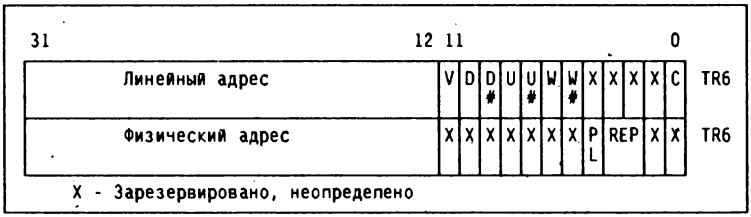


Рис. 7.1. Формат регистров TR6 и TR7

Линейный адрес: это поле тэга TLB. При записи в TLB для данного линейного адреса резервируется вход в TLB, а другие входы в TLB получают значения, зависящие от содержимого регистра TR7 и величины, только что занесенной в TR6. При просмотре TLB в буфере ищется эта величина. Если найдено одно и только одно совпадение, остальные поля регистров TR6 и TR7 устанавливаются в соответствии с этим входом.

Физический адрес: это поле данных TLB. При записи в TLB вход, зарезервированный для линейного адреса, указанного в регистре TR6, получает это значение. При просмотре TLB сюда считывается поле данных (физический адрес).

PL: При записи в TLB, если PL=1, то переключение на один из блоков ассоциативной памяти производится в соответствии со значением в поле REP регистра TR7, если PL=0, то переключение осуществляется внутренним указателем в блоке управления страницами. При просмотре TLB бит PL сообщает о результате поиска: в случае успеха PL=1, в противном случае PL=0.

V: бит достоверности для выбранного входа TLB. Достоверность означает успешную трансляцию адресов. Все биты достоверности могут быть очищены при операции записи в регистр управления CR3.

D,D#: бит мусора входа TLB и инверсный бит.

U,U#: бит "управление у пользователя" входа TLB и инверсный бит.

W,W#: бит записи для входа TLB и инверсный бит.

Для опций D, U, W имеются биты прямого и инверсного значения, что позволяет ввести "безразличное" состояние при поиске в TLB. Значение комбинаций в этих парах бит следующее:

X	X#	Действие при поиске в TLB	Значение бита X после записи в TLB
0	0	Неуспех всегда	Бит X становится неопределенным
0	1	Успех, если X=0	Бит X сбрасывается в 0
1	0	Успех, если X=1	Бит X устанавливается в 1
1	1	Успех всегда	Бит X становится неопределенным

Для записи по входу TLB необходимо:

1. Записать в регистр TR7 требуемый физический адрес, значения PL и REP.

2. Записать в регистр TR6 соответствующий линейный адрес и все необходимые биты (удостовериться, что C=0 для команд записи).

Для чтения по входу TLB необходимо:

1. Записать в регистр TR6 соответствующий линейный адрес (удостовериться, что C=0 для команд поиска).

2. Прочитать содержимое регистров TR7 и TR6. Если в регистре TR7 бит PL указывает на успех трансляции, то все другие величины показывают содержимое TLB. Если бит PL указывает на неуспех операции, то все остальные значения в регистрах TR7 и TR6 являются неопределенными.

7.3. Средства поддержки отладки

МП 80386 поддерживает три основных механизма, упрощающих отладку разрабатываемого программного обеспечения:

* специальный код (0CCH) останова выполнения программы;
* пошаговый режим, обеспечиваемый битом TF регистра флагов;
* точки останова по коду и данным, обеспечиваемые регистрами отладки DR0-DR3 и DR6,DR7.

Команда INT 3 длиной один байт, имеющая код 0CCH, обычно используется программными отладчиками и позволяет устанавливать точки безусловного останова в разрабатываемой программе. Отладчик должен вставлять эту команду во все контрольные точки программы. При выходе на точку останова команда генерирует исключение 3 и управление передается

соответствующей процедуре обработки. Однобайтная команда останова является частным случаем двухбайтной команды INT n при n=3, однако в отличие от последней, которая в защищенном режиме и режиме виртуального 8086 чуствительна к уровню защиты IOPL, однобайтная команда такой чуствительности не имеет.

Пошаговый режим обеспечивает останов после выполнения каждой команды программы. Он вводится установкой бита TF в регистре флагов EFLAG. В конце каждой команды, если бит TF установлен, имеет место исключение 1 (код исключения вводится изнутри). Точнее, исключение 1 имеет место как ловушка после выполнения команды, следующей за той командой, где обнаружен установленный флаг TF. Прерванная команда загружает в стек текущее содержимое регистра флагов (где TF установлен) и сбрасывает бит TF, разрешая нормальное выполнение процедуры обслуживания пошагового прерывания. После завершения процедуры команда IRET выгружает содержимое регистра флагов и передает управление следующей команде.

