Введение в микропроцессор 80386

Аннотация

Данный документ является введением в микропроцессор 80386 и предназначен для программистов, разработчиков аппаратуры и системного программного обеспечения.

С о д е р ж а н и е

1. Основные характеристики

Микропроцессор 80386 является высокопроизводительным 32-битным процессором, предназначенным для построения наиболее совершенных вычислительных систем сегодняшнего и завтрашнего дня. Станции САПР, графические системы с высокой разрешающей способностью, издательское дело, автоматизация контор и производства - вот те области, где сегодня может быть применен 80386. Применения завтрашнего дня скорее будут ограничены воображением разработчиков систем, чем вычислительной мощностью и возможностями 80386.

Микропроцессор 80386 дает разработчику систем большое число новых и эффективных возможностей, включая производительность от 3 до 4 миллион операций в секунду, полную 32-битную архитектуру, 4 гигабитное (2 байт) физическое адресное пространство и внутреннее обеспечение работы со страничной виртуальной памятью.

Несмотря на введение в него последних достижений микропроцессорной техники, 80386 сохраняет совместимость по об'ектному коду с программным обеспечением, в большом количестве написанным для его предшественников, 8086 и 80286. Особый интерес представляет такое свойство 80386, как виртуальная машина, которое позволяет 80386 переключаться в выполнении программ, управляемых различными операционными системами, например, UNIX и MS-DOS. Это свойство позволяет производителям оригинальных систем непосредственно вводить прикладное программное обеспечение для 16-битных машин в системе на базе 32-битных микропроцессоров.

Объединяя в себе производительность супермини ЭВМ и низкую стоимость и функциональную гибкость микропроцессора, 80386 может открыть новые рынки для микропроцессорных систем. Применения, недопустимые прежде из-за невысокого быстродействия микропроцессоров или не экономности использования супермини ЭВМ, стали теперь практически осуществимы благодаря 80386. Такие новейшие применения, как машинное зрение, распознавание речи, интеллектуальные работы и экспертные системы, бывшие до недавнего времени в основном на стадии эксперемента, теперь могут быть предложены на рынке.

Для тго, чтобы удовлетворить требованиям будущих применений, мало иметь 32-битные регистры, команды и шины. Эти основные свойства являются лишь отправной точкой для 80386. В нижеследующих разделах в общих чертах будет рассмотрена 32-битная архитектура 80386, обладающая такими новыми дополнительными свойствами, как:

- высокопроизводительная технология,
- обеспечение работы с виртуальной памятью,
- механизмы защиты,
- расширенное отладочное обеспечение,
- совместимость по об'ектному коду с 8086/80286.

1.1. 32-битная архитектура

32-битная архитектура 80386 обеспечивает программные ресурсы, необходимые для поддержки "больших " систем, характеризуемых операциями с большими числами, большими структурами данных, большими программами (или большим числом программ) и т.п. Физическое адресное пространство 80386 состоит из 2 байт или 4 гбайт; его логическое адресное пространство состоит из 2 байт или 64 терабайт (тбайт). Восемь 32-битных общих регистров 80386 могут быть взаимозаменяемо использованы как операнды команд и как переменные различных способов адресации. Типы данных включают в себя 8-, 16- или 32-битные целые и порядковые, упакованные и неупакованные десятичные, указатели, строки бит, байтов, слов и двойных слов. Микропроцессор 80386 имеет полную систему команд для операций над этими типами данных, а также для управления выполнением программ. Способы адресации 80386 обеспечивают эффективный доступ к элементам стандартных структур данных: массивов, записей, массивов записей и записей, содержащих массивы.

1.2. Высокопроизводительная технология

32-битная архитектура не гарантирует высокой производительности. Реализация потенциала архитектуры требует новейшей микроэлектронной технологии, точного разделения функций и внимания к внешним операциям кристалла, в особенности к взаимодействию процессора с памятью. Включение этих свойств обеспечивает 80386 самую высокую произвидительность по сравнению с любым другим существующим микропроцессором.

Микропроцессор 80386 реализован с помощью технологии фирмы ИНТЕЛ CH MOSIII - технологического процесса, объединяющего в себе возможности высокого быстродействия технологии HMOS с малым потреблением технологии кмоп. Использование геометрии 1,5 мкм и слоев металлизации дает 80386 более 275000 транзисторов на кристаллле. Сейчас выпускаются оба варианта 80386, работающих на частоте I2 и I6 мгц без состояний ожидания, причем вариант 80386 на 16 мгц обеспечивает скорость работы 3-4 миллиона операций в секунду.

Микропроцессор 80386 разделен внутри на 6 автономно и параллельно работающих блоков с соответствующей синхронизацией. Все внутренние шины, соединяющие эти блоки, имеют разрядность 32 бит. Конвейерная организация функциональных блоков в 80386 допускает временное наложение выполнения различных стадий команды и позволяет одновременно выполнять несколько операций. Кроме конвейерной обработки всех команд, в 80386 выполнение ряда важных операций осуществляется специальными аппаратными узлами. Блок умножения/деления 80386 может выпонять 32-битное умножение за 9-41 такт синхронизации, в зависимости от числа значащих цифр; он может разделить 32-битные операнды за 38 тактов (в случае чисел без знаков) или за 43 такта (в случае чисел со знаками). Регистр группового сдвига 80386 может за один такт сдвигать от 1 до 64 бит.

Во многих 32-битных применениях, в таких как, например, перепрограммируемые ЭВМ коллективного пользования, требуется преобразование логических адресов в физические и защита памяти с помощью блока управления памятью, БУП. В других применениях, например, в системах управления в реальном времени, это не требуется. Для большинства микропроцессорных систем с 32-битной архитектурой такое разделение функций реализуется путем использования дополнительного корпуса блока управления памятью. В отличие от них буп 80386 входит в состав процессора как один из двух функциональных блоков конвейерной струк-

туры. Операционная система, управляющая работой буп, позволяет, например, системе реального времени обходить страничное преобразование. Введение управления памятью внутрь кристалла дает повышенную производительность в системах, использующих буп и не приводит к ее снижению в тех систмах, которые БУП не используют. Такие характеристики стали возможны благодаря снижению задержек распространения, использованию внутреннего полупериодного тактирования и параллельной работы.

Еще одно свойство, необходимое в одних применениях и не требующееся в других, это обработка больших чисел, в особенности в арифметических операциях с плавающей запятой с одинарной и двойной точностью. Операнды с плавающей запятой имеют большую длину, а необходимый набор команд для операций над ними является довольно сложным; для реализации стандартного набора операций с плавающей запятой в соответствии со стандартом IEEE754 требуется несколько тысяч транзисторов. В этих целях в 80386 имеется аппаратное обеспечение совместной работы с отдельным математическим сопроцессором. К 80386 может быть подключен математический сопроцессор либо 80287, либо более производительный 80387. Для прикладного программного обеспечения сопроцессоры прозрачны; они лишь расширяют архитектуру 80386 с помощью регистров, типов данных и операций, требуемых стандартом IEEE754. Комбинация 80386 и 80387 может исполнять 1,8 миллион операций.

32-битный процессор, работающий с частотой 16 мгц, имеет большее быстродействие, чем большинство быстродействующих памятей, вследствии чего его производительность может быть ограничена временами доступа к памяти. 80386 был спроектирован так, чтобы с максимальной эффективностью использовать как наиболее быстродействующие статистические ОЗУ, так и недорогие динамические ОЗУ. Для обращения к быстрой памяти, например типа кэш, 80386 вырабатывает двухтактный магистральный цикл для адреса/данных. (Памяти типа кэш 80386 могут иметь любой объем от минимального полезного 4 кбайт до максимального, охватывающего все физическое адресное пространство). Обращение к более медленной памяти (или к устройствам ввода/вывода) может производиться с использованием конвейерного формирования адреса для увеличения времени установки данных после адреса до 3 тактов при сохранении двухтактных циклов в процессоре. Вследствие внутреннего конвейерного формирования адреса при исполнении команды, 80386, как правило, вычисляет адрес и определяет следующий магистральный цикл во время текущего магистрального цикла. Узел конвейерного формирования адреса передает эту опережающую информацию в подсистему памяти, позволяя, тем самым, одному банку памяти дешифрировать следующий магистральный цикл, в то время как другой банк реагирует на текущий магистральный цикл.

1.3. Обеспечение работы с виртуальной памятью

Виртуальная память позволяет ставить максимальный объем программы или группы программ в зависимость от имеющегося адресного пространства на диске, а не от об'ема физической памяти (ОЗУ), которая в настоящее время приблизительно в 400 раз дороже. Из вытекающей отсюда гибкости выигрывают изготовители оборудования (которые могут поставлять изделия, отличающиеся лишь в конфигурациях памяти и в уровне производительности), программисты (которые могут предоставлять управление хранением программ операционным системам и избегать написания программ с перекрывающимися структурами) и конечные пользователи (которые могут вводить новые и большие по объему

прикладные программы, не опасаясь нехватки памяти).

Виртуальная память реализуется операционной системой с соответствующей аппаратурной поддержкой. Микропроцессор 80386 обеспечивает работу с системами виртуальной памяти с сегментной или страничной организацией. Сегментная виртуальная память больше подходит для небольших 16-битных систем, в которых об'ем сегмента не превышает 64 кбайт. 80386 обеспечивает работу с сегментами объемом до 4 гбайт; поэтому в большинстве больших систем на базе 80386 системы виртуальной памяти будут использовать возможность страничного запроса. Для каждой страницы 80386 вырабатывает биты присутствия, занятости или регистрации обращения, которые необходимы для эффективной реализации виртуальной памяти со страничными запросами. В случае обращения к несуществующей странице 80386 автоматически делает переход к операционной системе, если операционная система считала с диска отсутствующую страницу, 80386 выполняет команду повторно. Высокая производительность в работе с виртуальной памятью обеспечивается в 80386 использованием внутренней кэш-памяти для хранения страничной информации. Эта кэш-память (называемая буфером просмотра трансляции, TLB) содержит информацию о распределении адресов 32 страниц, использовавшихся последними. Страницы виртуальной памяти 80386 имеют об'ем 4 кбайт, храня одновременно распределение 128 кбайт памяти, буфер TLB позволяет 80386 преобразовать адреса внутри кристалла, не обращаясь к хранящейся в памяти таблице страниц. В типичных системах 98-99% поиска адресов будет осуществляться через буфер TLB.

1.4. Механизмы защиты

Выполняя 3-4 миллиона операций в секунду, 80386 имеет достаточно вычислительной мощности для обеспечения самых сложных систем, состоящих из сотен или тысяч программных модулей. В таких системах вопрос заключается не в том, будут ли ошибки, а в том как их накти и повозможности быстро устранить и насколько их действие может быть ограничено. Такие системы могут быть быстро отлажены и сделаны более надежными при серийном освоении, если процессор будет проверять каждую команду по критерию защиты. При этом степень и тип используемой защиты зависит от конкретного применения. Обычно простые системы реального времени работают достаточно хорошо без использования защиты. Различные требования к защите могут быть наиболее полно удовлетворены с помощью набора выборочно используемых функций защиты, введенных в 80386:

- разделение адресных пространств задач;
- введение 0-4 уровней привилегий ;
- использование привилегированных команд (например,HALT);
- разделение сегментов по типам (например, кодовый сегмент или сегмент данных);
- введение прав доступа к сегментам и страницам (например, право только чтения или только исполнения);
- проверка границ сегмента.

Для сохранения максимельной производительности все проверки защиты в 80386 выполняются одновременно с выполнением команды.

1.5. Расширенные возможности отладки

Четыре внутренних отладочных регистра 80386 помогают значительно сократить время отладки программы. Эти регистры работают независимо от системы защиты и поэтому могут быть

использованы в любых системах, включая те, которые будут работать без защиты. Не менее важно и то, что они дают возможность устанавливать контрольные точки данных, помимо контрольных точек команд. 80386 отслеживает все четыре текущих адресных контрольных точки одновременно, не снижая скорости выполнения программы.

Контрольные точки команд вызывают переход (обычно в программу-отладчик) при выполнении команды, в большинстве процессоров это осуществляется с помощью специальной команды, которую отладчик записывает после команды, представляющей интерес. Задавая адреса контрольных точек в регистрах, 80386 устраняет программные искажения, неизбежные при внесении команд перехода в защищенную или общую часть программы. Контрольные точки данных, наличие которых является, для микропроцессора свойством уникальным, для целей отладки особенно полезны. По контрольной точке данных можно установить момент чтения адреса или же момент его записи или чтения. Используя контрольные точки данных, программист может, например, быстро установить команду, ответственную за ошибочную запись в структуре данных.

Кроме регистров контрольных точек, 80386 имеет и более традиционные отладочные функции в виде контрольных точек команд и пошагового исполнения программы.

1.6. Совместимость с микропроцессорами 8086/80286

Два поколения процессоров семейства 86 предшествуют процессору 80386 - 80286 и 8086, с каждым из них 80386 совместим на уровне двоичных кодов. Благодаря такой совместимости экономятся программные затраты, обеспечивается быстрый выход на рынок и доступ к обширной библиотеке программного обеспечения, написанного для машин на базе микропроцессоров семейства 86.

Микропроцессор 80386, конечно, может выполнять программы для 8086, он также может одновременно выполнять программы для 80286 и 80386. Однако наиболее важным свойством совместимости 80386 представляется свойство, называемое VIRTUAL 86 (виртуальный 86), устанавливающее защищенную структуру для 8086 внутри системы задач 80386. Дополняя свойство виртуального 8086 страничной организацией памяти, 80386 может закрепить за каждой задачей виртуального 8086 1 мбайтное адресное пространство в любой области физического адресного пространства 80386. Более того, если операционная система 80386 обеспечивает работу с виртуальной памятью, то задачи виртуального 8086 могут переноситься с диска и обратно как любые другие задачи. Таким образом, свойство виртуального 8086 позволяет 80386 одновременно выполнять программы, написанные для трех поколений семейства 86.

1.7. Заключение

Микропроцессор 80386 обеспечивает ту базовую производительность, которая необходима для построения высокопроизводительных микропроцессорных систем. Архитектура 80386 достаточно гибка: не ориентируясь на одно представление о вычислительной машине, она дает разработчикам систем воможность выбирать те варианты, которые наилучшим образом подходят для конкретного применения.

Полный набор свойств для управления памятью, включающий сегментацию, страничное разделение и обеспечение работы с виртуальной памятью, реализуется внутри кристалла. До четырех

уровней защиты может быть использовано для возведения границ между программными компонентами, однако защита может и не использоваться. Задачи виртуального 8086 могут обогатить 32-битные системы необычайно большим набором стандартных программ, уже разработанных для машин на базе 8086.

Производительность и гибкость микропроцессора 80386 могут быть дополены другими устройствами фирмы ИНТЕЛ и доведены до максимума. К этим устройствам относятся контроллеры локальных сетей, усовершенственные контроллеры прямого доступа к памяти, контроллеры дисков и графические сопроцессоры.

2. Прикладная архитектура

Микропроцессор 80386 дает разработчику прикладных программ на языке ассемблера или разработчику компилятора широкий набор 32-битных ресурсов. В данной главе эти ресурсы рассматриваются в трех разделах:

1) регистры;
2) память и логическая адресация;
3) типы данных и команды.

2.1. Регистры

Во всех вычислительных машинах, включая 80386, имеются регистры, которые программисты могут использовать для срочного промежуточного хранения. К данным, хранящимся в этих регистрах, можно обратиться без магистральных циклов, что сокращает время использования команды и предоставляет больше магистрального времени другим процессором, например, контроллерам прямого доступа к памяти. В 80386 имеется восемь программно доступных общих регистров, еще восемь регистров добавляется при подключении математического сопроцессора 80287 или 80387. Два других регистра 80386, предназначенных не для хранения данных, а для хранения статуса и управления процессором, также важны для программистов. Это регистр флагов и счетчик команд.

2.1.1. Общие регистры

Как видно из рис.2-1, Общие регистры 80386 имеют разрядность 32 бит, внутренние шины данных, внешние шины данных и адреса процессора также имеют разрядность 32 бит. В соответствии с любым обще принятым определением 80386 являются 32-битной машиной. Однако в соответствии с практикой других процессоров, предшественниками которых были 16-битные машины, принято, что в 80386 слово означает 16 бит, а 32-бита образуют двойное слово.

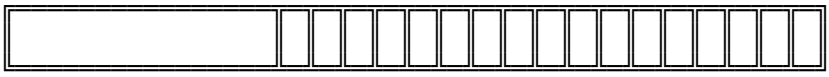
Как видно из рис.2-1 все общие регистры могут использоваться как 16 или 32-битные регистры, а четыре из них могут быть использованы и как 8-битные регистры. Почти во всех операциях любой общий регистр может быть определен как операнд. Любые два регистра, например, могут быть перемножены. Аналогичным образом, любой регистр при вычислении адреса может быть исползован в качестве базового или индексного. Поскольку в любой практической программе требуется стек, общий регистр ESP подразумевается как указатель вершины стека.

2.1.2. Флаги и счетчик команд

На рис.2-2 показан формат регистра флагов 80386. Флаги

делятся на три класса: статусные, управляющие и системные. Процессор устанавливает статусные флаги после многих команд, чтобы отразить результат операции. Например, если два операнда при сравнении оказываются равными, то процессор устанавливает флаг нулевого результата. Другие команды, преимущественно команды условного перехода, проверяют флаг статуса и дают различные результаты в зависимости от состояния флага. Программист может устанавливать флаги управления для изменения семантики некоторых команд. Например, команда просмотра строки может иметь направление в сторону больших или меньших адресов в зависимости от состояния флага направления. Системные флаги предназначены для использования операционной системой и в прикладных программах могут игнорироваться. (Системные флаги рассматриваются в главе 3). На практике для исключения возможного изменения системных флагов прикладными программами может быть использована система защиты 80386.

1 - статусные флаги
2 - перенос
3 - четность
4 - вспомогательный перенос
5 - ноль
6 - знак
7 - переполнение



8 -
9 -
10 -
11 -
12 -
13 -
14 -
15 -
16 -

Рис.2-2 Регистр флагов

Счетчик команд 80386, обозначаемый EIP, имеет разрядность 32 бит. Счетчик команд управляет выборкой команд (включая предварительную выборку) и после выполнения команды процессор автоматически увеличивает его на I. Содержимое счетчика команд меняется по прерываниям, в особых и при выполнении команд меняется по прерываниям, в особых случаях и при выполнении команд передачи управления, например, переходах и вызовах.

2.1.3. Регистры математического сопроцессора

Регистры математического сопроцессора, показанные на рис.2-3, повышают производительность систем с большим объемом вычислений. При подключении к 80386 математического сопроцессора 80287 или 80387 эти регистры добавляются к 80386. Хотя математический сопроцессор распознает форматы целых, упакованных десятичных и чисел с плавающей запятой различной длины, внутри него все величины хранятся в формате с плавающей запятой в регистровом стеке 8x80 бит. В математических операциях могут быть как неявные ссылки на верхние элементы стека, так и явные на дригие регистры. Статусный регистр содержит указатель вершины стека, флаги, идентифицирующие особые слу-

чаи (например, переполнение) и коды состояний, отражающие результат последней команды. Регистр управления содержит биты вариантов и масок, которые программист может устанавливать для выбора алгоритма округления, интерпретации бесконечности, а также задания того, как будут обрабатываться особые случаи - сопроцессором или программно.

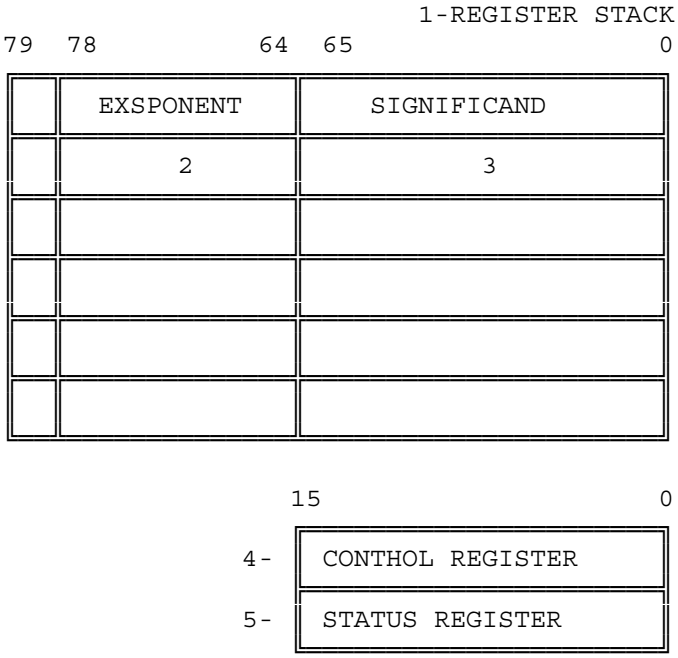


Рис.2-3 Регистры математического сопроцессора; 1 - регистровый стек; 2 - порядок; 3 - мантисса; 4 - регистр управления; 5 - регистр статуса.

2.2. Память и логическая адресация

Для адресации операндов в 4г байтном физическом адресном пространстве прикладные программы 80386 используют логическую адрсацию. Процессор автоматически транслирует эти логические адреса в физические, которые затем выдаются на системную магистраль. Как будет рассмотрено более подробно в главе 3, операционная система 80386 может перестраивать представление прикладной программы о ее логическом адресном пространстве. Например, операционная система 80386 может определить логическое адресное пространство так, как это делается во многих архитектурах , а именно, как простой массив из 2 байт. С другой стороны, она может организовать логическое адресное пространство как набор сегментов переменной длины. Операционная система может определить как большое число сегментов, так и всего несколько, в зависимости от ее представления о логической памяти; 80386 не дисктует конкретное использование сегментов и позволяет использовать их так, как того требует данное применение. Читая дальнейшие разделы, следует помнить о том, что степень активного использования сегментов прикладной программой зависит от того, как они организованы операционной системой.

2.2.1. Сегменты

Как было отмечено выше, операционная система может определить адресное пространство как один или несколько сегмен-

тов. Сегменты являются логоческими блоками, хорошо приспособленными под программные структуры, которые по сути своей имеют переменную длину. Например, 1516-байтная процедура полностью содержит сегменте 1516 байт, так же, как и 8 мбайтный массив (например, дисплейный буфер 1028X1028X8) полностью входит в сегмент такого же размера. Имея для сегментов соответствующие архитектурные решения, 80386 повышает производительность систем, в которых механизм структурирования основан на сегментах. (Страницы, которые рассматриваются в главе 3, имеют фиксированные размеры; они не приспособлены под программные структуры, но, с другой стороны, более подходят для таких функций операционной системы, как, например, пересылки между ОЗУ и диском).

Сегмент в системе 80386 может иметь любой размер от 1 байта до 4гбайт. Для каждого сегмента операционная система поддерживает заданный архитектурой дескриптор, содержащий описание сегмента. Описание сегмента включает в себя 32-битный базовый адрес и длину сегмента, а также информацию о защите, предотвращающей неправильное использование сегмента. Ввиду того, что дескрипторы поддерживаются операционными системами, их рассмотрение откладывается до главы 3. Прикладные программы пользуются дескрипторами лишь косвенно, обращаясь к сегментам с помощью логических адресов.

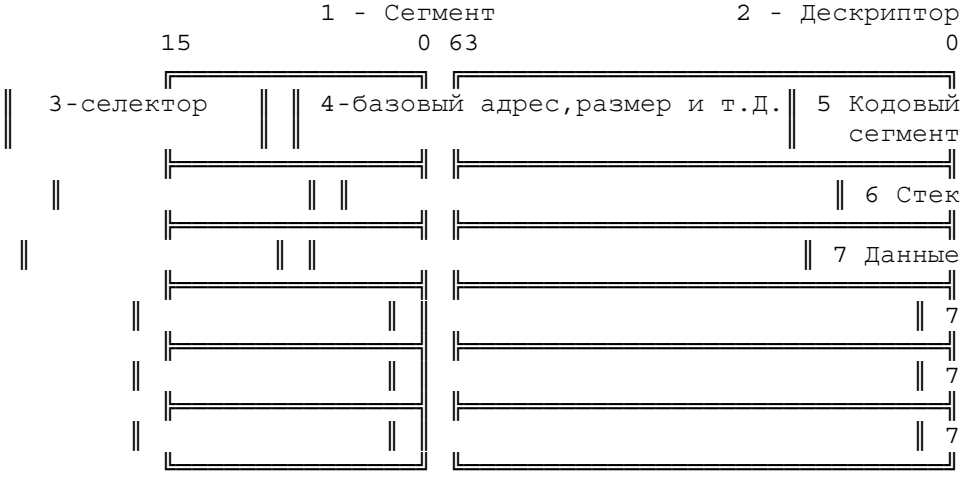
2.2.2. Логические адреса

Ввиду того, что программа может в принципе обращаться к нескольким сегментам, логический адрес 80386 должен идентифицировать сегмент. Поэтому логический адрес 80386 состоит из двух частей, 16-битного селектора сегмента и 32-битного смещения в выбранном сегменте (см. Рис.2-4). После селектора в логическом адресе указывает на дескриптор сегмента. В принципе процессор определяет адрес сегмента с помощью селектора, как указателя для таблицы дескрипторов, поддерживаемой операционной системой. Добавление смещения логического адреса к базовому адресу, полученному по дескриптору сегмента, дает адрес операнда.

Рис.2-4.

2.2.3. Регистры сегментов и дескрипторов

Для повышения эффективности логической адресации в 80386 введено 6 регистров сегментов и дескрипторов (см. Рис.2-5).



8 остальные поля дескрипторов описываются в главе 3.

Фактически эти регистры используются как программно управляемый кэш, позволяющий исключить из большенства команд селекторы и производить трансляцию большинства логических адресов внутри кристалла без обращений к таблице дескрипторов.

Адресные ссылки в большинстве программ производятся в небольших адресных диапазонах (такая "локальность ссылок" делают виртуальную память практичной). Например, если процедура хранится в сегменте, то вероятнее всего большое число команд будет считано из сегмента прежде, чем управление перейдет к другой процедуре в другом сегменте. Локальность ссылок в 80386 обеспечивается программно, путем запоминания во внутренних регистрах последних использовавшихся селекторов и дескрипторов. Внутреннее хранение дескрипторов позволяет транслировать большинство логических адресов без обращений к памяти, занимающих много времени.

В любой момент времени можно адресовать до шести сегментов: кодовый сегмент, сегмент стека и четыре сегмента данных. В сегментных регистрах CS, SS, DS, ES, FS и GS хранятся селекторы этих сегментов. Их дескрипторы хранятся в соответствующих регистрах дескрипторов. В случае необходимости программа может сделать адресуемым новый сегмент с помощью загрузки селектора нового сегмента в сегментный регистр. Процессор автоматически поддерживает регистры дескрипторов, загружая требуемый дескриптор каждый раз, когда программа меняет сегментный регистр. (Фактически, регистры дескрипторов могут загружаться только процессором; программе они не доступны). Счетчик команд содержит смещение текущей команды в текущем кодовом сегменте (определяемом регистром CS), а регистр ESP содержит смещение вершины стека в текущем сегменте стека (определяемом регистром SS).

Высокая эффективность дешифрации команд достигается для большинства из них благодаря отсутствию явных ссылок на сегментные регистры. Например, в командах перехода и занесение в стек ссылки, соответственно, на регистры CS и SS, делаются неявно. В случае необходимости программист может указать про-

цессору на использование конкретного сегмента в данной команде, что осуществляется путем добавления перед командой однобайтного префикса перемены сегмента. Префикс указывает процессору на регистр сегмента, который должен использоваться в трансляции адреса в следующей за префиксом команде.

Сегмент, базовый адрес которого равен 0, а предельный размер - 4гбайт, определяет 4гбайтное логическое адресное пространство. Поскольку процессор выбирает сегментный регистр автоматически, то команда может ссылаться на операнд в любой ячейке этого 4гбайтного пространства с помощью 32-битного смещения. Если, как показано на рис.2-6, Все регистры дескрипторов будут загружены базовым адресом 0 к предельным размерам 4гбайт, то в этом случае сегменты исчезают. Любой байт в логическом адресном пространстве, независимо от того, командный это байт или байт данных, или же элемент стека, адресуется с помощью простого 32-байтного смещения. Таким образом, сегментные регистры дают 80386 шесть одновременно адресуемых логических адресных пространств размером до 4гбайт каждое. В том случае, если эти сегменты совпадают, то программа будет располагать одним 4гбайтным логическим адресным пространстве, идентичным такому пространству, которое обеспечивается менее гибкими 32-битными архитектурами.

2.2.4. Способы адресации

Микропроцессор 80386 обеспечивает регистровую и непосредственную адресацию операндов, содержащихся, соответственно, в регистрах или командах. Еще более важным является способность 80386 обеспечивать различные способы адресации необходимые для эффективного обращения к таким элементам структур данных в памяти как массивы, записи (структуры), массивы записей и записи, содержащие массивы. При этом программа определяет поле смещения в логическом адресе по одному из способов адресации памяти в 80386. Процессор 80386 вычисляет поле смещения логического адреса по следующей формуле:

смещение = база + (идекс X масштаб) + отклонение

Для вычисления смещения могут быть использованы любые или все переменные базы, индекса и отклонения. Переменные базы и индекса являются величинами, хранящимися в общих регистрах, а величина отклонения содержится в команде. Для хранения базы или индекса может быть использован любой общий регистр. Величина в индексном регистре может быть отмаштабирована (умножена) коэффициэнтом 1,2,4 или 8, что дает возможность делать ссылки на элементы массива или записи соответствующей длины. Величина отклонения может иметь разрядность 8 или 32 бит и интерпретируется процесспром как величина со знаком в дополнительном коде.

Разумные комбинации базы, индекса и отклонения дают следующие способы адресации памяти 80386:

- прямая: используется только отклонение;
- регистривая косвенная: используется только база;
- базовая: используется база + отклонение;
- индексная: используется индекс (в масштабе);
- индексная с отклонением: используется индекс (в масштабе) + отклонение;
- базовая индексная: используется база + индекс (в масштабе);
- базовая индексная с отклонением: используется база + индекс (в масштабе) + отклонение;

2.3. Типы данных и команды

В данном разделе будут рассмотрены команды, чаще всего используемые программистами. Поскольку большинство команд оперирует с конкретными типами данных (например, с целыми), эти типы и команды рассматриваются совместно. Привилегированные команды, включая те, которые осуществляют ввод/вывод и обработку прерываний будут рассмотрены в главе 3.

2.3.1. Главные типы данных

В табл. 2-1 Перечислены типы данных и команды, обеспечиваемые процессором 80386. В этой таблице приведены только наиболее употребимые команды. Варианты команд, такие как (в случае циклического сдвига) циклический сдвиг вправо и циклический сдвиг через перенос, также опущены.

Таблица 2-1.

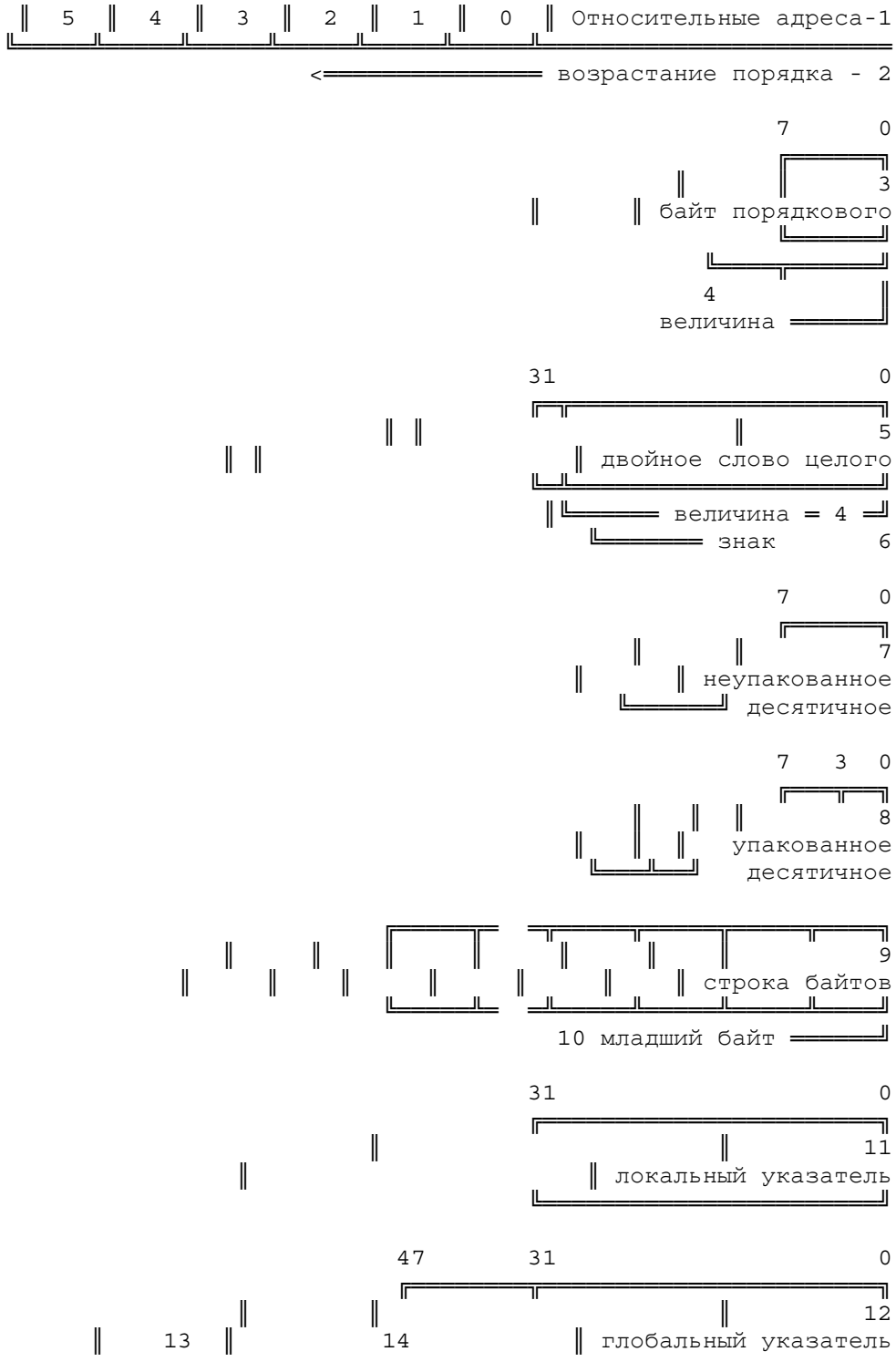
Главные типы данных и команды

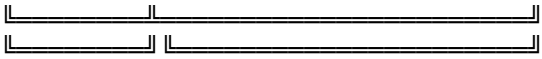
Тип	Разрядность	Команды
Целое, порядковое	8, 16, 32 бит	Пересылка, обмен, преобразование, проверка, сравнение, перевод, сдвиг, двойной сдвиг, циклический сдвиг, отрицание, логическое "и", "или", исключающее "или". Сложение, вычитание, умножение, деление, увеличение на 1, уменьшение на 1, перевод (пересылка с расширением знака/ноля)
Неупакованное десятичное	1 цифра	Коррекция для сложения, вычитания,умножения, деления
Упакованное десятичное	2 цифры	Коррекция для сложения, вычитания
Строка (байтов,слов, двойных слов)	0-4гбайт слов, двойных слов	Пересылка, загрузка, запоминание, сравнение, просмотр, повтор
Строка бит	1-4гбит	Проверка, проверка и установка, проверка и гашение, проверка и дополнение, просмотр, вставление, из'ятие
Локальный указатель	32 бит	(см. Порядковое)
Глобальный указатель	48 бит	Загрузка

Примечание. Локальный указатель - 32 битное смещение в сегменте, определенном одной из регистрированных пар сегмента/дескриптора. Глобальный указатель - это полный логический

адрес, состоящий из селектора и смещения.

На рис.2-7 показаны примеры того, как главные типы данных хранятся в памяти. Многобайтные элементы могут размещаться с любого адреса байта в зависимости от структуры магистрали, для обращения к операндам, размещенным по адресу, не кратным длине операнда в байтах, могут потребоваться дополнительные магистральные циклы. Поэтому для высокой производительности, не зависящей от структуры магистрали, большинство программ ориентируют словные операнды из двойных слов на границах двойных слов и т.п.





селектор смещение

2.3.2. Типы данных математического сопроцессора

Математический сопроцессор 80287 или 80387 добавляют к типам данных и командам процессора 80386 свои, приведенные в табл.2-2. В большинстве прикладных задач входные величины и получаемые результаты хранятся в виде типов целых, действительных или упакованных десятичных, а для промежуточных величин имеется тип данных промежуточное действительное, расширенный диапазон и точность которого в сложных вычислениях сводят к минимуму ошибки округления, переполнения и исчезновения порядка. В соответствии с такой моделью математический сопроцессор производит большую часть вычислений над промежуточными величинами, хранящимися в его регистрах. При загрузке любого типа данных в регистровый стек, этот тип автоматически меняется на промежуточный действительный. Промежуточная действительная величина в регистре, в свою очередь, может быть переведена в любой другой тип с помощью команды запоминания.

На рис.2-8 Показано, как типы данных математического сопроцессора хранятся в памяти.

Таблица 2-2.

Главные типы данных и команды математического сопроцессора

Тип	Разрядность	Команды
Целое	16,32,64 бит	Загрузка, запоминание, сравнение, сложение, вычитание, умножение, деление
Упакованное десятичное	18 цифр	Загрузка, запоминание
Действительное	32,64 бит	Загрузка, запомонание, сравнение сложение, вычитание, умножение, деление
Промежуточное действительное	80 бит	Сложение, вычитание, умножение, деление, извлечение квадратного корня, масштабирование остатка, вычисление части целого, смена знака, вычисление абсолютной величины, выделение порядка и мантиссы, сравнение, осмотр, проверка, обмен, арктангенс, 2-1,Y*LOG(X+1),Y*LOG(X), загрузка константы (0.0, П, и т.д.) (80387 добавляет синус, косинус, синус и косинус, неупорядоченное сравнение).

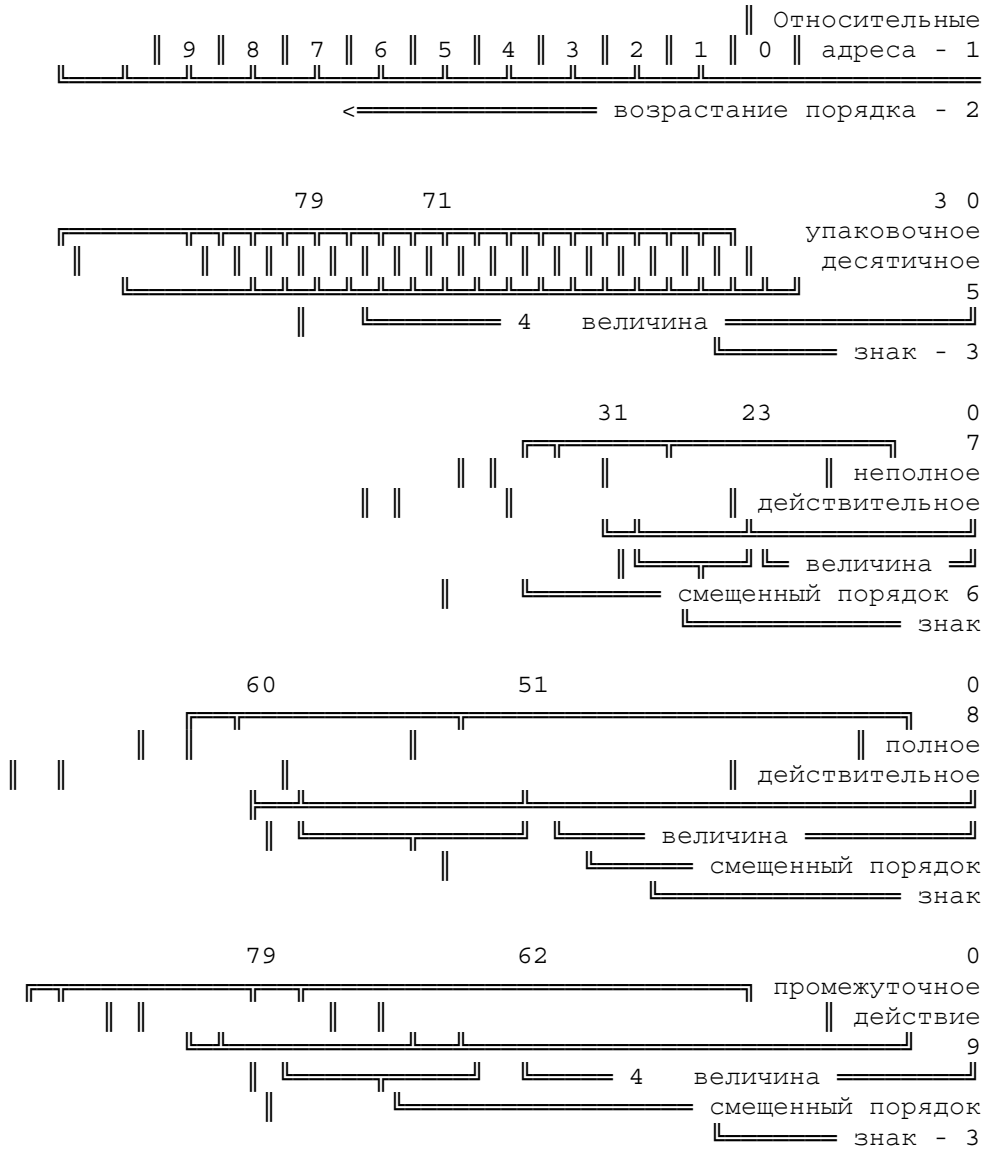


Рис.2-8.Примеры хранения типов данных математического сопроцессора

2.3.3. Другие команды

Не все команды процессора 80386 связаны с типами данных. Такие команды рассматриваются в нижеследующих параграфах.

2.3.3.1. Команды операций со стеком

Стек процессора 80386 является стеком двойных слов, основание и вершина которого определяются регистрами, соответственно, SS и ESP. Команда PUSH заносит двойное слово в стек, а команда POP извлекает с вершины стека двойное слово и харнит его в регистре памяти или в памяти. По команде PUSH ALL в стек заносятся все общие регистры, а по команде POP ALL производится обратная операция.

Команда INTER (входа в процедуру) и LEAVE (выхода из процедуры) предназначены для языков высокого уровня с блочной структурой. По команде ENTER создается кадр и образ стека, которые компиляторы используют для связки вызовов процедур.

По команде LEAVE кадр стека и образ удаляются из стека для подготовки возврата к процедуре, сделавшей вызов.

2.3.3.2. Команды передачи управления

Команда JUMP (переход) передает управление другой команде путем замены содержимого счетчика команд. Новая команда может быть в том же кодовом сегменте (со смещение до 2 байт) или совсем в другом. Операндом внутрисегментного перехода является локальный указатель, т.е. смещение новой команды в текущем кодовом сегменте; переход таким образом, может быть сделан к любой ячейке в максимально возможном сегменте. Операндом межсегментного перехода является глобальный указатель, что позволяет передавать управление в любую точку сегмента. (Поле селектора в глобальном указателе замещает величину в регистре CS, а поле смещение - величину в EIP). В системе команд также имеется полный набор команд условных переходов, ветвление которых основано на величине флага статуса; эти команды могут передавать управление ячейкам, которые также смещены на максимум 2 байта.