Регистры отладки (рис. 1.8) являются особенностью МП 80386. Они позволяют выполнять отладочные остановы программы как при выполнении кода команды, так и при обращении к данным. Поскольку состояние при остановах фиксируется во внутренних регистрах МП 80386, команда, на которую указывает контрольная точка, может располагаться и в ПЗУ и в области , которую делят несколько задач. В обоих этих случаях командой INT 3 отладочный останов не поддерживается. Шесть программно доступных регистров отладки позволяют установить четыре адреса останова, квалификаторы управления остановами, прочитать состояние МП в контрольных точках. После сброса контрольные точки не установлены. После программирования регистров отладки остановы в определяемых ими контрольных точках обрабатываются процедурой исключения 1.

Регистры линейного адреса точки останова DR0-DR3. Эти четыре регистра содержат 32-разрядные линейные адреса контрольных точек останова. В процессе выполнения программы МП 80386 непрерывно сравнивает линейные адреса в DR0-DR3 с текущим линейным адресом.

Регистр управления отладкой DR7. Назначение отдельных битов и битовых полей регистра DR7 (рис. 7.2) следующее:

LENi (поле, определяющее длину точки останова). Двухбитное поле LEN определяется для каждой из четырех точек останова. Оно указывает величину кадра соответствующей точки останова, т.е. количество байтов, внутри которых условия останова могут сработать. Для точки останова по доступу к данным длина может состовлять 1, 2, 3 и 4 байта. Для точек останова по

Рис. 7.2. Формат регистра DR7

выполняемой команде длина должна быть равна 1 байту (LEN=00). Значение кода в этом поле следующее:

00	длина один байт
01	длина два байта
10	длина неопределена
11	длина четыре байта

Код поля LEN указывает, все ли младшие биты линейного адреса участвуют в идентификации точки останова. Все точки останова должны быть выровнены; 2-байтовые точки останова должны быть выровнены по границам слов, 4-байтовые точки останова должны быть выровнены по границам двойных слов. На рис. 7.3 показано расположение поля контрольной точки останова длиной 1, 2 и 4 байта, линейный адрес которой равен 00000005H и занесен в регистр DR2.

RWi (поле, определяющее тип доступа к памяти). Двухбитовое поле RWi определяется для каждой из четырех точек останова.

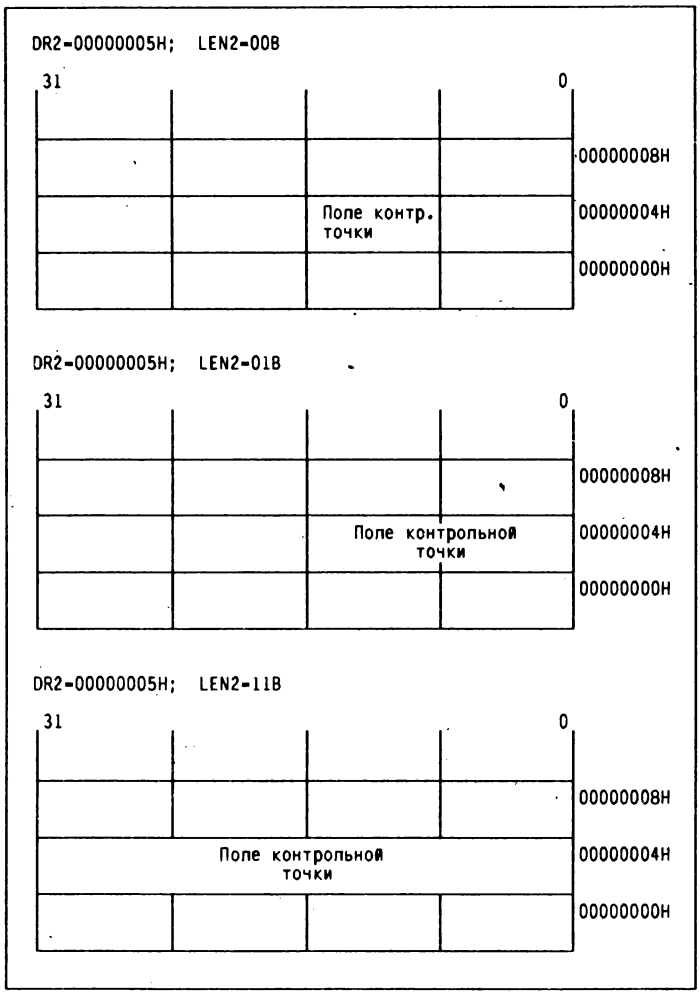


Рис. 7.3. Пример полей контрольных точек

Оно указывает тип обращения, который активизирует соответствующую точку прерывания. Значение кода в этом поле следующее:

00	выполнение команды
01	запись данных
10	значение неопределено
11	запись или чтение данных

Следует отметить, что запрос от контрольной точки, установленной на срабатывание от выборки команды, обрабатывается как отказ (до выполнения команды), а установленной на срабатывание по обращению к данным - как ловушка (после того как имела место передача данных).