Вызов процедур и функций (подпрограмм) производится по командам CALL (вызов), а возврат к вызывавшей подпрограмме осуществляется с помощью команды RETURN (возврат). Так же, как команды перехода, вызовы внутри сегмента имеют своими операндами локальный указатель, задающий новую величину в счетчике команд, а вызовы между сегментами используют в качестве операндов глобальный указатель, который кроме CS изменяет и величину EIP. По командам вызова адрес следующей команды заносится в стек, после чего производится загрузка счетчика команд (и регистра CS, если переход делается в другой сегмент). По команде возвратасохранненные величины извлекаются из стека в EIP и, если требуется, в CS. Вызовы могут иметь бесконечную вложенность и рекурсивность, ограниченные лишь размером стека.

Для управления циклами, помимо условных переходов, 80386 обеспечивает выполнение команд LOOP (безусловно и условного цикла). Команды цикла в качестве счетчика циклов используют регистр ECX; в каждом цикле ECX уменьшается на 1 и выполнение команды заканчивается, когда величина в ECX становится равной нулю. Команды условных циклов заканчиваются в том случае, если флаг содержит заданную величину. В то время как команды цикла предназначены для проверок "в конце цикла", команда "переход", если ECX=0 реализует проверку в начале цикла и позволяет выполнять цикл 0 раз.

2.3.3.3. Дополнительные команды

Команда BOUND (проверка границ) 80386 может быть использована для проверки того, что индекс нассива находится в его границах. Процессор 80386 имеет также команды установки и гашения флагов, загруаки и запоминания байта статуса регистра флагов.

Математический сопроцессор 80287 или 80387 добавляет команды, необходимые операционной системе для его инициализации, обработки особых случаев, а также для запоминания и восстановления статуся сопроцессора.

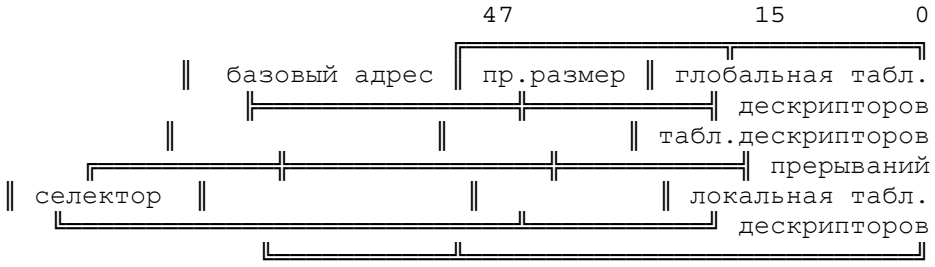
Наконец, естественно, процессое 80386 имеет команду "нет операции".

3. Системная архитектура

Назначение системной архитектуры заключается в обеспечении иперационных систем, однако операционные системы весьма различны по своим требованиям. Для решения этой проблемы, процессор 80386 обеспечивает набор ресурсов, которые разработчики операционных систем могут использовать по своему усмотрению. В результате системная архитектура 80386 может быть сконфигурирована так, чтобы удовлетворить всем требованиям разрабатываемой операционной системы.

3.1 Системные регистры

Кроме регистров, рассмотренных в предыдущей главе, операционная система иногда использует регистры 80386, показанные на рис.3-1. (Далее в этой главе эти регистры еще будут рассматриваться; здесь они показаны для справки). В основном этими регистрами пользуется сам 80386; операционная система лишь инициализирует системные регистры и игнорирует их при нормальной работе. Однако, операционная система может воспользоваться системным регистром при обработке особого случая. Например, при страничной ошибке процессор загружает ошибочный адрес в регистр CR2; обработчик страничных ошибок операционной системы использует этот адрес для поиска соответствующего элемента страничной таблицы. Системные регистры обычно недопустимы прикладным программам, поскольку оперировать с ними могут только привилегированные команды. (Особые случаи, страничные ошибки и привилегированные команды рассматриваются далее в этой главе).



системные адресные регистры

системные управляющие регистры

3.2. Обеспечение многозадачных операционных систем

Многие свойства системной архитектуры 80386 непосредственно обеспечивают многозадачные операционные системы, хотя, конечно, 80386 может быть использован и в однозазадачных системах с повушенными тербованиями. Многозадачная работа предствляет собой способ управления работой вычислительной систе-

мой в тех случаях, когда работа системы состоит из нескольких видов деятельности; тремя видами деятельности могут быть например, редактирование одного файла, компиляция другого и передача третьего файла в другую машину.

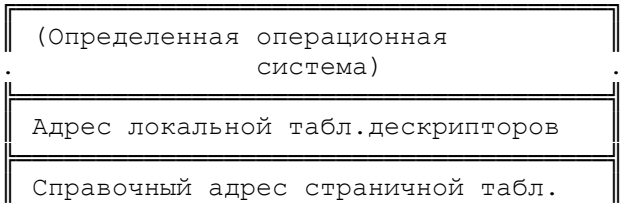
В многозадачной системе каждый вид деятельности, который может осуществляться одновременно с другими, называется задачей. (В данном материале термин "задача " эквивалентен термину "процесс"). Каждая задача выполняет программу, состоящую из команд и исходных данных. Одна и та же программа может выполнятся несколькими задачами; например, в многозадачной системе с разделением времени несколько задач (по числу пользователей) могут использоваться одним и тем же компилятором или редактором. Программы и задачи соотносятся друг с другом подобно партитуре музыкального произведения и его исполнению: программа - это текст, описывающий алгоритм, а задача - это однократное исполнение этого алгоритма.

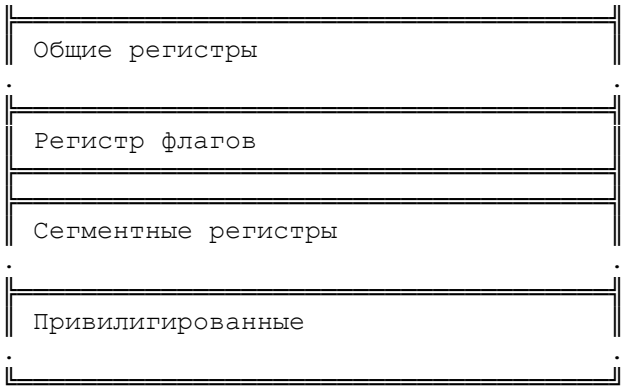
Программы, выполняемые задачами. Составлены так, как если бы они выполнялись на отдельных процессорах с общей памятью, т.е. Используя паузы, необходимые для связи или синхронизации с другими задачами, теоретически каждая задача выполняется непрерывно параллельно с другими задачами. На самом деле , однако, задачи выполняются поочередно одним процессором.

Многозадачная операционная система моделирует несколько процессоров, предоставляя каждой задаче "виртуальный процессор". В каждый момент времени операционная система передает реальный процессор одному из виртуальных процессоров, выполняющему свою задачу. Для поддерживания впечатления, что каждая задача имеет свой процессор, операционная система часто переключает реальный процессор на различные виртуальные процессоры. В системной архитектуре 80386 для операции смены задачи предусмотрены сегменты состояния задачи и команды, выполняющие эту операцию.

3.2.1. Сегмент состояния задачи

Сегмент состояния задачи (TSS) является одной из нескольких структур данных, определяемых системоной архитектурой 80386. Фактически, эти структуры данных являются "типами данных" для операционных систем. Сегмент TSS(см.Рис.3-2) Соответствует тому, что в некоторых операционных системах называется блоком управления задачей; в этом сегменте хранится состояние виртуального процесора задачи. Каждая задача 80386 представлена своим TSS, который делится на две части. Младшая часть TSS определена системной архитектурой 80386 и содержит значения регистров процессора. Старшая часть TSS может быть определена операционной системой для хранения данных, связанных с задачей, например, приоритета выполнения, дескрипторов файлов и т.д. Дла создания своей задачи операционная система формирует TSS и инициализирует его величинами, необходимыми задаче для начала ее выполнения. В результате 80386 поддерживает младшую часть TSS, а за его старшую часть несет ответственность операционная система.





3.2.2. Смена задачи

Операционная система разрешет выполнение ??????? в соответствии с планом. Этот план устанавливает время выполнения задач. Поскольку методы планирования ???????? различные, 80386 предоставляет это операционной системе. ???? Когда операционная система решает начать выполнение ????? задачи, она направляет процессор на выполнение еще одного ?????, иногда называемого сменой контекста.

Процессор 80386 хранит селектор и дескриптор ????? текущей задачи в своем регистре задачи ?????? операционная система выдает команду перехода ???????? является селектор сегмента TSS новой задачи ?????? эту команду, занося вначале свои регистры в текущий TSS, а затем загружая TR селектором (и связанным с ним дескриптором), указанным в команде. Получив адрес нового TSS, процессор загружает свои регистры величинами из нового TSS. После чего выполнение программы продолжается с команды, на которую указывает счетчик команд новой задачи. Для возобнавления старой задачи операционная система должна выдать команду перехода и TSS старой задачи, после этого выполнение старой задачи продолжается с командой, следующей после команды перехода TSS, прекратившей ее выполнение. Такая смена задачи занимает 17 мкс (при рабочей частоте 16 мгц и отсутствии состояний ожидания).

3.3. Адресация

Физическое адресное пространство большинства вычислительных машин организовано просто как массив байтов. С появления блоков управления памятью (БУП), архитектура машин стала различать физическое адресное пространство, реализуемое, аппаратурой памяти и логическое адресное пространство, видимое программистом. Блок управления памятью транслирует логические адреса программ в физические адреса, выдаваемые на системную магистраль. В большинстве архитектур логическое адресное пространство задачи представляется как набор следующих вариантов:

Байты	логическое адресное простарнство состоит из массива байтов, не имеющего определенной структуры (такое адресное пространство иногда называется "плоским" или "линейным"). Трансляция адреса в этом случае не требуется, поскольку логический адрес точно эквивалентен физическому.
Сегмент	логическое адресное пространство состоит

из нескольких или большого числа сегментов, каждый из которых содержит переменное число байтов. Логический адрес разделен на две части, номера сегмента и сМещения внутри сегмента. Буп производит трансляцию логического адреса в физический.

Страницы — логическое адресное пространство состоит из большого числа страниц, каждая из которых включает в себя фиксированное число байтов. Логический адрес состоит из номера страницы и смещения внутри страницы. Буп производит трансляцию логического адреса в физический.

Страничные сегменты — логическое адресное пространство состоит из сегментов, которые в свою очередь, состоят из страниц. Логический адрес состоит из номера сегмента и смещения внутри сегмента. Буп производит трансляцию логического адреса в номер страницы и смещение в ней, которые затем транслируются в физический адрес.

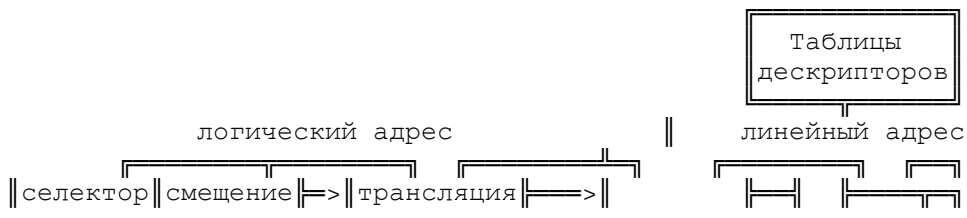
Каждый из этих вариантов хорошо подходит для одних систем и мало пригоден для других. Например, линейное пространство вполне подходит для систем с простыми ветвлениями, в то время как для систем, которые выполняют индивидуальное управление и защиту отдельных программных структур, больше подходит вариант с сегментацией памяти. В 80386 реализован вариант, представляющий память как набор сегментов, которые по желанию могут быть разделены на страницы. На практике архитектура 80386 обеспечивает операционные сситемы любым из четырех вариантов представления памяти.

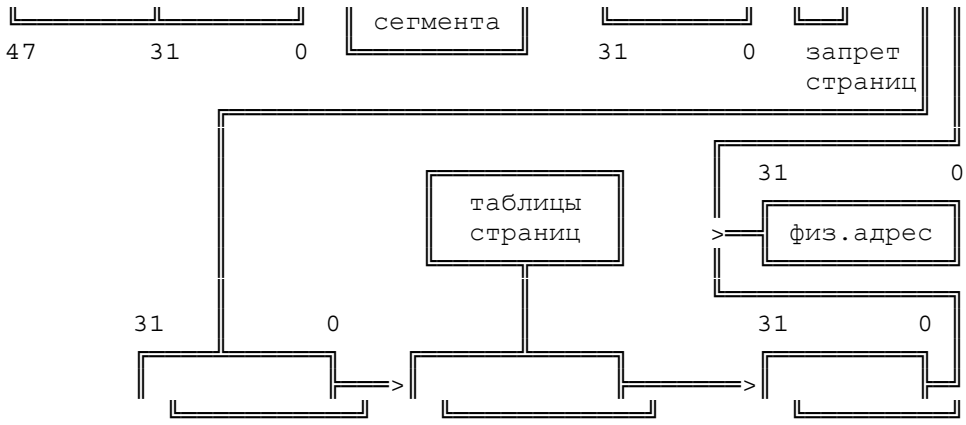
3.3.1. Принцип трансляции адреса

Принцип трансляции логического адреса в физический в процессоре 80386 иллюстрируется на рис.3-3. Последовательность операций, показанная на рис.3-3, является центральной как для адресации, так и для защиты. Здесь она рассматривается в схематичной форме с целью дать ясное общее представление о ней прежде, чем перейти к рассмотрению таких свойств, как виртуальная память и защита.

В последующих разделах будут подробно рассмотрены различные этапы трансляции адреса и будет показано, как они могут быть приспособлены под требования конкретной системы.

Как показано в предыдущей главе, способы адресации памяти 80386 дают 32-битное смещение искомого операнда. Совместно с селектором сегмента это смещение образует составной логический адрес: селектор этого адреса идентифицируетискомый сегмент, а смещение указывает на операнд в сегменте. В большинстве команд селектор задается неявно как содержимое сегментного регистра.





Селектор представляет собой индекс в таблице дескрипторов сегментов, т.е. это поле содержит номер сегмента. Каждый элемент таблицы дескрипторов сегментов содержит базовый адрес сегмента. Процессор добавляет к нему смещение и получает 32-битный линейный адрес. Если страницы не разрешены, процессор считает, что линейный адрес является физическим, и выдает его на адрсные выводы.

Если страницы разрешены, то 80386 транслирует линейный адрес в физический. Это делается с помощью страничных таблиц. Страничная таблица по своей организации аналогична таблице дескрипторов, за исключением того, что каждый элемент страничной таблицы содержит физический базовый адрес страницы 4кбайт.

Поскольку способы адресации 80386 охватывают как традиционные элементы структурного деления адресного пространства (сегменты, и дополнительно, страницы) и поскольку сегменты могут быть очень большими (до 4 гбайт), то эти способы адресации оказываются очень гибкими. Таким образом, операционная система может дать задаче одно линейное адресное пространство, линейное адресное пространство из страниц, адресное пространство из сегментов или сегментированное адресное пространство со страничным делением.

По всей своей гибкости многоступенчатая трансляция адреса в 80386 выполняется достаточно быстро. Типичное время вычисления смещения и трансляции логического адреса в физический состовляет 1,5 такта. Более того, время трансляции адреса незаметно для программы, поскольку внутренний БУП 80386 транслирует адрес параллельно с другими операциями процессора (кроме случаев, когда команды перехода или вызова временно прерывают совмещенное выполнение операций).

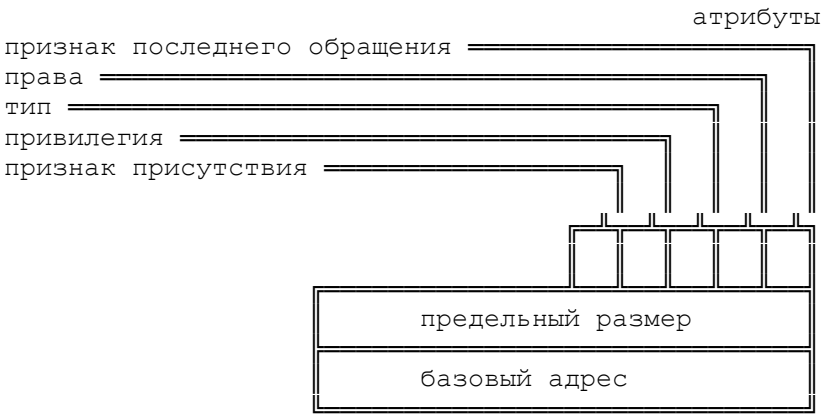
3.3.2. Сегменты

Сегмент является единицей логического пространства, представляемого процессором задаче, т.е. логическое адресное пространство задачи состоит из одного или нескольких сегментов. Операционные системы существенно отличаются друг от друга по способу определения логического адресного пространства задачи. Например, система реального времени с ветвлениями может определить логическое адресное пространство как единое целое, совместно используемое всеми задачами и самой операционной системой, другими словами один сегмент коллективно используется всей системой. В другом крайнем случае система может распределить каждую структуру данных и процедуру по своим

сегментам, вследствии чего логическое адресное пространство задачи предстанет в виде десятков или сотен адресных пространств, каждое из которых будет соответствовать своей структуре данных или процедуре. Между этими крайними случаями оказывается универсальная система с разделением времени, в которой задачи выполняются в отдельных логических адресных пространствах, и в которых программы задач отделены от их данных, а прикладные программы и данные отделены от программы и данных операционной системы. Свойство сегментации в процессоре 80386 достаточно гибкое, и может обеспечить каждый из этих примеров, как и любые другие.

Как уже было показано в главе 2, команда 80386 ссылается на операнд в памяти с помощью составного логоческого адреса, состоящего из селектора сегмента и смещения внутри сегмента. В принципе 80386 транслирует логический адрес в линейный с помощью селектора, указывающего на дескриптор сегмента в таблице дескрипторов. Дескриптор содержит базовый адрес сегмента в линейном адресном пространстве, добавление смещения к базовому адресу дает линейный адрес операнда. На практике трансляция логического адреса в линейный оптимизирована с помощью неявного указания на селекторы и хранения дескрипторов в регистрах. Поэтому обращение к таблице дескрипторов производится толькодля тех команд, которые загружают новые селекторы в сегментные регистры (например, вызов процедуры из другого сегмента приводит к замене селектора в регистре CS). Хотя на практике это бывает редко, тем не менее удобнее представлять трансляцию процессором логического адреса как обращение к дескрипторам в таблицах дескрипторов сегментов, поскольку отсюда следует, что именно дескрипторы в таблицах дескрипторов сегментов задачи определяют логическое адресное пространство задачи. Без дескриптора задача не в состоянии сгенерировать линейные адреса.

Таблица дескрипторов сегментов представляет собой массив дескрипторов, на рис.3-4 дан логический формат дескриптора. Поле базового адреса уже рассматривалось выше. Поле предельного размера определяет длину сегмента, 80386 использует поле предельного размера для проверки правильности величины смещения в логическом адресе, а именно, что оно попадает внутрь сегмента. Аттрибуты сегмента, в основном, относятся к защите и будут рассмотрены далее в этой главе.



Каждая задача может иметь системное и индивидуальное логическое адресное пространство. Эти пространства описываются, соответственно глобальной таблицей дескрипторов (GDT) и локальной таблицей дескрипторов (LTD). (В селекторе имеется бит, связывающий его с той или иной таблицей). Эти таблицы

дескрипторов могут содержать максимум 8192 дескриптора и совместно они определяют логическое адресное пространство задачи. Это означает, что для того, чтобы новый сегмент мог адресоваться задачей, операционная система должна внести новый дескриптор этого сегмента либо в GDT, либо в LDT задачи. В системах с защитой GDT и LDT могут быть сделаны привилегированными структурами так, что изменения в них сможет производить толцко операционная система.

Операционная система из ее названия, глобальная таблица дескрипторов доступна всем задачам, обычно операционные системы заносят дескрипторы системных сегментов в GDT. Кодовый сегмент (или сегменты) операционной системы является хорошим примером сегмента, который должен быть доступным всем задачам, и дескриптор которого обычно находится в GDT. В противоположность этому каждая задача может иметь свою локальную таблицу дескрипторов. Процессор поддерживает адрес LDT текущей задачи в своем регистре локальной таблицы дескрипторов (LDTR), однако он перезагружает этот регистр (так же, как он перезагружает свои общие и сегментный регистры) новым значением TSS при смене задач.

Задачи могут делить одни сегменты в следующух случаях (см.Рис.3-5):

1. Сегмент, дескриптор которого находится в GDT, является общим для всех задач.
2. Задачи, имеющие общую LDT, имеют и общие сегменты, описываемые этой LDT, такой подход допустим в случае тесно взаимодействующих задач.
3. Дескрипторы в различных LDT могут указывать на один о тот же сегмент, такие дескрипторы называются псевдоименами. Использование псевдоимени позволяет задачам иметь общим только один сегмент, а не все сегменты, определенные таблицей дескрипторов.

3.3.3. Страницы

Независимо от того, содержит ли логическое адресное пространство задачи один или много сегментов, операционная система может разделить линейное адресное пространство на страницы. С точки зрения операционной системы страницы являются удобными элементами для распределения и перераспределения памяти, поскольку они все имеют одинаковый размер. Страницы также являются средством защиты частей больших сегментов и, что особенно важно, обеспечивают удобный способ организации виртуальной памяти. Указанные применения страничной организации памяти обсуждаются ниже.

Страница в 80386 имеет длину 4к байт. Этот размер находится в согласии с промышленной тенденцией к увеличению длины страниц и ведет к увеличению производительности по двум направлениям.

Во-первых, он обеспечивает хорошее отношение показаний страниц в кэш при данном объеме кэш-памяти, которая может быть в настоящее время реально размещена на кристалле, исходя из достигнутого уровня технологии. (Кэш-память 80386, расположенная на кристалле, описывается кратко). Во-вторых, 4к байтов являются эффективным размером блока для дисковых передач. Большинство операционных систем, работающих на машинах с меньшей длиной страницы, должны группировать страницы в "кластеры", чтобы сократить количество дисковых передач.

Операционная система 80386 включает страничный механизм, устанавливая бит PG в управляющем регистре 0 при помощи при-

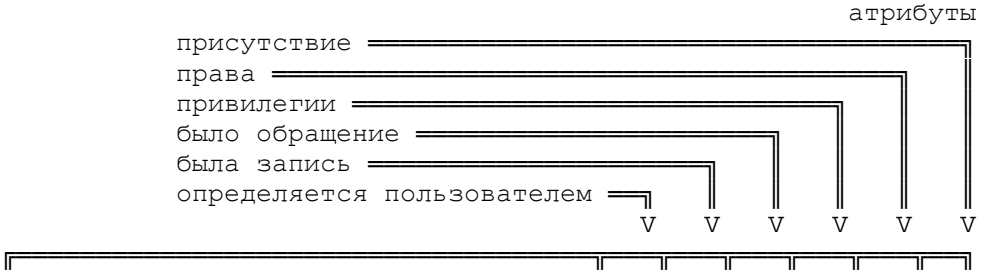
вилегированной команды. Когда страничный механизм включен, процессор транслирует линейный адрес в физический адрес, используя страничные таблицы. Страничные таблицы являются частью таблиц дескрипторов сегментов. Так же как таблицы дескрипторов сегментов задачи определяют ее логическое адресное пространство, страничные таблицы задачи определяют ее линейное адресное пространство. Аналогично супер-мини-эвм и большим эвм страничные таблицы 80386 организованы по принципу двухуровневой иерархии, как показано на рис.3-6. Каждая заадча может иметь справочник системных таблиц. Системный регистр CR3 процессора 80386 (база справочника системных таблиц) указывает на справочник страничных таблиц работающей задачи. Процессор обновляет регистр CR3 при каждом переключении задачи, получая новый адрес задачи из TSS новой задачи. Справочник страничных таблиц имеет длину, равную одной странице и содержит элементы страничной таблицы описывают 1024 страницы. Таким образом, кацдая страничная таблица описывает 4м байта, а оглавление может описать до 4г байт - максимальное адресное пространство.

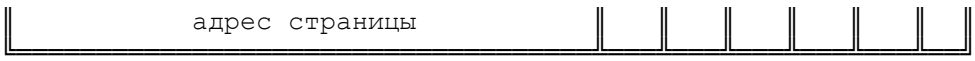
На рис.3-6 в функциональных обозначениях показано как 80 786 транслирует линейный адрес в физический адрес, если страничная адресация включена. Процессор использует старшие 10 бит линейного адреса как индекс в оглавлении. Выбранный элемент оглавления содержит адрес страничной таблицы. Процессор добавляет средние 10 бит линейного адреса к адресу страничной таблицы, чтобы найти элемент страничной таблицы, который описывает необходимую страницу. Добавляя младшие 12 бит линейного адреса к адресу начала страницы, процессор получает 32-битный физический адрес.

Для того, чтобы сократить накладные расходы, возникшие от поиска в страничных таблицах, 80386 помещает справочную информацию о 32-х последних страницах, использовавшихся им, в специальную быструю память (кэш), находящуюся внутри самого микропроцессора. Эта память называется перекодированным буфером просмотра исключений (TLB). Только в том случае, если процессор не находит справочной информации для страницы в TLB, он обращается к справочникам и страничным таблицам, находящимся в памяти. Как правило, 98-99% адресных ссылок попадают в TLB, не требуя обращений к основной памяти для трансляции. Когда необходимой информации в TLB не оказывается, процессор заменяет наиболее старый элемент TLB новым элементом. Принцип локальности ссылок предполагает, что новый элемент скорее всего будет использоваться в будещем.

Поскольку включение старничного механизма не увеличивает времени трансляции адреса, оно мало влияет на время исполнения команды только при непопаданиях в TLB.

На рис.3-7 показаны основные поля элемента страничной таблицы (PTE). Элементы справочника страничных таблиц идентичны показанному за тем исключением, что поле адреса рассматривается как физический адрес страничной таблицы, а не адрес страницы.





Задачи могут совместно пользоваться отдельными страницами или целыми страничными таблицами. Элементы 1 различных страничных таблиц, указывающие на одну и ту же страницу , являются синонимами один для другого, также как дескрипторы с теми же самыми адресами являются синонимами друг для друга.

В двух уровневой структуре страничных таблиц процессора 80786 легче разделять страницы между задачами путем разделения целой страничной таблицы. Поскольку адрес разделяемой страницы в этом случае находится в одной страничной таблице, операционной системы необходимо изменить только один элемент страничной таблицы, когда она перемещает эту страницу.

3.3.4. Виртуальная память

Виртуальная память позволяет очень большим программам или группам работать в значительно меньшем объеме физической памяти без использования техники оверлеев. Системы с виртуальной могут основываться или на сегментах, или на страницах. В обоих случаях основная идея виртуальной памяти состоит в том, чтобы использовать значительно более дешевую дисковую память вместо полупроводниковой памяти. Операционная система в случае с виртуальной памятью запоминает все сегменты или страницы в большой области дисковой памяти, называемой часто "областью обмена". Значительно меньшая физическая (реальная) память содержит только наиболее часто используемые сегменты или страницы. До тех пор, пока сегменты или страницы запомненные на диске, используются нечасто, система с виртуальной памятью будет вести себя также хорошо, как и система со значительно большей оперативной памятью, но за значительно меньшую цену. Ключевые архитектурные свойства, необходимые для эффектовного использования виртуальной памяти, перечисляются ниже:

- бит для каждого сегмента или старницы, который говорит процессору (или устройству управления памятью), имеется ли данный сегмент или страница в памяти, или необходимо загрузить его (ее) с диска;
- механизм ловушки или особого прерывания, при помощи которого процессор может указать операционной системе о необходимости загрузки в память отсутствующего сегмента или страницы;
- перезапускаемые команды, которые позволяют процесору выбирать заново команду после того,как операционная система загрузила только что неприсутствующую страницу в память и пометила ее присутствующей.

Процессор 80786 имеет все эти необходимые свойства, а также другие, которые улучшают эффективность механизма управления виртуальной памятью. Как дескрипторы, так и элементы старничных таблиц имеют бит присутствия и поэтому могут быть использованы как основа для построения виртуальной памятью. Обмен сегментами между памятью и диском являются разумным подходом, когда сегменты относительно малы, как это было в 16-битных архитектурах. Когда сегменты могут быть очень велики, как это возможно в 80786, обмен страницами обычно является более эффективным приемом благодаря фиксированному размеру страниц. В страничных системах операционная система распределяет и освобождает память элементами размером со страницу, называемыеми старничными кадрами; старница , загружаемая с

диска, будет подходить любому имеющемуся кадру. Поскольку большинство из 32-битных систем с виртуальной памятью основывается на механизме страниц, оставшаяся часть этого раздела описывает страничный виртуальный механизм 80786.

В общем случае операционная система со страничной виртуальной памятью пересылает отсутствующие страницы с диска в старничные кадры по запросам, т.е. когда она будет оповещена процессором, что команда обращается к отсутствующей странице. Когда количество свободных кадров становится маленьким, операционная система также передает страницы из страничных кадров на диск, пытаясь изъять те страницы, которые вряд ли будут использованы в ближайшем будущем. Путем такого "прозрачного" обмена старницами между страничными кадрами и диском операционная система создает у прикладного программного обеспечения иллюзию, что физическая память имеет размер очень большой области обмена на диске. Ниже подробно описывается этот механизм.

Когда в процессе трансляции логического адреса процессор получает линейный адрес, который ссылается на элемент страничной таблицы, чей бит присутствия сброшен, в процессоре возникает особая ситуация, называемая, условно, страничной ошибкой. Особые ситуации рассматриваются ниже в этом разделе, но основным следствием этой ошибки является вызов процессором процедуры операционной системы, называемой обработчиком страничной ошибки. При входе в обработчик страничной ошибки Uправляющий регистр 2 содержит линейный адрес, связанный с отсутствующей страницей. По этому адресу обработчик страничной ошибки может найти относящийся к нему элемент страничной таблицы путем трансляции линейного адреса, как это делает процессор. Отметим, что все остальные биты, за исключением бита присутствия в элементе страничной таблицы, определяются пользователем; они представляют для операционной системы удобное место для запоминания дискового адреса отсутствующей страницы. Определив дисковый адрес отсутствующей страницы, обработчик страничной ошибки может найти свободный страничный кадр и передать страницу с диска в этот кадр. Затем обработчик старничной ошибки изменяет адресное поле в элементе страничной таблицы и бит присутствия, после чего возвращает управление программе, прерванной страничной ошибкой. После этого процессор автоматически заново выбирает команду, в которой произошла страничная ошибка, и результат будет таким же, как если бы страница ранее находилась в памяти, когда команда начала исполнятьяся в первый раз.

Другие поля в элементе старничной таблицы процессора 80786 помогают операционной системе эффективно производить действия, связанные с управлением виртуальной памятью. Кроме загрузки старниц по запросам операционная система должна поддерживать набор свободных страничных кадров, которые могут быть распределены обработчиком страничных ошибок. Для увеличения набора свободных старничных кадров операционная система должна знать, какой кадр освободить. Прежде чем освободить кадр, операционная система должна записать эту страницу на диск, если страница была модифицирована с тех пор, когда она была загружена. Для того, чтобы помочь операционной системе в этой длительности, архитектура 80786 имеет бит доступа и бит изменения в каждом элементе страничной таблицы, процессор изменяет эти биты автоматически, для всех имеющихся в памяти страниц 80786 устанавливает бит доступа всегда, когда происходит чтение или запись в данной странице, иустанавливает бит изменения, когда происходит запись в эту страницу. Путем периодического просмотра и сброса битов доступа операционная

система может определить страницы, которые давно не использовались. Кадры, содержащие такие страницы, являются хорошими кандидатами на освобождение, поскольку стрсницы, которые давно не использовались, вряд ли будут использованы в ближайшем будущем. Когда операционная система выбрала страницу для освобождения ее страничного кадра, страница не должна записываться на диск за исключением случая, когда процессор установил ее бит изменения.

Каждый элемент старничной таблицы также содержит 3-х битное поле, которое операционная система может использовать по своему усмотрению. Операционные системы обычно используют это поле для маркировки страниц специальными знаками такими как, например, "закрыто для ввода/вывода".

3.4. Защита

Процессор 80786 обеспечивает несколько мрханизмов защиты, которое операционная система может селективно выбирать для своих нужд. Одна из форм защиты - отделение пространств адресов задач при помощи таблиц дескрипторов сегментов и старничных таблиц, обсуждалась выше. Это разделение эффективно предотвращает налоложение кодов и данных различных прикладных задач. В дополнение к изоляции задач друг от друга 80386 обеспечивает возможность защиты операционной системы от прикладных задач, защиту одной части операционной системы от других частей и защиту задач от некоторых их собственных ошибок. Кроме того, что система защиты 80386 делает операционные системы более надежными, она может упростить процесс отладки, используя прерывания (ловушки) по ошибкам для конкретных задач. Все свойства защиты процессора 80386 реализованы внутри кристалла, поэтому действия по проверке защиты не вляют на производительность процессора.

3.4.1. Привилегии

Многие из свойств защиты процессора 80386 базируются на понятии иерархии привилегий. В любой момент привилегии задачи эквивалентна уровню привилегий исполняемого кодового сегмента. В каждом дескрипторе сегмента имеется поле, которое определяет уровень привилегии связанного с ним сегмента, поле может принимать 0 - это наиболее привилегированный уровень, а уровень привелегии 3 - наименее привилегированный.

На рис.3-8 Показано, как уровни привилегии процессора 80386 могут быть использованы для установки различных стратегий защиты. Система без защиты может быть реализована путем простого помещения всех процедур в сегмент (или сегменты), чей уровень привилегии равен 0. Традиционное разделение на супервизора и пользователя может быть реализовано путем помещения прикладной задачи в сегмент с уровнем привилегии 3, а процедур супервизора - в сегмент чей уровень привилегий равен 0. Операционная система может также использовать уровни привилегии 1 и 2, если это необходимо. Например, наиболее критические и наиболее изменяемые процедуры операционной системы (иногда называемые ядром операционной системы) могут иметь уровень привилегий 0. Уровень привилегий 0. Уровень привилегий 1 может быть использован для функций операционной системы , которые являются менее критическими и более часто изменяются или расширяются, например, для драйверов устройств. Уровень 2 может быть использован для использования производителями оригинального обурудования, такие производители оригинального оборудования должны затем присваивать уровень приви-

легий 2 своим программам, оставляя уровень привилегий 3 для конечных пользователей. В этом случае программы производителей оригинального оборудования защищены от программ конечных пользователей. Операционная система защищена как от программ производителей оригинального оборудования и программ конечных пользователей, а ядро операционной системы защищено от всех остальных программ, включая ту часть операционной системы, которая является предметом частных изменений.

Как будет показано ниже, уровень привилегий задачи определяет, какие команды можно использовать и какое подмножество сегментов и/или страниц в их адресном пространстве они могут обрабатывать (исполнять). Процессор осуществляет проверку на допустимость работы согласно уровню привилегии задачи и уровню привилегии сегментов или страниц, которые являются операндами команд. Любая попытка задачи использовать более привилегированный сегмент или страницу приводит к остановке работы процессора над командой и возникновению особой ситуации защиты. (Особые ситуации описываются в данном разделе ниже как системные вызовы, которые обеспечивают управляемый путь для вызова менее привилегированными процедурами более привилегированных процедур).

3.4.2. Привилегированные команды

В дополнение к тому, какие сегменты и страницы могут быть использованы, уровень прувулегии задачи определяет команды, которые задачей могут быть использованы. Процессор 80386 имеет подмножество команд, исполнение которых должно быть тщательно проанализировано для того, чтобы предотвратить серьезные разрушения системы. Все команды, которые загружают новые значения в системные регистры, являются примерами привилегированных команд. Только задача, работающая на уровне привилегий 0, может исполнять привилегированные команды.

3.4.3. Защита сегментов

Дескрипторы в регистрах LDT и GDT - задач определяют логическое адресное пространство задачи. Сегменты, определенные в этих таблицах, теоретически адресуемы, поскольку таблицы дескрипторов обеспечивают информацией, необходимой для вычисления адреса сегмента. Однако адресуемый сегмент не может быть доступен для некоторых операций, из-за дополнительных проверок защиты, осуществляемых процессором 80386. Процессор проверяет каждое обращение к сегменту (сгенерированное при исполнении команды или при выборке команды), чтобы определить, что обращение согласуется с атрубутами защиты сегмента, как это описано ниже.

Привилегия. Чтобы получить доступ к сегменту, программа должна иметь, по крайней мере, такую же привилегию как сегмент. Например, программа, работающая на уровне 3, может обращаться только к тем сегментам, чей уровень привилегий также равен 3, в то время, как программа, работающая на уровне 0, может обращаться ко всем сегментам в своем адресном пространстве.

Граница. Обращение к сегменту должно находиться внутри границ сегмента. Границы сегмента позволяют процессору обнаруживать программные ошибки, такие как переполнение стека, неверные указатели и индексы массивов, а также неправильные адреса вызовов и переходов. В случаях, когда операционная система может определить, что обращение за границы сегмента не является ошибкой (переполнение стека является примером для

некоторых систем), операционная система может расширить сегмент (например, путем добавления старницы к нему) и начать команду с начала.

Тип. Каждый дескриптор содержит поле типа, которое процессор проверяет на соответствие команде, которую он исполняет. Обычные сегменты имеют тип команд или данных , позволяя процессору, например, обнаружить попытку записи в существующий код, например, типы сегментов, непосредственно работающие в прикладных программах - это команды и данные. Системные дескрипторы также имеют тип, так что процессор может, например, проверить при переключении задач, что сегмент, указанный в команде JUMP TSS, действительно является сегментом сосотояния задачи.

Права. Дескриптор сегмента может быть помечен правами, ограничивающими операции, которые можно производить со связанным с ним сегментом. Сегменты команд могут быть помечены как исполняемый или читаемый. Сегменты данных могут быть помечены как доступные толко для чтения или как доступные для чтения и записи.

Все проверки, описанные выше, зависят от целостности дескрипторв. Если задача, исполняющая прикладную программу, могла бы изменять дескриптор, проверка ничего бы не гарантировала. По этой причине операционная система может ограничить доступ к таблицам дескрипторов только для программ с уровнем привилегии 0.

Заметим, что в случае разделяемых сегментов различные дескрипторы для одного и того же сегмента (т.е. синонимы) могут иметь различные атрибуты защиты, позволяя, например, одной задаче читать и писать сегмент, в то время, как другой только читать его. Синонимы также позволяют операционной системе преодолеть механизм защиты, если это необходимо, например для перемещения кодового сегмента.

3.4.4. Защита страниц

Системы, которые широко не используют защиту сегментов, вместо этого могут защищать страницы (защита старниц может быть также приложима к отдельным чатям больших сегментов). Аналогично дескриптору элемент страничной таблицы имеет набор атрибутов защиты, процессор 80386 проверяет каждое обращение к странице на соответствие этим атрибутам.

Элемент страничной таблицы может быть отмечен одним из двух уровней привилегий: пользовательский или супервизорный. Пользовательский уровень соответствует уровню привилегий 3, а супервизорные страницы могут быть досатупны только задачам, работающим с уровнями привилегий 0, 1 или 2, пользовательская страница может быть отмечена как доступная только для чтения или для чтения и записи.

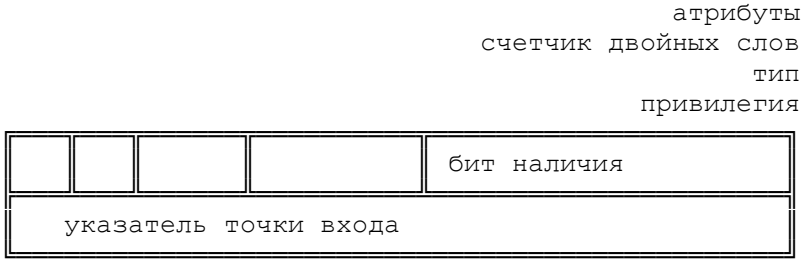
Процессор 80386 проверяет атрибуты защиты страниц после того, как он удостоверился, что доступ находится в соответствии с атрибутами сегмента. Таким образом, защита старниц является удобным средством для операционной системы реализовать дополнительную защиту частей сегментов. Например, операционная сиситема может базопасно заполнить данные операционной системе, относящиеся к задачам, такие как страничные таблицы и дескрипторы файлов, в сегменте данных задачи, обозначив страницы, где расположены эти данные, как супервизорные.

3.5. Системные вызовы

Большинство операционных систем организуют свои функции

как набор процедур, которые могут быть вызваны задачами. Незащищенная операционная система процессора 80386 может поместить свои процедуры и прикладной код в кодовой сегмент с уровнем привилегии 0/ или в более чем один такой сегмент, прикладная задача может затем вызвать функцию операционной системы обычной командной вызова. Такой подход является быстрым, но требует от прикладных задач, чтобы в них не было ошибок и чтобы они выполнялись правильно (как это, например, реализуется во встроенных системах). Ничто не запрещает задаче, работающей на уровне привилегии 0, вызывать процедуры, находящиеся в адресе, который не является точкой входа в операционную систему, ничто не запрещает такой задаче испортить данные операционной системы. Для защиты операционной системы прикладные прогарммы и данные могут быть помещены в менее привилегированные сегменты. Также как задача, работающая на данном уровне привилегии, не может читать или писать данные в сегмент с большим уровнем привилегии, задача также не может непосредственно вызвать процедуру из более привилегированного сегмента.

Для того, чтобы позволить задаче, исполняющей команды из менее привилегированного сегмента, сделать вызов защищенной системной процедуры, операционная система должна определить одну или более входных точек. В процессоре 80386 эти входные точки называются шлюзами (см.Рис.3-9).



Счетчик двойных слов относится только к шлюзам

Имеются два типа шлюзов, которые могут быть использованы для реализации входных точек операционной системы: шлюзы ловушек и шлюзы вызовов. Два типа шлюзов, вообще говоря, похожи, однако шлюз вызова позволяет сделать интерфейс операционной системы идентичным с интерфейсом обычной процедуры. Используя шлюзы вызовов, программисты компиляторов и ассемблеров могут использовать общий набор соглашений для вызова любых процедур, оставляя за процессором 80386 заботы о дополнительной обработке, необходимой для изменения уровней привилегии.

Как показано на рис.3-9, шлюз содержит логический адрес входной точки и набор атрибутов. Наиболее важный атрибут - это уровень привилегии шлюза. Уровень привилегии шлюза определяет уровни привилегии, которые могут использовать шлюз, для использования шлюза вызывающая процедура должна быть, как минимум, также привилегирована как шлюз. На рис.3-10 Показан пример. В этой гипотической системе программа пользователя имеет уровень привилегии 3, в то время как операционная система разделена на 2 уровня. Ядро операционной системы работает на уровне привилегии 0, а менее критичные функции операционной системы работают на уровне привилегии 1. (Уровень привилегии 2 не используется). В этой системе пользовательская программа позволяет вызывать сервисные процедуры, но не ядро. В соответствии с этим обеспечивает шлюз для сервисных проце-

дур. Уровень привилегий этого шлюза равен 3, так что программа пользователя может вызывать процедуры через него. Присваивая шлюзу ядра уровень привилегиии 1, операционная система позволяет сервисным процедурам вызывать ядро, но запрещает доступ к программе пользователя, которая менее привилегирована, чем шлюз ядра. Таким образом операционная система может применять шлюзы для аккуратного определения своих точек входа, включая уровни привилегии, необходимые для использования этих точек входа. Для того, чтобы сделать функции операционной системы вызываемыми из всех задач, операционная система, обычно помещает их шлюзы вызовов в глобальную таблицу дескрипторов.