Если при обращении к данным адрес полностью или частично попал в поле точки останова, условие контрольной точки выполнено и при наличии разрешения для этой точки будет иметь место исключение 1.

Линейный адрес контрольной точки, установленной на срабатывание от выборки команды, должен быть равен адресу первого байта (с учетом префиксов).

GD (бит идентификации обращений к регистрам отладки). Регистры отладки доступны только в реальном режиме или в защищенном режиме при уровне привилегий 0. Установка бита GD в реальном режиме и защищенном режиме на уровне привилегий 0 обеспечивает сверхзащиту всех обращений к регистрам отладки. При необходимости это гарантирует программному отладчику полный контроль над регистрами отладки. Когда бит GD установлен, попытка обратиться к любому из регистров отладки вызывает отказ исключения 1. При вызове процедуры обработки исключения 1 бит GD автоматически сбрасывается, что обеспечивает этой процедуре свободный доступ к регистрам отладки.

GE и LE (сообщения о глобальных и локальных точках останова по данным). Если какой-либо из этих битов установлен, сообщение о любой ловушке в контрольной точке по данным будет иметь место непосредственно после завершения команды, которая вызвала пересылку операнда. Если эти биты не установлены, сообщение об останове по данным может поступить спустя несколько команд или не поступить вообще. При определении контрольных точек по данным рекомендуется устанавливать и биты GE/LE. При выполнении

микропроцессором переключения задач бит LE очищается, чтобы предотвратить ненужные сообщения в новой задаче. Таким образом осуществляется быстрое переключение на новые условия сообщений об остановах в задачах, использующих локальное определение содержимого регистров отладки. Новое требование сообщений о точках останова должно быть осуществлено программно. На бит GE переключение задач не влияет. Он поддерживает сообщения о контрольных точках, которые действуют во всех задачах системы. Следует отметить, что сообщения об остановах по выборке команд всегда следуют немедленно, независимо от установки опций GE/LE.

Gi и Li (разрешение глобальных и локальных точек останова). Установка битов Gi/Li разрешает соответствующую контрольную точку (с линейным адресом, указанным в поле DRi, длиной - в LENi, критерием активизации - в RWi). Если бит Gi или Li установлен, то при обнаружении условий останова МП 80386 передает управление процедуре обработки исключения 1. При переключении задач биты Li сбрасывается, чтобы предотвратить ложное исключение. Таким образом осуществляется быстрое переключение на новые условия контрольных остановов в задачах, использующих локальное определение содержимого регистров отладки. Новое разрешение точек останова должно быть осуществлено программно. Все биты Gi от процесса переключения задач не зависят, поскольку поддерживают точки останова, действующие во всей выполняемой программной системе.

Регистр состояния DR6. Регистр состояния DR6 (рис. 7.4) позволяет процедуре обработки исключения 1 быстро определить причину вызова.

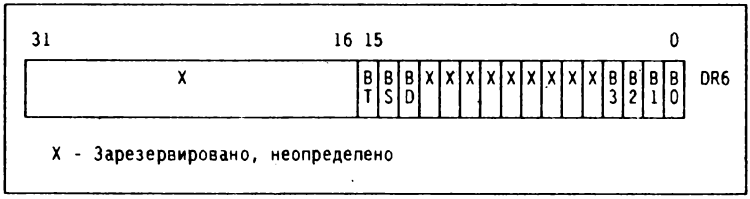


Рис. 7.4. Формат регистра DR6

Напомним, что исключение 1 возникает как результат следующих событий:

* отказ/ловушка в контрольной точке по одному из адресов, занесенных в регистры DR0-DR3;
* ловушка пошагового режима;
* ловушка переключения задач;
* отказ, возникший при попытке обращения к регистру отладки при установленном бите GD=1.

Биты регистра DR6 устанавливаются аппаратно, но аппаратно никогда не сбрасываются. Процедура обработки исключения 1 должна очищать регистр DR6 перед возвратом в программу пользователя, чтобы в дальнейшем можно было идентифицировать источник этого исключения. Назначение отдельных битов и битовых полей регистра DR6 следующее:

Bi (отказ/ловушка в контрольной точке 0-3). Четыре бита индикации останова соответствуют четырем регистрам DR0-DR3. Бит Bi устанавливается при выполнении условий, определенных полями DRi, LENi, RWi. При обнаружении условий останова и установленных битах Gi/Li процессор вызывает процедуру обработки исключения 1. Это исключение обрабатывается как отказ, если сработала контрольная точка, установленная на выборку команды, и как ловушка, если произошел останов по выборке данных.

BD (отказ при попытке обращения к регистрам отладки при установленном бите GD). Этот бит устанавливается, если имело место исключение при обращении к регистрам отладки при установленном бите защиты GD. При вызове процедуры обслуживания исключения бит GD автоматически сбрасывается, позволяя ей обратиться к регистрам отладки.

BS (ловушка пошагового режима). Этот бит устанавливается, если исключение 1 имело место по причине установленного бита TF в регистре EFLAGS.

BT (ловушка переключения задач). Этот бит устанавливается, если исключение 1 имело место из-за переключения на задачу, в которой установлен бит T сегмента TSS. Переключение проходит нормально, но перед выполнением первой команды новой задачи имеет место исключение 1.

Бит **RF** регистра EFLAGS может подавить обработку прерывания/исключения отладки, если процедура обработки

исключения 1 возвращает управление программе пользователя на адрес, который тоже является адресом контрольной точки

Литература

1.Уокерли Дж. Архитектура и программирование микроЭВМ:Пер. с англ.-М.:Мир,1984.-кн.2.341с.

2.Микропроцессоры./Под ред. Л.Н.Преснухина.-М.:Высшая школа, 1986.-кн.1.495с.

3.Каган Б.М., Сташин В.В. Основы проектирования микропроцессорных устройств автоматики.-М.:Энергоатомиздат, 1987.-304с.

4.Лю Ю.-Ч., Гибсон Г. Микропроцессоры семейства 8086/8088. Архитектура, программирование и проектирование микропроцессорных систем:Пер. с англ.-М.:Радио и связь, 1987.- 512с.

5.Щелкунов Н.Н., Дианов А.П. Микропроцессорные средства и системы.-М.:Радио и связь. 1989.,-288с.

6.Микропроцессоры и микропроцессорные комплекты интегральных микросхем:Справочник./Под ред. В.А.Шахнова.-М.:Радио и связь, 1988.-т.2.386с.

7.Проектирование микропроцессорной электронно-вычислительной аппаратуры:Справочник/В.Г.Артюхов, А.А.Будник, В.Д.Лапий и др.-К.:Техника, 1988.-263с.

8.Микропроцессорный комплект К1810. Структура, программирования, применение.:Справочная книга/Под ред. Ю.М.Казаринова.-М.:Высшая школа, 1990.-269с.

9.Морс С.П., Алберт Д.Д. Архитектура микропроцессора 80286:Пер. с англ.-М.:Радио и связь,1990.-304с.

10.Брамм П., Брамм Д. Микропроцессор 80386 и его программирование:Пер. с англ.-М.:Мир, 1990.-448с.

11.386TMDX Microprocessor.Intel Corporation, 1989.

12.386TMSX Microprocessor.Intel Corporation, 1990.

Оглавление

Вниманию разработчиков и производителей цифровых систем

Лаборатория "Микропроцессорные системы" МИФИ предлагает контрольно-диагностическую аппаратуру и ПО для микропроцессорных и других цифровых систем:

* схемные эмуляторы ОЭВМ К1816ВЕ48,49,35;
* схемные эмуляторы ОЭВМ К1816ВЕ51,31;
* схемный эмулятор МП К1816ВМ86,88;
* схемный эмулятор МП К1821ВМ85;
* схемный эмулятор МП Z80;
* программатор БИС ПЗУ, ППЗУ, ПЛМ, ОЭВМ;
* эмулирующий тестер цифровых систем;
* программируемый логический анализатор/генератор.
* программно-логические модели (отладчики) ОЭВМ и МП;
* интегрированную систему "Паскаль-51" для ОЭВМ К1816ВЕ51,31;
* интегрированную систему программирования ОЭВМ К1816ВЕ51,31 на языке Типизированного Ассемблера;
* транспьютерные платы-ускорители (10-40 MIPS, 2-8 Mb); системное ПО, инструментальные средства и библиотеки для них.

Все приборы работают с IBM PC. Мы сопровождаем их новыми версиями ПО, осуществляем гарантийное обслуживание.

Разрабатываем и изготавливаем микропроцессорные контроллеры и системы (возможно применение матричных БИС и полиимидных плат) по заказам организаций.

Научный руководитель лаборатории проф. Шагурин И.И.
Зав. лабораторией Бродин Б.В.

Москва, Каширское шоссе, 31, к.27. Тел. 324-91-55