Для осуществления вызова через шлюз ловушки задача использует команду прерывания, для осуществления вызова через шлюз вызова задача исполняет команду обычного межсегментного вызова. Обе команды изменяют уровень привилегии заадчи переходят к стеку, определенному (в TSS задаче) для старшего уровня привилегии. (Определенная система должна иметь свой собственный стек для того, чтобы гарантировать достаточно стековое пространство для работы, нельзя верить прикладным задачам, что они имеют достаточное стековое пространство).

Перед вызовом через стек вызова задача может заслать параметры в свой стек, как она сделала бы это перед вызовом другой процедуры. Процессор 80386 автоматически копирует параметры в привилегированный стек (поле счетчика двойных слов в шлюзе вызова говорит процессору 80386, сколько двойных слов параметров необходимо скопировать). Системы, которые осуществляют вызовы через шлюзы ловушек, могут пересылать параметры в регистрах.

3.6. Прерывания и особые ситуации

Устройства генерируют прерывания, когда они требуют внимания, в то время как команды могут вызвать особые ситуации, если при их использовании возникают особые условия, такие как несуществующая страница. Типичное прерывание или особая ситуация требуют быстрого вмешательства программного драйвера, который отвечает на прерывание или особую ситуацию. После того, как драйвер вернет управление, 80386 возобнавляет исполнение командного потока, который был прерван или который вызвал особую ситуацию. Поскольку прерывания и особые ситуации весьма похожи, процессор 80386 рассматривает их унифицированным способом.

Каждый источник прерывания и каждый тип особой ситуации имеет идентофикационных номер в диапазоне от 0 до 255, процессор 80386 использует этот номер для того, чтобы вызвать обработчик, связанный с прерыванием или особой ситуацией. При возникновении особых ситуаций они распознаются процессором 80386, который определяет номера особых ситуаций, как это показано в табл.3-1. Номера прерываний определяются операционной системой. Операционная система инициализирует программируемый контроллер прерываний 8259а таким образом, что каждый источник прерываний связывается со своим номером. При появлении прерывания 8259а передает процессору 80386 номер прерывания. Команды прерываний указывают свои номера своих операндах. Заметим, что для совместимости с существующим и будущим оборудованием фирмы интел номера прерываний и особых ситуаций от 0 до 71 не должны использоваться иначе, чем это указано в табл.3-1. Все другие номера могут применяться без ограничений.

Таблица 3-1.

Номер	Описание
0	деление на 0
1	особая ситуация отладки
3	контрольная точка программы
4	переполнение
5	нарушение гарниц массива
6	недопустимый код операции
7	сопроцессор отсутствует
8	двойная ошибка
10	неправильный TSS
11	сегментная ошибка
12	переполнение стека сверху или снизу
13	нарушение общей защиты
14	страничная ошибка
16	ошибка сопроцессора

3.6.1. Таблица дескрипторов

Сгенерировав или получив номер прерывания или особой ситуации, процессор 80386 использует его как индекс в таблице дескрипторов прерываний (IDT). IDT Может быть расположена в любом месте памяти, операционная система инициализирует IDT и загружает ее адрес в регистр таблицы дескрипторов прерываний (IDTR). Подобно GDT или LDT, IDT является вектором дескрипторов, хотя шлюзы являются единственным типом дескрипторов, допустимых в IDT. В IDT имеется один шлюз для каждого обработчика прерывания и особой ситуации (IDT функционально подобна таблице прерываний, имеющейся во многих архитектурах).

Обработчик прерываний или особых ситуаций процессора 80386 может быть реализован в виде процедуры или задачи, ниже кратко обсуждаются достоинства этих двух способов. Процессор 80386 вызывает обработчик, организованный в виде процедуры так, он выполняет системный вызов через шлюз. Для вызова обработчика, организованного в виде задачи, процессор 80386 осуществляет переключение задач. Тип шлюза IDT обработчика говорит процессору, каким образом необходимо вызвать обработчик (см. Табл.3-2). Как было указано, шлюзы прерываний и ловушек функционально подобны шлюзам вызовов, за исключением того что они заставляют 80386 запомнить регистр флагов в стек обработчика. Они отличаются один от другого только состоянием флага разрешения прерывания (IF) при входе в обработчик, в обработчик прерываний входят с запрещенными прерываниями, в то время как в обработчик ловушки, который обычно используется для обработки особых ситуаций, входят без изменения запрета. В процессе входа в обработчик - зада^у 80386 загружает регистр флагов значением, запомненным в его TSS - задаче, разрешая обработчику работать с разрешенными или запрещенными прерываниями.

Таблица 3-2.

Характеристики шлюзов прерываний и особых ситуаций

Тип шлюза	Обработчик	Прерывания
прерывание	процедура	запрещены

шлюз задача	процедура задача	разрешены (флаг IF обработчика)

Обработчики - процедуры являются подходящим средством для программ, которые должны работать в контексте (т.е. использовать адресное пространство и значения регистров) задачи, которая прервана или вызвала особую ситуацию. При 16-мгц-синхросигнале процедура входа в обработчик занимает 3,6 мкс. Подобно любой другой процедура прерывания или особой ситуации имеет доступ ко всем ресурсам работающей задачи: к ее данным и коду, ее регистрам и стеку. Так и должно быть для большинства особых ситуаций, поскольку в задаче возникла особая ситуация и может потребоваться доступ к данным задачи, чтобы эту особую ситуацию разрешить. Например, обработчику страни^ных ошибок необходимы страничные таблицы работающей задачи для того, чтобы найти дисковый адрес отсутствующей страницы. В идеале прерывания должны обрабатываться задачами, а не процедурами, так как прерывание , вообще говоря, не относится к задаче, которую оно прерывает. Более того, обработчик прерываний должен иметь свои собственные ресурсы (например, свой собственный стек), а не внедрять в стек какой-то задачи, которая работала в момент появления прерывания. С другой стороны переключение задач требует значительно большего времени чем вызов процедуры (17 мкс вместо 3,6), поскольку процессор запоминает и восстанавливает регистры при переключении задач. Системы, которые чрезвычайно чувствительны ко времени реакции на прерывания, могут обрабатывать прерывания при помощи процедур.

3.6.2. Особые случаи и регистры отладки

Подобно большинству процесоров 80386 имеет команду контрольной точки, которую можно использовать для вызова отладчика. Однако отладочная аппаратура 80386 имеет форму регистров отладки, показанных на рис.3-11. Регистры отладки поддерживают контрольные точки данных являются важным нововведением, которое может сократить время отладки на многие часы, т.К. Например, позволяет установить, когда происходит непредусмотренная запись в структуры данных.

Регистры отладки также сокращают искажения в программах, необходимые для записи команды точек останова в код, особенно для программ, которые защищены от записи или разделены другими задачами.

Отладчик 80386 реализует в виде обработчика особой ситуации с номером 1. Процессор может непосредственно вызвать отладчик после исполнения любой команды (путем установки флага TF - флага ловушки пошагового исполнения), после переключения определенной задачи или после появления условия точки останов, определенного осним из регистров отладки. Анализируя регистр статуса отладки, обработчик отладочной особой ситуации может определить, какая из причин его вызвала.

Будучи вызван при переключении задач, отладчик может перезагрузить регистры регистры отладки значениями, подходящими для новой задачи.

80386 Может ожидать появления одновременно до четырех условий контрольных точек, вызывая обработчик отладочной особой ситуации про появлении одного из условий. Каждое условие контрольной точки определяется содержимым регистра отлдаки, эти регистры могут быть загружены и запомнены при помощи привилегированных форматов команды MOVE. Условие контрольной

точки содержит 32-битный линейный адрес, 2-битное поле длины и поле доступа, последние два элемента указываются в полях регистра управления отладкой DR7. Адрес условия контрольной то^кой и длина формируют адресный диапазон, который контролируется процессором при каждой ссылке на память. Поле доступа определяет тип доступа, для которого процесор может вызвать особую ситуацию 1. Могут быть указаны 3 типа доступа:

1) исполнение команды по указанному адресу;
2) запись данных в диапазон адресов;
3) чтение или запись данных в диапазоне адресов.

3.7. Ввод/вывод

Системы, базирующиеся на 80386, могут распределять устройства ввода/вывода в пространство памяти процессора или в отдельное пространство ввода/вывода. Устройствам ввода/вывода, распределенным в пространство памяти, можно обращаться для чтения или записи, используя такие команды обращения, как MOVE, OR или аналогичные. Устройства, распределенные в памяти, могут быть защищены с помощью стандартного механизма защиты сегмента и страницы процессора 808386.

В дополнении к своему адресному пространству процессор 80386 имеет 64к-байтное адресное пространство ввода/вывода. Устройства, распределенные в это пространство, управляются командами ввода, вывода, ввода строки и вывода строки. Первые две команды передаеют байт, слово, двойное слово в или из цредит аедит EAX-регистра. Последние две команды передают строку байтов, слов или двойных слов в память или из памяти.

Команды ввода-вывода 80386 чувствительны к уровню привилегий. В регистре флагов имеется поле, называемое уровнем привилегий ввода/вывода (IOPL), которое определяет минимальный уровень привилегий , на котором работающая задача может исполнять команды ввода/вывода (IOPL загружается из TSS, так что задачи могут иметь различные IOPL). Например, если IOPL задачи равен 1, то задача не может выдавать команды ввода-вывода, исключая случаи, когда она работает на уровне привилегии 0 или 1. Механизм IOPL поддерживает операционные системы с многоуровневой защитой, в которых, например, критичные и стабильные процедуры ядра работают на уровне привилегии 0, а более изменчивые процедуры ввода-вывода работают на уровне привилегии 1, в этом случае операционная система должна только установить IOPL равным 1, когда она создает задачу, поскольку IOPL характерен для задачи, те задачи, которым можно доверять, могут исполнять команды ввода-вывода на прикладном уровне, позволяя тем самым непосредственно работать со специальными устройствами, для которых не имеется драйвера операционной системы.

Для того, чтобы реализовать прямой доступ к памяти (DMA) от устройств ввода/вывода, операционная система 80386 передает физический адрес контроллеру DMA и должна гарантировать, что сегменты и/или страницы, ко которым происходит обращение при DMA-передачах , не будут преремещаться во время этой операци. Один из способов пометить страницы "фиксированными на время ввода-вывода" - это использование одного из трех битов пользователя в страничной таблице.

4. Архитектурная совместимость

80386 совместим на уровне объектного кода с 80286 и с 8086. Хотя можно просто использовать 80386 как быстрый 80286

или как очень быстрый 8086, их совместимость обладает достаточно большой гибкостью. Процессор 80386 может выполнять программы 80286 и 80386 параллельно, и, используя виртуальный режим 86 процессора 80386, существующие программы 8086 могут также исполняться параллельно. Таким образом при помощи 80386 становится возможным строить системы, которые могут параллельно исполнять программы, написанные для различных поколений семейства микропроцессоров.

4.1. Совместимость с 80286

Архитектура 80286 является точным подмножеством архитектуры 80386. Так как процесор 80386 распознает все команды 80286, регистры, дескрипторы и т.Д., То операционная система 80286 и прикладные программы могут быть перенесены на аналогичное оборудование, построенное на основе 80386 без изменения хотя бы одного бита.

Прямой перенос, упомянутый выше, является наиболее быстрым путем для того, чтобы запустить существующие программы, написанные для 80286 на системе, построенной на 80386. С другой стороны операционные системы для 80386 могут разрабатываться так, чтобы они поддерживали существующие прикладные программы для 80286, и, позволяя в то же время, новым прикладным программам полностью использовать свойства архитектуры 80386 (например, 32-битные параметры и длинные сегменты). В таких гибких разработках новые прикладные программы вызывают операционную систему непосредственно, передавая 32-битные параметры. Возовы старых прикладных программ, которые организованыв 16-битном формате 80286, перехватываются и преоразуются в 32-битный формат, а затем передаются в операционную систему.

4.2. Режимы реального и виртуального 8086

Процессор 80386 может выполнить объектные программы 80386 водном из 2 режимов: реальном режиме или виртуальном режиме 86. 80386 Входит в реальный режим при сбросе. В реальном режиме процессор обеспечивает быстрое исполнение без защиты так, как на 8086. Многие операционные системы будут переключаться из реального режима в режим защиты после инициализации, но также возможно работать все время в реальном режиме 8086. Принципиальное различие между реальным режимом 80386 и действительным режимом 8086 заключается в скорости: программы 8086, которые критичны по скорости исполнения (например, использующие рассчитанные временыые циклы), могут потребовать небольших изменений для того, чтобы они работали правильно в значительно более быстром реальном режиме 80386. Но основное множество программ 8086 будет работать без каких-либо трудностей, т.к. они работают в реальном режиме 80286.

Виртуальный режим 86 устанавливает исполнительную среду 8086 внутри защищенной многозадачной среды 80386. В то время, как реальный режим управляет всем, что делает процессор, виртуальный режим 86 может применяться в избранных задачах 80386. Когда процессор работает в виртуальном режиме 86, он ведет себя, как в 8086, но поле переключения к нормальной задаче, процессор работает как 80386 (который, на самом деле, может интерпретировать программы как для 80286, так и 80386). Таким образом виртуальный режим 86 позволяет операционной системе поддерживать исполнение программ 8086, 80286 и 80386 одновременно.

В разделе 3 было описано, как сегмент состояния задачи отражает состояние своего виртуального процессора. Флаг VM86 в регистре флагов, который загружается из TSS, определяет работающий виртуальный процессор задачи как 8086 или 80386. Когда 80386 загружает свои регистры из TSS, у которого флаг VM86 установлен, процессор входит в виртуальный режим 86. Когда при последующем переключении задачи, процессор загружает значения в регистр из TSS, у которого флаг VM86 сброшен, он выходит из виртуального режима 86. Таким образом от задачи к задаче процессор эмулирует 80386 или 8086 согласно значению флага VM86. 80386 Также выходит из виртуального режима 86, когда он выдает особую ситуацию или возникает прерывание, и предоставляет полные ресурсы архитектуры обработчику прерываний и исобых ситуаций. При возврате из обработчика, вызванного в виртульном режиме 86,80386 автометически возвращется в виртуальный режим 86. Так как адресное пространство 8086 равно 1 мбайту, логические адреса, генерируемые задачей в виртуальном режиме 86, попадает в первый мбайт линейного адресного пространства 80386. Множество задач, работающих в виртуальном режиме 86, могут пересекаться друг с другом, поэтому они все должны разделять младший мбайт линейного адресного пространства. Операционная система может использовать страничный механизм 80386 для перемещения линейных адресных пространств задач, работающих в виртуальном режиме 86, в различные области физического адресного пространства. Используя страничный механизм таким способом, не только предотвращают наложение задач, работающих в виртуальном режиме 86, но позволяют операционной системе с виртуальной памятью обменивать страницы задач, работающих в виртуальном режиме 86, как если бы они были задачами 80386.

Задача, работающая в виртуальном режиме 86, может выполнять программу, которая была написана для выполнения на однозадачном персональном компьюторе. Такая программа может содержать команды, которые потенциально опасны, когда они выполняются в многозадачной среде. Например позволяя задаче, работающей в виртуальном режиме 86, выполнять команду Очистки флага прерываний, тем самым запрещая прерывания, можно остановить всю систему. Для предотвращения таких нарушений 80386 выдает особую ситуацию, когда задача, работающая в виртуальном режиме 86, пытается исполнять команду ввода/вывода или команду, относящуюся к прерываниям.

Запрещение исполнения таких команд защищает остальную часть системы от задач, работающих в виртуальном режиме 86, но не удовлетворяет потребности задач виртуального режима 86 в исполнении команд. Решение заключается в моделировании опасных команд в процедуре операционной системы, называемой монитором виртуальной машины. Когда вызывается обработчик особых ситуаций, он может проверить флаг VM86 в образе регистров флагов в стеке, чтобы определить, является ли источник особой ситуации задачей виртуального режима 86, в этом случае обработчик особых ситуаций может вызвать монитор виртуальной машины, который может промоделировать команду и вернуть управление задаче виртуального режима 86. Следует заметить, что монитор виртуальной машины моделирует только несколько команд 8086, и как моделируемые команды, так и те команды, которые 80386 исполняет непосредственно, выполняются значительно быстрее на 80386, чем на 8086.

Работая совместно, 80386 и монитор виртуальной машины реализует полный набор команд 8086, и страничный механизм может обеспечить каждую задачувиртуального режима 86 своим собственным защищенным адресным пространством. Однако, большинс-

тво задач 8086 требуют дополнительных ресурсов, обеспечиваемых опеарционной системой и периферийным оборудованием. В качестве примера первого типа ресурсов можно привести файловую систему, в качестве второго типа можно привести контроллер растрового дисплея, работающего непосредственно под управлением прикладной программы. Эти ресурсы могут присутствовать в системе, основанной на 80386 в форме, отличной от топ, в которой они присутствовали ранее в системах, для которых была создана программа 8086, чтобы упростить работу для предоставления этих ресурсов в различных средах, 80386 может реализовать ловушки при обращении к операционной системе и к периферийным устройствам, которые делаются задачами виртуального режима 86.

Например, большинство операционных систем 8086 используют команду прерывания для реализации вызовов операционной системы. 80386 Выдает особую ситуацию, когда задача виртуального режима 86 пытается исполнить команду прерываний. Монитор виртуальной машины может затем протранслировать вызов операционной системы в вызов операционной системы 80386, как это показано на рис.4-1. Если IOPL задачи виртуального режима 86 установлен равным значению, меньшему 3, то 80386 будет аналогичным образом создавать ловушки при исполнении всех команд ввода/вывода программой 8086. Страничный механизм 80386 может быть использован для перенаправления обращений к устройствам ввода/вывода, распределенным в память, на другие адреса, если это необходимо. Подобные обращения могут также вызвать особые ситуации путем указания соответствующих страниц памяти, как предназначенных только для чтения (чтобы быть обнаруженным в случае записи) или как отсутствующих (для обнаружения при чтении или записи).

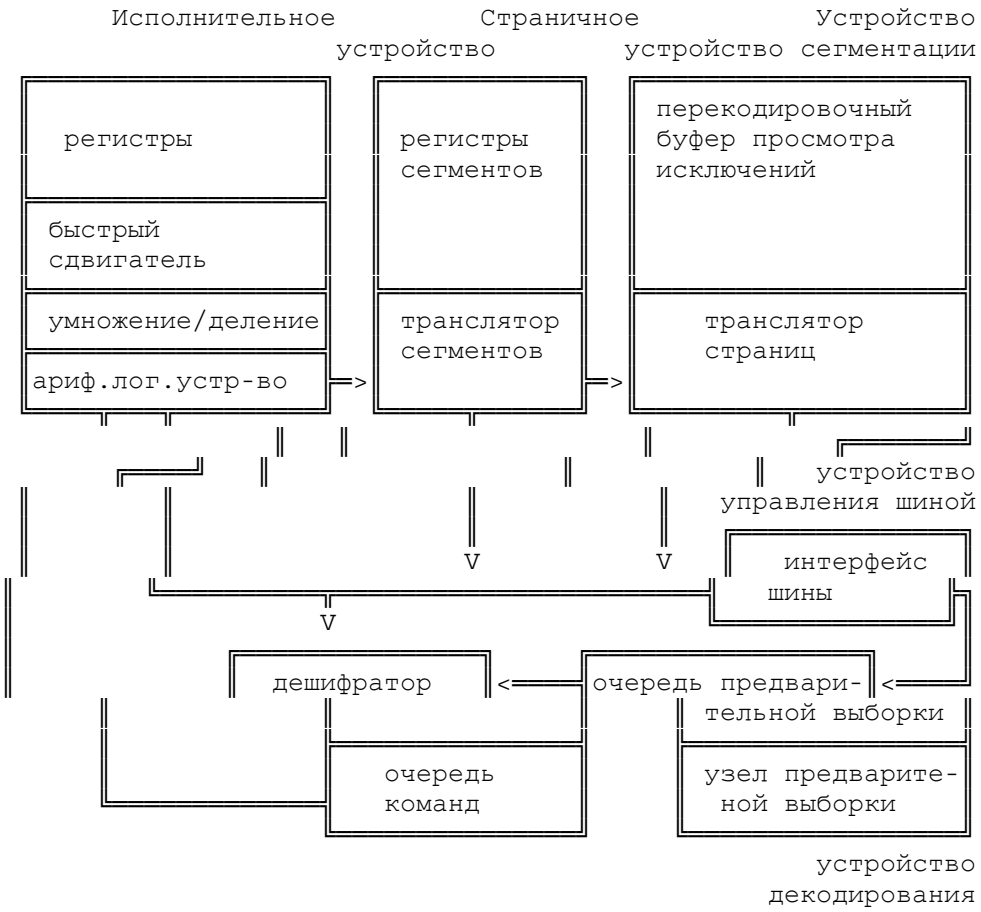
5. Аппаратурная реализация

Архитектура процессора 80386, описанная в предыдущих разделах, реализована более чем в 275000 транзисторах, использующих технологический процесс CHMOS III фирмы ИНТЕЛ. В данном разделе кратко рассматривается внутреннее устройство кристалла 80386 и более подробно сигналы, при помощи которых 80386 взаимодействует с другими компонентами.

5.1. Внутренняя структура

На рис.5.1. приведена обобщенная функциональная структура процессора 80386. Показанные 6 устройств организованы в конвейерную структуру, которая позволяя им рабитать параллельно над различными комнадами или над различными частями одной и то же команды. Устройство управления шиной управляет передачами на шине для других устройств. Если ни одно из других устройств не требуется шиной, устройство предварительной выборки читает следующее двойное слово комнадного потока из памяти в очередь предварительной выборки. Таким образом, большинство чтений байтов команды производится параллельно с исполнением других команд во время свободных циклов шины. Устройство декодирования расшифровывает каждый код операции., Преобразовывает его в указатель на микрокод, которые реализует данную команду. Исполнительное устройство выполняет микроинструкции. Исполнительное устройство может складывать два 32-битные регистра за 2 такта. Аппаратура умножения/деления выполняет 32-битное умножение за время от 9 до 41 такта, в зависимости от каличества значащих цифр, а 32-битное деление

за время от 38 до 42 тактов, в зависимости от того, являются операнды знаковыми или беззнаковыми. Сдвиг, циклический сдвиг и операции над полями битов выполняются при помощи быстрого сдвигателя, который может сдвигать до 64 бит за один такт. В типичной смеси команд, которая включает переходы и вызовы, 80386 исполняет команды со средней скоростью в 4,4 такта каждая.



Конвейерная организация выборки команды, ее декодирование и исполнение на одном кристалле не является типичным для современных микропроцесоров. С другой стороны помещение устройства управления памятью (MMU) на кристалл, содержащий конвейерную структуру, также является нововведением. Включение MMU в процессорный кристалл улучшает производительность, ускоряет трансляцию адреса. За счет уменьшения задержки распространения сигнала (большинство устройств управления памятью, организованных в виде отдельных кристаллов, вводят, как минимум, один такт ожидания на каждое обращение к памяти). Еще одним средством ускорения работы процессора является использование границ полутактов синхросигнала, которые доступны внутри кристалла (частота синхронизации на входе 80386 вдвое выше частоты, используемой в кристалле). Устройство управления памятью 80386 состоит из сегментного устройства и устройства управления страницами, как показано на рис.5-1.

Устройство управления сегментами транслирует логические адреса и проверяет каждый доступ на соответствие атрибутам защиты сегмента. Для большинства команд устройство управления сегментами получает данные для трансляции и защиты из регистров сегментов и дескрипторов, находящихся на кристалле 80386. Устройство управления страницами включается или выключается

программами операционной системы. Когда устройство выключено, линейные адреса, полученные устройством управления, проходят через устройство управления страницами без изменения. Когда страничный механизм включен, устройство управления страницами транслирует линейные адреса в физические адреса и проверяет, что доступ соответствует атрибутам страницы. Устройство управления страницами включает 32-элементный перекадировочный буфер просмотра исключений (TLB), который запоминает необходимую для трансляции информацию для некоторого числа страниц, к которым происходили последние обращения. Используя TLB, устройство управления страницами может транслировать большинство обращений к страницам (обычно 98-99%) без обращения к страничным таблицам, находящимся в памяти. При необходимости устройство управления страницами генерирует циклы шины, необходимые для возврата старых элементов TLB в их страничные таблицы и для загрузки свободных мест в TLB элементами старничных таблиц, обращение к которым имеет место в текущей команде.

5.2. Внешний интерфейс

На рис.5-2. показана блок-схема типичной системы, использующей процессор 80386. Это, фактически рабочее место инженера-проектировщика.

На рис.5-3 показан внешний интерфейс 80386 более подробно, выводы процессора сгруппированы по функциональному назначению. Ниже описаны сигналы, связанные с этими выводами.

5.2.1. Синхросигнал

Первые версии 80386 работают при частоте 12,5 или 16 мгц. Сигнал синхронизации (CLK2) имеет частоту вдвое большую, чем частота кристалла. Генератор синхросигнала 82384 генерирует сигнал CLK2, который делится процессором 80386 на два, чтобы получить свою внутреннюю синхрочастоту.

5.2.2. Шины данных и адреса

Процессор 80386 имеет отдельные 32-битные шины адреса и данных. Для совместимости с существующим оборудованием и драйверами устройств эффективная разрядность шины может динамически переключаться между 16 и 32 битами. Ниже этот вопрос обсуждается более подробно.

Система команд 80386 поддерживает 8-, 16- и 32-битные передачи. Адресная шина организована так, чтобы непосредственно указывать байты данных, которые являются активными в данном цикле шины. Старшие 30 бит каждого адреса поступают на выводы A2-A31. Выводы BE0-BE3 (разрешение байта) указывают, какие байты шины данных относятся к текущей передаче. BE0 Соответствует разрядам D0-D7, BE1 соответствует D8-D15 и т.Д. Эти управляющие байты непосредственно соответствуют способу, которым большинство 32-битных подсистем памяти организованы, и устраняют необходимость в аппаратуре декодирования байтов (см.Рис.5-4). Если, например, необходимо присоединиться к системной шине, которая требует наличия младших битов адреса, A0 и A1 могут быть сгенерированы из BE0-BE3 при помощи 4 клапанов.

Операнды в памяти 80386 не должны быть расположены на каких-нибудь границах, однако производительность повышается, когда они попадают на границы адресов, которые кратны их размеру в байтах. Это значит, что слова наилучшим образом распо-

лагаются на адресах, делящихся на два, а двойное слово - на адресах, делящихся на 4. (Элементы длиннее 32 бит, такие как числа с плавающей запятой с двойной точностью, должны также располагаться на 4-байтных границах для наилучшего быстродействия). 80386 Автоматически генерирует необходимое количество циклов шины для передачи операндов, не располагающихся на оптимальных границах, например, целое, расположенное в двойном слове, запомненное в четном адресе, не делящимся на 4, передается за два 16-битных цикла шины.

5.2.3. Определение циклов шины

80386 информирует внешнее оборудование о том, что на шине начинается нормальный цикл шины путем установки сигнала ADS (статус адреса). В тот же самый момент процессор определяет тип цикла шины с помощью сигналов W/R, D/C и M/IO. Эти сигналы отличают чтение от записи, данные от кодов команд и обращение к вводу/выводу от обращения к памяти, соответственно.

80386 Вырабатывает сигнал LOCK (захват шины) для мультипроцессорных применений и применений с несколькими ведущими устройствами. Сигнал говорит другим ведущим устройствам шины, что процессор выполняет операцию с несколькими циклами шины, которая не должна прерываться. 80386 автоматически выдает LOCK, когда он изменяет дескриптор сегмента и страничные таблицы, во время циклов шины, связанных с процедурой подтверждения, и когда он выполняет команду EXCHANGE. Команда EXCHANGE обеспечивает неделимую операцию "проверка и установка", которая является основным строительным элементом при реализации семафоров в разделяемой памяти. Программисты, работающие на языке ассемблера, могут захватить шину во время исполнения некоторых других команд, если этим командам предшествует префикс LOCK.

5.2.4. Управление циклом шины

По указанию внешнего оборудования 80386 может реализовать два типа циклов шины: неконвейерный и конвейерный. Первый тип цикла обеспечивает 2-тактный доступ к высокоскоростным кэш-памятям и локальным памятям любого объема (эффективность обращений к Кэш-памятям зависит от их размера по отношению к элементам информации, с которыми обращаются к ним прикладные программы). Второй тип цикла шины позволяет низкоскоростным памятям иметь больше времени для ответа на цикл шин, позволяя процессору 80386 в то же время работать с максимальной скоростью. Внешнее оборудование может динамически разрешать и запрещать конвейеризацию шины путем установки сугнала NA (следующий адрес), как описано ниже. Предоставляя возможность динамического управления циклом шины, процессор 80386 позволяет инженеру-разработчику использовать комбинации из различных компонентов (элементов) памяти, которые удовлетворяют критериям стоимости, необходимого объема и требуемой производительности, а также приспосабливать проект для использования перспективных технологий при создании микросхем памяти.

Процессор 80386 выводит тип цикла шины, как описаны выше, а внешнее оборудование сигнализирует, что оно ответило на цикл шины путем установки сигнала READY. Если, как это часто бывает, другой запрос шины ожидает внутри процессора 80386, когда сигнал READY уже установлен, процессор выводит следующий тип цикла шин. Если конвейеризация отключена, минимальное

время между адресами и данными составляет два такта. Внешнее оборудование, которое не может ответить за два такта, может удлинить цикл шины путем удержания сигнала READY в неактивном состоянии, т.е. путем вставления тактов ожидания в цикл. Если 32-битные циклы шины исполняются друг за другом, то максимальная пропускная способность шины 80386 составит 32 мегабайта в секунду при частоте синхросигнала 16 мгц или 25 мегабайт в секунду при частоте 12,5 мгц.

Благодаря внутренней конвейеризации процессор 80386 очень часто знает адрес и тип следующего цикла шины, прежде чем внешнее оборудование ответит на текущий цикл. Внешнее оборудование может использовать свойство внутренней конвейеризации адреса 80386 для получения более раннего доступа к следующему типуцикла шины. Конвейеризация адреса может дать внешнему оборудованию время, равное трем тактам между адресом и данными, в то время как для процессора пропускная способность шины останется равной двум тактам.

Конвейеризация адреса наилучшим образом используется в системах с перемежающейся адресацией, которая может отвечать на доступ в различных блоках памяти параллельно. Устанавливая сигнал NA, внешнее оборудование может запрашивать 80386 выдать тип следующего цикла шины, как только он станет известен внутри процессора, а не ожидать сигнала READY (см.Рис.5-6).

5.2.5. Динамическое управление разрядность шины

В дополнение к управлению типами циклов шины подсистема памяти (и ввода/вывода) также может динамически управлять разрядностью шины данных. Динамическое управление разрядностью шины позволяет:

1) произвольно комбинировать 16- и 32-битные подсистемы памяти, программное обеспечение также может осуществлять 32-битные передачи независимо от того, имеет оно доступ к 16- или к 32-разрядной памяти;

2) достаточно просто подсоединятся к 16-битным шинам, таким как шина MULTIBUS I;

3) реализовывать совместимость с 16-битными периферийными устройствами (и их драйверами), регистры которых обычно располагаются на 16-разрядных, а не на 32-разрядных границах.

Устанавливая сигнал "разрядность шины 16" (BS16), внешнее оборудование может проинструктировать процессор, чтобы он выполнил текущую передачу только на 16 младших битах шины данных. Если сигнал BS16 установлен, а обращение 32-разрядное, процессор 80386 принимает сигнал BS16 позже в цикле шины, позволяя вне[нему оборудованию установить его толцко для соответствующих типов памяти и ввода/вывода.

5.2.6. Статус процессора и управление

Другое ведущее устройство шины (процессор или интелигентное периферийное устройство, такое как пдп-контроллер), может запросить использования локальной шины 80386 путем выставления сигнала HOLD. Процессор подтверждает передачу шины установкой сигнала HLDA (подтверждение захвата) в конце текущего цикла шины (если он имел место), затем он подавляет свой следующий цикл шины до тех пор, пока сигнал HOLD не будет снят. Когда процессор 80386 освобождает шину для другого устройства, он поддерживает сигнал HLDA в активном состоянии, а остальные свои выводы - в высокоимпедансном состоянии, электрически изолируясь от системы.

Прерывания 80386 классифицируются как маскируемые и не

маскируемые, маскируемые прерывания поступают на вход процессора INTR (запрос прерывания), а прерывание второго типа на вход NMI (запрос немаскируемого прерывания). Программы операционной системы могут игнорировать вход INTR путем очистки флага разрешения прерывания. Процессор всегда принимает сигнал на входе NMI, многие системы используют этОт входдля того, чтобы информировать процессор об аварии системы по питанию или глобальной системной ошибке.

Запросы маскируемых прерываний обычно подсоединяются ко входу INTR через один или несколько программируемых контроллеров прерываний 8259a (пкп). Каждый 8259a может обрабатывать до 8 источников прерываний, несколько контроллеров 8259a могут быть каскадированы, чтобы обеспечить прием сигналов от максимум 64 различных источников прерываний. Операционная система инициализирует каждый 8259a при помощи идентифицируемого номера (вектора), обеспечивая тем самым для каждого входа прерываний свою программу обработки. 8259A предоставляет этот номер процессору 80386 в ответ на цикл шины процессора, связанной с подтверждением прерывания. 80386 Использует этот нимер для вызова обработчика, предназначенного для ответа на прерывание.

Установка сигнала RESET ставит процессор в начальное состояние (в реальный режим с запрещенными прерываниями) и заставляет его выбрать команду из физического адреса FFFFFFF04.

5.2.7. Управление сопроцессором

Процессор 80386 посылает команды и операнды в числовой сопрпцессор 80287 или 80387 путем выполнения циклов ввода/вывода шины к резервным адресам выше обычного 64-кбайтного пространства ввода/вывода . Числовой сопроцессор может быть выбран высоким уровнем сигнала на линии A31 при низком сигнале M/I0. 80386 Использует различные протоколы связи для каждого сопроцессора, посылая 16-битные величины в 80287 и 32-битные величины в 80387. Процессор 80386 знает в момент сброса, присутствует ли 80387, программное обеспечение инициализации может проверить наличие сопроцессора 80287.

Сопроцессор устанавливает сигнал BUSY если он выполняет команду. 80386 Не посылает следующую команду сопроцессору до тех пор, пока сигнал BUSY - низкий. Программное обеспечение может синхронизировать процессор 80386 с сопроцессором, выдавая команду WAIT, которая приостанавливает 80386 до тех пор, пока сигнал BUSY остается неактивным. Сопроцессор устанавливает сигнал ERROR, когда он обнаруживает особую ситуацию, которая должна быть обработана операционной системой, в ответ на это 80386 вызывает обработчик особых ситуаций числового сопроцессора, выдавая особую ситуацию 7. Вывод PEREQ используется для реализации сопроцессорного протокола процессора 80386.

6. Сведения о функционировании

6.1 Введение

Характерной чертой 80386 является наличие простого функционального интерфейса для взаимосвязи с внешними модулями.

В 80386 имеются две раздельные шины: шина адреса и шина данных. Шина данных - 32-разрядная и двунаправленная. В большинстве применяемых модулей для высококоростной локальной ши-

ны используются 32 разряда адреса, передаваемого по адресной шине, из них 2 младших разряда дешифрируются в 4 сигнала строба данных (каждый из этих сигналов разрешает или запрещает передачу соответствующего байта данных), а остальные 30 разрядов представляют собой двоичный код адреса. Для управления обменом по шине адреса и шине данных используются соответствующие управляющие сигналы.

Изменяемая ширина (разрядность) шины данных позволяет процессору взаимодействовать как с 32-х, так и с 16-разрядными внешними шинами в синхронном режиме (см. 6.3.4).

Если передача информации состоит из нескольких циклов обмена, каждый из которых требует 16-разрядной ширины шины данных, то все равно 80386 в каждом цикле автоматически выполнит необходимую процедуру установления разрдности шины. *-разрядные периферийные устройства могут быть подключены к 32-х или 16-разрядным шинам, при этом их производительность не снижается. В 80386 применяется новый режим конвейерной адресации, который обеспечивает в случае 32-х и 16-разрядной шин наиболее рациональное использование памяти, особенно это проявляется в очень напряженном режиме работы с ресурсами памяти (когда доступ к памяти требуется большому числу абонентов).

Режим конвейерной адресации по сравнению с другими способами адресации значительно сокращает время нахождения интерфейса памяти в состоянии ожидания (см. 6.4.2). Конвейерную адресацию целесообразно применять в системах, имеющих в своем составе память с расслоением. В системах с рабочей частотой 16 МГц, включающих в себя память с расслоением со временем обращения 100 Нс (динамические ОЗУ), можно совсем исключить состояние ожидания, применяя конвейерную адресацию. Когда внешние модули потребуют режим конвейерной адресации, 80386 сформирует адрес и определит длительность цикла шины для предстоящего цикла шины (если позволяют внутренние ресурсы), даже если в настоящий момент процессор ожидает подтверждение в текущем цикле.

Однако, неконвейеризированный способ адресации идеально подходит для устройств, в состав которых входит кэш-память, так как высокое быстродействие кэш-памяти позволяет работать в неконвейеризированном режиме. Для обеспечения максимальной гибкости системы на основе совмещения циклов конвейерный способ адресации применяется в синхронных системах.

Цикл шины процессора является основным средством передачи информации из системы в процессор или из процессора в систему.

Минимальная длительность цикла передачи данных по шине в 80386 состовляет два периода тактовой частоты. Поскольку процессор 80386 имеет 32-разрядную шину данных и рабочую частоту 16 МГц, то следовательно максимальная пропускная способность 80386 составляет 32 Мбайт/сек. Однако, длительность любого цикла шины может быть увеличена, если в этом цикле задерживается выдача подтверждения обмена от внешнего модуля. В соответствующий момент времени подтверждение выдается путем формирования сигнала на входе READY (ГОТОВ) процессора 80386.

80386 может терять доступ к своим локальным шинам, передавая управление ими другим устройствам, например, каналам прямого доступа к памяти. При этом, за исключением единственного выхода HLDA, формируемого 80386, обеспечивается почти полная изоляция процессора от системы. Изолязия процессора необходима при передаче управления тестовому оборудованию или в отказоустойчивых применениях (в многопроцессорных системах для изоляции отказавшего процессора и замены его другим).

В данном разделе представлены сведения об интерфейсе процессора. Во-первых, описано назначение и функции выводов процессора (см. 6.2 Описание сигналов). Кроме того, описаны изменения сигналов в течение циклов шины (см. 6.3 Механизм обмена по шине, 6.4 Описание функционирования шины и 6.5 Другие сведения по функционированию).

6.2 Описание сигналов

6.2.1 Введение

Подраздел начинается с краткого описания входных и выходных сигналов 80386, объединенных в функциональные группы. Отметим, что наличие символа # после названия сигнала означает, что активное состояние сигнала - состояние низкого уровня. И наоборот, когда такого символа # нет после названия сигнала, то сигнал активен при высоком уровне.

Пример обозначения сигнала: M/IO#

- Высокий уровень означает обращение к памяти.
- Низкий уровень означает обращение к устройствам ввода/вывода.

В описаниях сигналов встречаются иногда обозначения временных параметров, таких как "t25 Reset/Setup Time" (Время сброса при включении питания) и "t26 Reset Hold Time" (Время удержания сигнала сброса). Значения этих параметров приведены в табл.7-4 и табл.7-6.

6.2.2 Синхросигнал (CLK2)

CLK2 обеспечивает основную синхронизацию работы 80386. Эта тактовая частота делится пополам для того, чтобы сформировать внутреннюю процессорную тактовую частоту, используемую при выполнении команд внутри процессора. Внутренний синхросигнал состоит из двух фаз: "фазы один" и " фазы два". Каждый период частоты. На рис.6-2 показано соотношение двух синхросигналов. Если необходимо, фаза внутреннего синхросигнала может быть синхронизирован от такого отрицательного фронта сигнала RESET, который обеспечит заданные времена установки и удержания, t25 и t26 (setup and hold times).

6.2.3 Шина данных (D0-D31)

Двунаправленные с тремя состояниями линии шины данных обеспечивают перемещение данных от 80386 к другим устройствам. Наличие высокого уровня напряжения на входах/выходах шины данных обозначает наличие кодов логической единицы "1" на этих выводах. Шина данных может передавать данные как на 32-, так и на 16-разрядные шины благодаря тому, что есть возможность изменения размера шины данных; размер шины данных определяется значением входного сигнала BS16# (см. параграф 6.2.6 Шина управления).

Для правильного выполнения операций считывания сигналов с шины данных требуется обеспечение необходимых значений времени установки t21 и времени удержания t22 считываемых данных.

При любой операции записи (включая циклы останова и выключения) 80386 всегда передает все 32 разряда данных, даже если в текущем цикле размер шины обмена равен 16 разрядам.

6.2.4 Шина адреса (BE0#-BE3#, A2-A31)

Эти выходы с тремя состояниями обеспечивают физическую адресацию памяти или адресацию устройств ввода/вывода. Шина адреса обеспечивает физическое пространство адресов памяти объемом 4 гигабайта (от 00000000H до FFFFFFFFH) и пространство адресов ввода/вывода объемом 64 килобайта (от 00000000H до 0000FFFFH) для обращения к устройствам ввода/вывода. Для передачи сигалов ввода/вывода, автоматически формируемых для обеспечения взаимодействия 80386 с сопроцессором, используется адресное пространство ввода/вывода от 800000F8H до 800000FFH, так как для обращения к сопроцессору необходимо совпадение двух условий: наличие высокого уровня напряжения на линии адреса A31 и наличие низкого уровня на линии M/IO#.

Значения сигналов стробов данных BE0#-BE3# определяют соответственно те байты 32-разрядной шины данных, которые участвуют в текущей передаче. Это особенно удобно для взаимодействия с внешней аппаратурой.

BE0# определяет участие в обмене разрядов D0-D7,
BE1# -"- D8-D15,
BE2# -"- D16-D23,
BE3# -"- D24-D31.

Количество стробов данных BE0#-BE3#, находящихся в активном состоянии, определяет размер операнда обмена (1,2,3 или 4 байта) (см. параграф 6.3.6 Выравнивание данных).

Когда выполняется цикл записи в память или в устройство ввода/вывода, и передаваемый операнд занимает только старшие 16 разрядов шины данных (D16-D31), копия этого операнда одновременно передается по младшим 16 разрядам шины данных (D0-D15). Это дублирование выполняется для обеспечения оптимального режима записи на 16-разрядные шины. Процедура дублирования записываемых данных зависит от значений стробов данных BE0#-BE3#.

Таблица 6-1.

Зависимость дублирования записываемых данных от значений BE0#-BE3#

Стробы данных 80386				Записываемые данные 80386				Есть ли автоматическое дублирование?
BE3#	BE2#	BE1#	BE0#	D24-D31	D16-D23	D8-D15	D0-D7	
выс.	выс.	выс.	низ.	неопр.	неопр.	неопр.	A	нет
выс.	выс.	низ.	выс.	неопр.	неопр.	B	неопр	нет
выс.	низ.	выс.	выс.	неопр.	C	неопр.	C	да
низ.	выс.	выс.	выс.	D	неопр.	D	неопр	да
выс.	выс.	низ.	низ.	неопр.	неопр.	B	A	нет
выс.	низ.	низ.	выс.	неопр.	C	B	неопр	нет
низ.	низ.	выс.	выс.	D	C	D	C	да
выс.	низ.	низ.	низ.	неопр.	C	B	A	нет
низ.	низ.	низ.	выс.	D	C	B	неопр	нет
низ.	низ.	низ.	низ.	D	C	B	A	нет

Обозначения: D=логические записываемые данные в байте D24-D31;
C= -"- D16-D23;
B= -"- D8-D15;
A= -"- D0-D7.

6.2.5 Сигналы определения типа цикла шины (W/R#, D/C#, M/IO#, LOCK#)

Эти выходы с тремя состояниями определяют тип текущего цикла шины. В зависимости от значения W/R# все циклы подразделяются на циклы записи и циклы чтения. D/C# разделяет все циклы на циклы обмена данными и циклы обмена управляющими сигналами. M/IO# отличает циклы обращения к памяти от циклов обращения к устройствам ввода/вывода. По сигналу LOCK# различаются циклы с блокированной шиной.

Основными сигналами определения типа цикла шины являются W/R#, D/C# и M/IO#, так как эти сиганлы принимают действительное значение одновременно с установлением активного уровня сигнала ADS# (выход строба адреса). Действительное значение сигнала LOCK# устанавливается тогда, когда начинается цикл шины (причем, цикл с конвейерной адресацией) и после установления активного уровня сигнала ADS# (см. параграф 6.4.3.4 конвейерная адресация).

Точное соответствие типов циклов шины значениям сигналов W/R#, D/C# и M/IO# приведено в табл.6-2. Отметим одну комбинацию сигналов W/R#, D/C# и M/IO#, которая никогда не может быть получена при активном уровне сигнала ADS# (однако, эта комбинация, которая обозначена как "запрещенная" может иметь место в нерабочих состояниях шины, при неактивном уровне сигнала ADS#). Поскольку действительные значения сигналов M/IO#, D/C# и W/R# определяются временем действия сигнала ADS#, то в другое время для оптимального использования дешифрирующей схемы можно использовать и запрещенную комбинацию.

Таблица 6-2.

Определение типа цикла шины

M/IO#	D/C#	W/R#	Тип цикла шины	Блокирована ли шина?
низкий	низкий	низкий	Подтверждение прерывания	да
низкий	низкий	высокий	Запрещенная	да
низкий	высокий	низкий	Чтение данных из устройства ввода/вывода	нет
низкий	высокий	высокий	Запись данных в устройство ввода/вывода	нет
высокий	низкий	низкий	Чтение команды из памяти	нет
высокий	низкий	высокий	Останов: Адрес=2 Выключение: Адрес=0	нет

			(BE0#Выс. BE1#Выс. BE2#Низк. BE3#Выс. A2-A31Низк.) (BE0#Низк BE1#Выс. BE2#Выс. BE3#Выс. A2-A31Низк.)	
высокий	высокий	низкий	Чтение данных из памяти	Некоторые циклы
высокий	высокий	высокий	Запись данных в память	Некоторые циклы

6.2.6 Сигналы управления шиной

6.2.6.1 Введение

Нижеперечисленные сигналы позволяют процессору определять начало цикла шины, а также дают возможность другим устройствам системы управлять конвейерной адресацией, размером шины данных и определять конец цикла шины.

6.2.6.2 Строб адреса (ADS#)

Этот входной сигнал с тремя состояниями на входе указывает на то что на выводах 80386 установлены действительные значения сигналов, определяющих тип цикла шины, и сигналов адреса (W/R#, D/C#, M/IO#, BE0# - BE3# и A2-A31). Сигнал ADS устанавливается в течение тактов T1 и T2 состояний шины (дополнительную информацию о состояниях шины см. 6.4.3.2 Неконвейеризированная адресация и 6.4.3.4 Конвейерная адресация).

6.2.6.3 Сигнал подтверждения (READY#).

Этот вход указывает на то, что текущий цикл шины завершен, и те байты, участие которых в цикле обмена определено значениями BE0#-BE3# и BE16#, приняты или переданы. Когда в течение цикла чтения или цикла подтверждения прерывания формируется активный уровень сигнала READY#, 80386 "защелкивает" входные данные и завершает цикл. Когда сигнал READY# формируется в цикле записи, процессор завершает цикл шины.

Сигнал READY# игнорируется в первом такте всех циклов шины, затем в каждом такте состояние READY# опрашивается до тех пор, пока не установится активный уровень сигнала READY#. READY# должен быть сформирован для подтверждения в каждом цикле шины, включая циклы отображения останова и отображения выключения. Для правильной работы время установки t19 и время удержания t20 сформированного сигнала READY# должны иметь определенные необходимые значения (См. все параграфы раздела 6.4 Описание функционирования шины).

6.2.6.4 Запрос следующего адреса (NA#)

Этот сигнал используется для запрашивания адреса в режиме конвейерной адресации. Этот вход сообщает процессору о том, что система готова принять из 80386 новые значения сигналов BE0#-BE3#, A2-A31, W/R#, D/C# и M/IO#, даже если завер-

шение текущего цикла не подтверждено сигналом READY#.

Если 80386 обнаружит на входе NA# активный уровень, он выдает на шину следующий адрес, обеспечив внутреннюю подготовку к следующему запросу шины (см. параграф 6.4.2 Конвейерная адресация и 6.4.3 Циклы чтения и записи).

6.2.6.5 Указатель 16-разрядной шины (BS16#)

Благодаря сигналу BS16# осуществляется непосредственная связь 80386 с 32-разрядной и 16-разрядной шинами данных. Установка активного уровня этого входа приведет к тому, что в текущем цикле шины обмен будет производиться только по младшей половине шины данных (D0-D15) в соответствии со значениями сигналдов BE0# и BE1#. Дополнительное влияние сигнала BS16# (установленного в активное состояние) не проявится, если в текущем цикле сформированы активные уровни только сигналов BE0# или BE3#, действие сиганала BS16# (активного уровня) заставит процессор 80386 выполнить необходимые переключения для правильной передачи старшего(их) байта(ов) по линиям D0-D16.

Если операнд занимает обе половины шины данных и BS16# в активном состоянии, то 80386 автоматически выполнит второй 16-разрядный цикл шины. Для правильной работы время установки t17 и время удержания t18 сигнала BS16# должны иметь определенные необходимые значения.

Циклы ввода/вывода, автоматически выполняемые 80386 для взаимосвязи с сопроцессором, не требуют установки сигнала BS16#. Сопроцессоры типа 80287 и 80387 анализируют входной сигнал ERROR# сразу после отрицательного фронта сигнала RESET. 80386 обменивается только 16-разрядными посылками с 80287, а в случае взаимодействия 80386 с сопроцессором 80387 обмен производится только 32-разрядными посылками. Таким образом, значение BS16# влияет на циклы с участием 80287, а в течение циклов с участием 80387 сигнал BS16# должен поддерживаться на неактивном уровне.

6.2.7 Сигналы арбитража шины

6.2.7.1 Введение

В этом разделе описывается механизм, благодаря которому процессор передает управление своими локальными шинами другим активным абонентам, запрашивающим урпавление шиной (см. 6.6.1).

6.2.7.2 Запросы на захват шины (HOLD)

Этот выход указывает на то, что каким-то устройствам кроме 80386 требуется управление шиной.

Сигнал HOLD должен поддерживаться в активном состоянии в течение всего времени, пока любое другое устройство является владельцем локальной шины. Сигнал HOLD игнорируется во время действия RESET. Если сигнал RESET появится во время действия сигнала HOLD, то более приоритетный сигнал RESET установит шину в нерабочее состояние быстрее, чем установится состояние подтверждения захвата шины (состояние высокого сопротивления).

Процессор срабатывает по фронту сигнала HOLD и, пока HOLD поддерживается в активном состоянии, постоянно анализирует уровень этого сигнала. Вход HOLD - синхронизированный. Для правильной работы время установки t23 и удержания t24

должны всегда иметь определенное необходимое значение.

6.2.7.3 Подтверждение захвата шины (HLDA)

Формирование активного уровня на выходе HLDA указывает на то, что 80386 передает управление своей локальной шиной в ответ на установление сигнала HOLD и переходит в состояние подтверждения захвата шины.

Состояние подтверждения захвата предполагает почти полную изоляцию процессора. Сигнал HLDA в этом состояние - единственный сигнал, выдаваемый 80386. Другие выходные сигналы или двунаправленные сигналы (D0-D31, BE0#-BE3#, A2-A31, W/R#, D/C#, M/IO#, LOCK# и ADS#) переключаются в третье (высокоимпедансное) состояние, поэтому запросившее шину устройство может захватить их. К некоторым сигнальным линиям желательно подсоединить фиксирующие резисторы для того, чтобы избежить ложное срабатывание по этим сигналам, когда они не формируются текущим владельцем шины (см. 7.2.3 Рекомендации по выбору и применению резисторов). Кроме того, один используемый фронт, который может быть сформирован на входе NMI во время состояния подтверждения захвата, запоминается с тем, чтобы он был проанализирован и обработан после снятия сигнала HOLD. Кроме обычного использования состояния подтверждения захвата при взаимодействии 80386 с контроллерами ПДП (прямого доступа к памяти) или активными периферийными устройствами, состояние почти полной изоляции процессора особенно удобно использовать в режиме тестирования системы, когда тестовое оборудование управляет системой, а также в отказоустойчивых системах.

6.2.8 Сигналы интерфейса с сопроцессором

6.2.8.1 Введение

В следующих параграфах этого раздела дано описание сигналов, предназначенных для интерфейса с арифметическим сопроцессором. Эти сигналы, дополняя сигналы шины данных, шины адреса и сигналы определения типа цикла шины, управляют взаимодействием 80386 с его сопроцессором 80287 или 80387.

6.2.8.2 Запрос сопроцессора (PEREQ)

Активный уровень этого сигнала указывает на то, что сопроцессор требует, чтобы в ответ на его запрос операнды данных были переданы в/из памяти.

Активный уровень этого входного сигнала указывает на запрос сопроцессора на передачу процессором 80386 операнда данных в/из памяти, в ответ на этот запрос 80386 передает информацию между сопроцессором и памятью. Поскольку в 80386 хранится код операции, выполняемой сопроцессором, 80386 осуществляет запрошенную передачу данных в заданном нарпавлении и по заданному адресу памяти. 80386 анализирует и срабатывает по уровню сигнала PEREQ. Сигнал PEREQ может быть асинхронным по отношению к CLK2.

6.2.8.3 Сопроцессор занят (BUSY#)

Активный уровень этого сигнала указывает на то, что сопроцессор еще выполняет заданную текущую инструкцию и пока не может принять другую инструкцию.

Когда 80386 встречает любую инструкцию сопроцессора, которая оперирует с арифметическим стеком (стеком сопроцессора)

(т.е. инструкции загрузки, "POP" - инструкции (убрать в стек) или арифметические операции), или инструкцию ожидания WAIT, он сразу автоматически анализирует состояние входного сигнала BUSY# и будет просматривать его до тех пор, пока BUSY# не переключится в неактивное состояние.

Такой просмотр входного сигнала BUSY# предотвращает преждевременную выдачу следующей инструкции во время выполнения сопроцессором предыдущей инструкции.

Инструкция сопроцессора FNINIT и FNCLEX могут быть выполнены даже при наличии активного уровня на входе BUSY#, так эти инструкции используются для инициализации и прерывания-сброса сопроцессора.

80386 анализирует и срабатывает по уровню сигнала BUSY#.

Сигнал BUSY# может быть асинхронным по отношению к CLK2.

Сигнал BUSY# служит еще одной цели. Если во время отрицательного фронта сигнала RESET на входе BUSY# имеется сигнал низкого уровня, то 80386 выполнит процедуру самодиагностирования (см. 6.6.3 Функционирование шины в течение и после действия сигнала RESET). Если же в этот момент сигнал BUSY# будет иметь высокий уровень, то самодиагностирование выполняться не будет.

6.2.8.4 Ошибка сопроцессора (ERROR#)

Этот входной сигнал указывает на то, что при выполнении сопроцессором предыдущей инструкции им был сформирован код ошибки, немаскируемый управляющим регистром сопроцессора. При выполнении сопроцессором инструкции процессор 80386 автоматически анализирует входной сигнал ERROR#, и если установится активный уровень сигнала ERROR#, то 80386 вырабатывает прерывание 7, чтобы обратиться к программам обработки ошибок.

Некоторые инструкции сопроцессора, в основном те, которые сбрасывают флаги арифметических ошибок в сопроцессоре или сохраняют состояние сопроцессора, исполняются без выработки процессором 80386 прерывания 7, даже если установлено активное состояние сигнала ERROR#. К таким инструкциям относятся FNINIT, FNCLEX, FSTSW, FSTSWAX, FSTCW, FSTENV, FSAVE, FESTENV и FESAVE.

80386 анализирует и срабатывает по уровню сигнала ERROR#. Сигнал ERROR# может быть асинхронным по отношению к CLK2.

Сигнал ERROR# выполняет еще одну функцию. Если низкий уровень сигнала ERROR# установится не позже, чем через 20 периодов тактовой частоты CLK2 после отрицательного фронта сигнала RESET, и сохранится таким по меньшей мере до тех пор, пока 80386 не начнет свой первый цикл шины, то это указывает на то, что в системе используется сопроцессор типа 80387 (разряд ET в регистре CR0 автоматически устанавливается в 1). В обратном случае в системе используется сопроцессор типа 80287 или не используется никакой (разряд ET в регистре CR0 автоматически устанавливается в 0). См. 6.6.3 Функционирование шины в течение и после действия сигнала RESET. Изменение сигнала на выходе ERROR# влияет на установку только бита ET. Программно устанавливаются необходимые значения битов EM и MP в регистре CR0. Следовательно, для различения случая наличия в системе сопроцессора типа 80287 от случая, когда в системе вообще нет сопроцессора, необходимо программное задание соответствующего значения бита EM в регистре CR0 (единичное значение бита EM устанавливается в случае, когда в системе нет сопроцессора). Если анализ изменения состояния сигнала ERROR# показал наличие в системе 80387 (сигнал ERROR установлен в

низкое состояние после сброса), но позднее программно установлено единичное состояние бита EM (EM=1), то 80386 ведет себя так, как если бы в системе не было сопроцессора.

6.2.9 Сигналы прерывания

6.2.9.1 Введение

В этом разделе описываются входные сигналы, которые могут прерывать или приостанавливать выполнение процессором текущего набора инструкций.

6.2.9.2 Маскируемый запрос прерывания (INTR)

Активный уровень этого входного сигнала обозначает запрос на обслуживание прерывания, которое может быть замаскировано битом IF флагового регистра Flag Register 80386. В ответ на входной сигнал INTR 80386 выполняет два цикла подтверждения прерывания и в конце второго цикла "защелкивает" 8-битовый вектор прерывания, принятый по линиям D0-D7, чтобы идентифицировать источник прерывания. 80386 анализирует уровень и срабатывает по уровню сигнала INTR. Сигнал INTR может быть асинхронным по отношению к CLK2. Для того, чтобы гарантировать опознание процессором маскируемого запроса прерывания, активный уровень сигнала INTR должен поддерживаться до начала первого цикла подтверждения прерывания.

6.2.9.3 Немаскируемый запрос прерывания (NMI)

Этот входной сигнал определяет запрос на обслуживание прерывания, которое не может быть программно замаскировано. Запрос немаскируемого прерывания всегда обрабатывается по программе, адрес начала которой указан в элементе (позиции) 2 таблицы прерываний. Когда обрабатывается NMI, то благодаря фиксированному значению позиции таблицы прерываний, соответствующей NMI, циклы подтверждения прерывания не выполняются.

80386 анализирует и срабатывает по положительному фронту сигнала NMI. Сигнал NMI может быть асинхронным по отношению к сигналу CLK2. Чтобы гарантировать опознание сигнала NMI, последний должен иметь неактивный уровень по меньшей мере в течение 8-и периодов CLK2, и затем должен быть установлен и поддерживаться активный уровень сигнала NMI по меньшей мере в течение 8-и периодов CLK2.

Как только начинается обработка запроса прерывания NMI, другие запросы NMI обрабатываться не будут до появления очередной инструкции IRET, которая означает конец процедуры обслуживания прерывания NMI. Однако, если все-таки раньше этого времени снова будет сформирован активный уровень сигнала NMI, то один положительный фронт сигнала NMI будет запомнен для последующей обработки после выполнения очередной инструкции IRET.

6.2.9.4 Сигнал сброса (установки в исходное состояние) (RESET)

Этот входной сигнал останавливает выполнение любой операции и переводит 80386 в состояние, известное как состояние сброса. Сброс 80386 производится установкой активного уровня сигнала RESET в течение 15-и или более периодов CLK2 (за 78 или более периодов CLK2 до запроса самодиагностирования). Когда установлен активный уровень сигнала RESET, сигналы на

всех остальных входных выводах игнорируются, а шинные выводы переводятся в нерабочее состояние как показано в табл.5-3. Если одновременно установлены активные уровни сигналов RESET и HOLD, то более приоритетным будет сигнал RESET. Сброс по сигналу RESET будет произведен, даже если 80386 находился в состоянии подтверждения захвата до установки RESET.

80386 анализирует и срабатывает по уровню (активному или неактивному) сигнала RESET. Сигнал RESET может быть асинхронным по отношению с CLK2. Если необходимо, фаза внутреннего синхросигнала процессора, а также целое состояние 80386 могут быть полностью синхронизированы с внешними схемами, если обеспечить необходимые для этого значения времени установки t25 и времени удержания t26 отрицательного фронта сигнала RESET.

Таблица 6-3.

Состояние выводов (неработающей шины) в течение действия сигнала RESET

Обозначение выводов	Уровни сигналов во время сброса RESET
ADS#	Высокий
D0-D31	Третье состояние (высокий импеданс)
BE0#-BE3#	Низкий
A2-A31	Высокий
W/R#	Высокий
D/C#	Высокий
M/IO#	Низкий
LOCK#	Высокий
HLDA	Низкий

6.2.10 Список сигналов

В табл.6-4 перечислены сигналы процессора 80386 и приведены некоторые их характеристики.

Таблица 6-4.

Перечень сигналов 80386

Название сигнала	Функции сигнала	Активный уровень	Вход/выход	Вход синхронный или асинхронный по отношению к CLK2	Переключается ли выход в третье (высокоимпедансное) состояние во время действия HLDA?

CLK2	Синхросигнал	-	Вход	-	-
D0-D31	Шина данных	Высокий	Вход/ выход	S	да
E0#-BE3#	Стробы данных	Низкий	Выход	-	да
A2-A31	Шина адреса	Высокий	Выход	-	да
W/R#	Указатель режима записи или чтения	Высокий	Выход	-	да
D/C#	Указатель обмена данными или упр. сигналами	Высокий	Выход	-	да
M/IO#	Указатель обращения к памяти или В/В	Высокий	Выход	-	да
LOCK#	Блокировка шины	Низкий	Выход	-	да
ADS#	Строб адреса	Низкий	Выход	-	да
NA#	Запрос следующего адреса	Низкий	Вход	S	да
BS16#	16-разрядная ширина шины	Низкий	Вход	S	да
READY#	Передача подтверждения	Низкий	Вход	S	да
HOLD	Запрос на захват шины	Высокий	Вход	S	да
HLDA	Подтверждение захвата шины	Высокий	Выход	S	нет
PEREQ	Запрос сопроцессора	Высокий	Вход	A	нет
BUSY#	Сопроцессор занят	Низкий	Вход	A	нет
ERROR#	Ошибка сопроцессора	Низкий	Вход	A	нет
INTR	Маскируемый запрос прерывания	Высокий	Вход	A	нет
NMI	Немаскируемый запрос прерывания	Высокий	Вход	A	нет

RESET	Сброс	Высокий	Вход	A(примечание	нет

Примечание: Если фаза внутреннего синхросигнала процессора должна быть синхронизирована с внешними схемами, то необходимо обеспечить определенные значения времени установки t25 и времени удержания t26 отрицательного фронта сигнала RESET.

6.3 Механизм обмена по шине

6.3.1 Введение

Все передачи данных занимают один или более циклов шины. Операнды логических данных длиной в байт, слово и двойное слово могут быть переданы без выравнивания физических адресов. Любой байт может быть краевым байтом (первым или последним) передаваемого операнда, но при этом для передачи невыровненного операнда может потребоваться два или даже три физических цикла шины. (См.6.3.4 Изменяемый размер шины данных и 6.3.6 Невыровненные операнды.)

Сигналы адреса 80386 предусмотрены для упрощения аппаратуры внешней системы. Старшие биты адреса реализованы линиями A2-A31. Младшие биты адреса в виде BE0#-BE3# обеспечивают выборку соответственно четырех байтов 32-битной шины данных. Благодаря этому физический операнд представляется в каждом цикле в наиболее удобной форме.

Активные уровни выходных сигналов стробов данных BE0#-BE3# устанавливаются, если соответствующие им байты шины данных будут принимать участие в предстоящем цикле шины, как указано в табл.6-6. Изменяя комбинацию установленных стробов данных можно осуществить любой вариант выборки смежных байтов, но никакая комбинация BE0#-BE3# не позволит выбрать байты, разделенные двумя или тремя нефункционирующими байтами.

Адресные биты A0-A1 физического адреса операнда могут быть образованы, когда необходимо (например, для интерфейсов Multibus I и Multibus II), как функция установленных стробов данных. Соответствие значений A0 и A1 сигналам BE0#-BE3# приведено в табл.6-6. Логические схемы формирования A0 и A1 приведены на рис.6-3.

Таблица 6-6.

Стробы данных и соответствующие им байты данных и операндов

Стробы данных	Соответствующие сигналы шины данных
BE0#	D0-D7 (байт 0 - младший байт)
BE1#	D8-D15 (байт 1)
BE2#	D16-D23 (байт 2)
BE3#	D24-D31 (байт 3 - старший байт)

Таблица 6-6.

Формирование шины A0-A31 из шины
BE0#-BE3# и A2-A31

Адресные сигналы 80386								
A31 ..... A2					BE3#	BE2#	BE1#	BE0#
	Физический адрес							
A31	.....	A2	A1	A0				
A31	.....	A2	0	0	X	X	X	Низкий
A31	.....	A2	0	1	X	X	Низкий	Высокий
A31	.....	A2	1	0	X	Низкий	Высокий	Высокий
A31	.....	A2	1	1	Низкий	Высокий	Высокий	Высокий

Рис.6-3. Логические схемы формирования A0, A1 как функций сигналов BE0#-BE3#

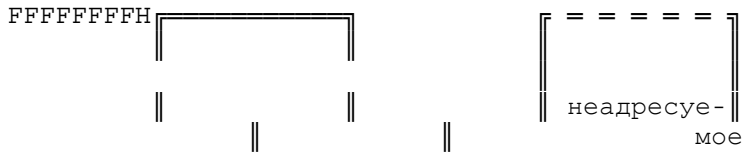
K - MAP for A1 Signal - карта Карно для сигнала A1.

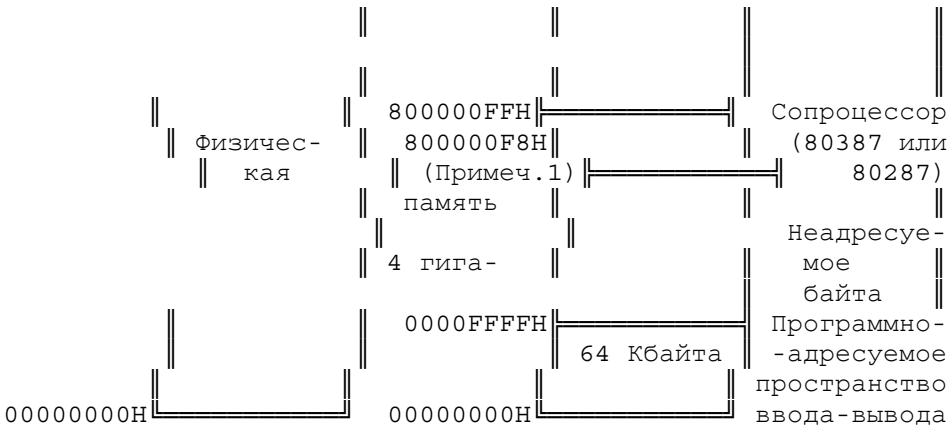
Каждый цикл шины включает в себя по меньшей мере два состояния шины. Каждое состояние шины занимает по времени один период тактовой частоты процессора. Простейший цикл шины может быть дополнен состояниями шины, которые называются состояниями ожидания. См. 6.4 Описание функционирования шины.

Поскольку для выполнение цикла шины требуется как минимум два состояния шины (что равняется двум периодам тактовой частоты процессора), то максимальная скорость передачи данных между внешними устройствами и 80386 равна одному 4-байтовому двойному слову в каждые два периода тактовой частоты процессора, что соответствует максимальной пропускной способности шины 32 мегабайт/сек (80386-16 работает на тактовой частоте 16 МГц).

6.3.2 Пространства памяти и ввода/вывода

В течение циклов шины возможно обращение к пространству памяти или к пространству ввода/вывода. Периферийные устройства в системе могут быть отнесены либо к пространству памяти, либо к пространству ввода/вывода, или и к тому и к другому пространствам. Как показано на рис.6-4, физические адреса памяти находятся в диапазоне от 00000000H до FFFFFFFFH (4 гигабайта), а адреса ввода/вывода - в диапазоне от 00000000H до 0000FFFFH (64 килобайта), необходимом для адресации устройств ввода/вывода. Отметим адреса ввода/вывода, используемые в автоматически выполняемых для взаимосвязи с сопроцессором циклах ввода/вывода. Эти адреса от 800000F8H до 800000FFH не входят в диапазон вышеуказанных адресов для программной адресации устройств ввода/вывода и позволяют легко сформировать сигнал выборки сопроцессора, используя сигналы A31 и M/IO#.





Пространство физической памяти

Пространство ввода/вывода

Примечание: Так как в течение автоматически выполняемых циклов взаимосвязи с сопроцессором устанавливается высокий уровень сигнала A31, то установка единичного уровня A31 и нулевого уровня строба формирования сигала выборки сопроцессора.

Рис.6-4 Пространства физической памяти и ввода/вывода

6.3.3 Организация памяти и ввода/вывода

Ширина магистрали данных от 80386 к пространствам памяти и ввода/вывода может составлять 32 бита или 16 бит. В случае 32-разрядной ширины магистрали пространства памяти и ввода/вывода организованы соответственно как массивы физических 32-разрядных двойных слов. Каждое двойное слово памяти или ввода/вывода состоит из 4-х индивидуально адресуемых (с помощью последовательных адресов байтов) байтов. Самый меньший (из четырех) адрес байта относится к сигналам D0-D7; самый больший - к сигналам D24-D31.

80386 имеет такой сигнал управления шиной, как BS16#, который обеспечивает правильную взаимосвязь с 16-разрядными пространствами памяти и ввода/вывода, организованными в виде последовательности 16-битных слов. Циклы обмена с 16-разрядными и 32-разрядными устройствами памяти или ввода/вывода могут встречаться в любой последовательности, так как состояние сигнала BS16# анализируется в течение каждого цикла шины. См. 6.3.4 Изменяемый размер шины данных. Сигналы стробов данных BE0#-BE3# позволяют обращаться к отдельным байтам при любой структуре памяти или ввода/вывода (32-разрядной или 16-разрядной).

6.3.4 Изменяемый размер шины данных

Изменяемый размер шины данных - отличительная особенность 80386, обеспечиваютщая непосредственную связь процессора с 32-разрядными или 16-разрядными шинами данных памяти или ввода/вывода. Один процессор может быть соединен с шинами двух размеров. Передачи в/из 32- или 16-разрядные порты сопровождаются определением в каждом цикле шины необходимой ширины шины. В течение каждого цикла шины схема дешифрации ад-

реса или подчиненное устройство сами могут установить активный уровень сигнала BS16# для 16-разрядного порта, или неактивный уровень BS16# для 32-разрядного порта.

Когда установлен активный уровень сигнала BS16#, процессор автоматически вместо одной передачи разрядностью больше 16 бит или одной 16-разрядной невыровненной передачи выполнит две или три передачи, как потребуется. При активном уровне BS16# все передачи операндов осуществляются только по линиям D0-D16. Поэтому 16-разрядные устройства памяти или ввода/вывода обмениваются только сигналами данных D0-D16. Специальных переключателей не требуется. Действие активного уровня сигнала BS16# проявляется только тогда, когда в текущем цикле шины установлены активные уровни сигналов BE2# и/или BE3#. Если в передаче участвуют только линии D0-D15, то установка активного уровня BS16# не будет иметь значение, так как передача будет производиться все равно по 16-разрядной шине независимо от состояния BS16#. Другими словами, установка активного уровня BS16# необязательна, когда только младшая половина разрядов шины участвует в текущем цикле.

Существуют две ситуации, при которых проявляется влияние активного уровня BS16# на действия процессора, зависящие таже от значений стробов данных BE0#-BE3# в текущем цикле шины:

- в обмене участвует только старшая половина линий шины: устанавливаются активные уровни только сигналов BE2# и/или BE3#;

- в обмене участвуют и старшая, и младшая половины линий шины: устанавливаются активные уровни по меньшей мере сигналов BE1# и BE2# (и возможно также сигналов BE0# и/или BE3#).

Воздействие BS16# на циклы чтения "с учетом только старшей половины линий шины":

Установка активного уровня BS16# в течение циклов чтения "с участием только старшей половины линий шины" вынудит 80386 считывать младшие 16 битов шины данных и игнорировать данные на старших 16 битах шины данных. Т.е. вместо считывания данных с линий D16-D31 в соответствии с установленными BE2# и BE3# будут считываться данные с линий D0-D16.

Взаимодействие сигнала BS16# на циклы записи "с участием только старшей половины линий шины":

Установка активного уровня BS16# в течение циклов записи "с участием только старшей половины" не отразится на процедуре записи. Когда в цикле записи установлены активные уровни сигналов BE2# и/или BE3#, 80386 всегда копирует сигналы данных D16-D31 на линии D0-D15 (см. табл. 6-1). Поэтому не требуется дополнительных действий 80386 для того, чтобы выполнить эти циклы записи по 32- или 16-разрядной шине.

Воздействие сигнала BS16# на циклы чтения "с участием и старшей и младшей половин шины":

Установка активного уровня сигнала BS16# в течение циклов чтения "с участием и старшей и младшей половин шины" заставит процессор выполнить два 16-разрядных цикла чтения для передачи всего физического операнда. Байты 0 и 1 (в соответствии с установленными BE0# и BE1#) будут считаны в первом цикле с линий D0-D16. Байты 2 и 3 (в соответствии с установленными BE2# и BE3#) будут считаны во втором цикле и снова с линий D0-D16. Сигналы на линиях D16-D31 игнорируются в течение обоих 16-разрядных циклов. BE0# и BE1# всегда находятся в неактивном состоянии в течение второго 16-разрядного цикла.

Активный уровень сигнала BS16# необязательно устанавливать на время второго 16-разрядного цикла. См. рис.6-14, циклы 2 и 2а.

Воздействие сигнала BS16# (активного уровня) на циклы записи "с участием и старшей и младшей половин шины":

Установка активного уровня сигнала BS16# в течение циклов записи "с участием и старшей и младшей половин шины" заставит процесор 80386 выполнять два 16-разрядных цикла записи для передачи целого физического операнда. Наличие всех байтов операнда на линиях D0-D15 в течение первого цикла записи позволит внешним устройствам получить байты 0 и 1 (в соответствии с установленными значениями BE0# и BE1#) по линиям D0-D16. Во втором цикле 80386 скопирует байты 2 и 3 на линии D0-D15, и запись этих байтов (в соответствии с установленными значениями BE2# и BE3#) будет произведена также по линиям D0-D16. Сигналы BE0# и BE1# всегда переключаются в неактивное состояние в течение второго 16-разрядного цикла. Установка активного уровня сигнала BS16# в течение второго 16-разрядного цикла необязательна. См. рис.6-14, циклы 1 и 1a.

6.3.5 Связь с 32- и 16-разрядным устройствами памяти

В 32-разрядных устройствах физической памяти, таких как на рис.6-5, каждое физическое двойное слово начинается с байта, адрес которого кратен 4. Сигналы A2-A31 обычно используются для выборки определенного двойного слова, а сигналы BE0#-BE3# - для выборки определенного байта в двойном слове. BS16# поддерживается в неактивном состоянии во всех циклах шины, оперирующих с 32-разрядным массивом.

Когда в состав системы входят 16-разрядные физические массивы, как показано на рис.6-6, адрес начала каждого 16-битного физического слова кратен 2. Отметим, что схема дешифрации адреса ADDRESS DECODER при дешифрации адреса формирует активный уровень сигнала BS16# только в течение циклов шины, оперирующих с 16-разрядными устройствами памяти, то схема дешифрации адреса анализирует также значения сигналов BE0#-BE3# и W/R#, чтобы определить, когда должен быть установлен активный уровень сигнала BS16#. См. 6.4.3.7 Оптимальное использование метода конвейерной адресации в случае 16-разрядной ширины шины.

Сигналы A2-A31 обычно используются для адресации 32-разрядных и 16-разрязных устройств. Для адресации 16-разрядных устройств необходимы также сигнал A1 и два сигнала строба данных.

Чтобы сформировать необходимые значения сигнала A1 и двух сигналов стоба данных для обращения к 16-разрядному устройству, сигналы BE0#-BE3# должен быть дешифрированы в соответствии с табл. 6-7. Отметим некоторые запрещенные комбинации BE0#-BE3#, никогда не вырабатываемые 80386. При наличии запрещенной комбинации BE0#-BE3# на входе дешифратора выходы его не анализируются и состояние их обозначается X. Запрещенные комбинации BE0#-BE3# могут быть использованы при необходимости для более оптимального использования дешифратора.

Таблица 6-7.

Формирование сигналов A1, BHE# и BLE# для адресации 16-разрядных устройств

Сигналы 80386				Сигналы 16-разрядной шины			Комментарии
BE3#	BE2#	BE1#	BE0#	A1	BHE#	BLE# (A0)	

H*	H*	H*	H*	X	X	X	X - нет ни одного активного байта
H	H	H	L	L	H	L	
H	H	L	H	L	L	H	
H	H	L	L	L	L	L	
H	L	H	H	H	H	L	
H*	L*	H*	L*	X	X	X	X - несмежные байты
H	L	L	H	L	L	H	
H	L	L	L	L	L	L	
L	H	H	H	H	L	H	
L*	H*	H*	L*	X	X	X	X - несмежные байты
L*	H*	L*	H*	X	X	X	X - несмежные байты
L	L	H	H	H	L	L	
L*	L*	H*	L*	X	X	X	X - несмежные байты
L	L	L	H	L	L	H	
L	L	L	L	L	L	L	

BLE# устанавливается (активный уровень) когда активизируются разряды D0-D7 16-разрядной шины.
BHE# устанавливается (активный уровень) когда активизируются разряды D8-D15 16-разрядной шины.
A1 имеет низкий уровень для всех четных слов; A1 имеет высокий уровень для всех нечетных слов.

Обозначения:
X - допустим и высокий и низкий логический уровень;
H - высокий логический уровень;
L - низкий логический уровень;
* - неиспользуемые комбинации BE0#-BE3#:
- комбинация, когда все стробы данных находятся в неактивном состоянии;
- комбинации стробов данных, при которых появляются несмежные активные байты.

6.3.6 Выравнивание операндов

Благодаря гибкой адресации памяти в 80386 возможна передача логического операнда, разрядность которого больше слова или двойного слова памяти или ввода/вывода, например 32-разрядного операнда (двойное слово), адрес начала которого кратен 4, или 16-разрядного операнда (слово), разделенного

между двумя физическими двойными словами массива памяти.

Когда передача операнда требует выполнения нескольких циклов, то во время этих циклов выполняется выравнивание данных и определение размера шины. Таблица 6-8 описывает определение типов циклов передачи для всех комбинаций таких характеристик, как длина логического операнда, выравнивание и ширина шины данных. Когда для передачи многобайтового логического операнда требуется несколько циклов шины, то первыми передаются старшие байты (но если установлен активный уровень BS16#, то будут выполнены два 16-разрядных цикла, причем первыми будут переданы младшие байты).

Таблица 6-8.

Циклы передачи байтов, слов и двойных слов

	Длина логического операнда в байтах								
	1	2				4			
Адрес байта физической памяти (младшие два байта)	XX	00	01	10	11	00	01	10	11
Циклы передачи по 32-разрядной шине данных	b	W	W	W	hb, lb	d	hb, l3	hw, lw	h3, lb
Циклы передачи по 16-разрядной шине данных	b	W	lb,* hb*	W	hb,* lb*	lw,* hw*	hb, lb,* mw*	hw, lw	mw, hb,* lb

Обозначения: b = передача байта
w = передача слова
l = младшая часть операнда
m = средняя часть операнда
x = не используется
* = активный уровень BS16# вызывает выполнение второго цикла шины
3 = передача 3-х байтов
d = передача двойного слова
h = старшая часть операнда

6.4 Описание функционирования шины

6.4.1 Введение

80386 имеет отдельные параллельные шины: шину адреса и шину данных. Шина данных - 32-разрядная и двунаправленная. Ширина шины адреса - 32 разряда: из них 30 старших разрядов - адрес операнда и 2 разряда формируются из 4-х сигналов стробов данных каждый из которых служит для выборки соответствующего байта в операнде. Эти шины анализируются и управляются соответствующими им управляющими сигналами.

Тип каждого цикла шины определяется тремя сигналами: M/IO#, W/R# и D/C#. Одновременно с этими сигналами устанавливается достоверный адрес на линиях BE0#-BE3# и A2-A31. Сигнал строба адреса указывает на выдачу процессором 80386 нового типа цикла шины и адреса.

Объединенные шина адреса, шина данных и все связанные с ними управляющие сигналы называются в тексте просто "шиной".

В рабочем состоянии шина выполняет один из нижеперечисленных циклов шины:

1) чтение из памяти;
2) чтение из памяти с блокировкой шины;
3) запись в память;
4) запись в память с блокировкой шины;
5) чтение из устройства ввода/вывода (или из сопроцессора);
6) запись в устройство ввода/вывода (или в сопроцессор);
7) подтверждение прерывания;
8) цикл останова или цикл выключения.

Табл. 6-2 показывает соответствие комбинаций сигналов определения типа шины каждому типу шины. См. параграф 6.2.5 Сигналы определения типа цикла шины.

Отличительной чертой шины данных является ее изменяемая ширина, которая может быть 32-разрядной и 16-разрядной. Ширина шины данных указывается процессору 80386 его входным сигналом BS16#. Все функции шины могут быть выполнены при любой ширине шины.

Когда шина 80386 не выполняет ни один из вышеперечисленных циклов, она находится или в нерабочем состоянии или в состояние подтверждения захвата шины, последнее может быть вызвано внешней схемой.

Нерабочее состояние шины может иметь место, когда 80386 не выдает дальнейших подтверждений на свой выход строба адреса (ADS#) после начала текущего цикла, и потому текущий цикл будет последним. Состояние подтверждения захвата шины идентифицируется установкой процессором 80386 активного уровня на своем выходе подтверждения захвата (HLDA).

Самой короткой временной единицей деятельности шины является состояние шины. Деятельность состояния шины составляет один период тактовой частоты процессора (два периода CLK2).

Законченная передача данных осуществляется в течение цикла шины, состоящего из двух или более состояний шины.

Самый короткий цикл шины 80386 состоит из двух состояний шины. Состояния шины в каждом цикле обозначены как T1 и T2. В течение такого цикла шины (из 2-х состояний) может быть выполнено обращение по любому адресу памяти или ввода/вывода, если внешняя аппаратура обладает достаточным быстродействием. Высокая пропускная способность шины и цикл шины, занимающий два периода тактовой частоты, наиболее полно реализуют возможности быстрой основной памяти или кэш-памяти.

Каждый цикл шины длится до тех пор, пока не придет подтверждение от внешних устройств системы, использующих для этой цели вход 80386 READY#. Если подтверждение цикла шины будет сформировано в конце первого из состояний T2, то это определит выполнение самого короткого цикла шины, состоящего всего из двух состояний T1 и T2. Однако, если активный уровень сигнала READY# не будет установлен сразу (в конце первого T2), то состояния T2 будут неограничено повторятся до тех пор, пока на входе READY# процессор не обнаружит активный уровень.

6.4.2 Конвейерная адресация

Режим конвейерной адресации обеспечивает определенные протоколы цикла шины.

Протокол конвейерной или неконвейерной адресации выбирается на основе совмещения циклов с использованием входа следующего адреса (NA#).

В режиме неконвейерной адресации текущий адрес и тип цикла шины остаются постоянными в течение всего цикла шины.

В режиме конвейерной адресации адрес (BE0#-BE3#, A2-A31) и тип цикла для следующего цикла устанавливаются и выдаются еще до окончания текущего цикла. Чтобы сигнализировать об их готовности, 80386 устанавливает также активный уровень на выходе строба адреса (ADS#). Рис.6-9 иллюстрирует самые быстрые циклы чтения в режиме конвейерной адресации. Из рис.6-9 следует, что самые короткие циклы шины, использующие метод конвейерной адресации, состоят всего из двух состояний шины, обозначенных T1P и T2P. Следовательно, циклы с конвейерной адресацией обеспечивают такую же пропускную способность данных, как и циклы с неконвейерной адресацией, но время выборки адреса увеличивается по сравнению с неконвейерными циклами.

Из-за увеличения времени выборки адреса режим конвейерной адресации сокращает требуемое количество состояний ожидания. Например, если в режиме конвейерной адресации требуется одно состояние ожидания, то в режиме конвейерной адресации может не потребоваться ни одного состояния ожидания.

Режим конвейерной адресации испольуеется в системах, имеющих адресные "защелки". В таких системах, сразу "защелкивающих" адрес, конвейерная выдача следующего адреса позволяет декодирующей схеме заранее сформировать сигналы включения микросхем (и другие необходимые сигналы выборки), поэтому обращение к выбранным устройствам осуществляется сразу, как только начинается следующий цикл. Другими словами, время декодирования для следующего цикла может частично перекрываться с окончанием текущего цикла.

Если в состав системы входит память с расслоением, имеющая 2 или более банков, то метод конвейерной адресации возможно обеспечит даже большее перекрытие циклов. Вышесказанное действительно, когда контроллер памяти с расслоением устроен так, чтобы позволить начать следующую операцию с памятью в одном банке памяти в то время, как текущий цикл шины еще оперирует с другим банком памяти. Рис.6-10 показывает основную структуру взаимосвязи 80386 с 2-банковой и 4-банковой памятью с расслоением. Отметим, что каждый банк памяти с расслоением имеет шину данных полной ширины (обычно разрядность данных составляет 32 бита, если не задается 16-разрядная ширина шины).

Дополнительные сведения о режиме крнвейерной адресации даны в параграфах 6.4.3.4 Конвейерная адресация, 6.4.3.5 Инициализация и поддержание режима конвейерной адресации, 6.4.3.6 Конвейерный адрес при изменении ширины шины и 6.4.3.7 Оптимальное использование конвейерного адреса в случае 16-разрядной ширины шины.

6.4.3 Циклы чтения и записи

6.4.3.1 Введение

Передачи данных осуществляется посредством выполнения циклов шины, которые подразделяются на циклы чтения и циклы записи. При выполнении циклов чтения данные передаются от внешнего устройства в процессор. При выполнении циклов записи данные передаются в обратном направлении: от процессора к внешнему устройству.

Два варианта адресации попеременно избираются: неконвейерная адресация или конвейерная. После нерабочего состояния шины процессор всегда работает в режиме неконвейерной адресации. Однако, может быть установлен активный уровень входного

сигнала NA# (следующий адрес), избирающий режим конвейерной адресации для следующего цикла шины. Когда выбран режим конвейерной адресации, и в процессоре имеется ожидающий обслуживания внутренний запрос шины, достоверные адрес и тип цикла для следующего цикла шины будут выданы даже до получения подтверждения текущего цикла шины на входе READY#. В каждом цикле шины 80386 обязательно анализирует состояние сигнала на входе NA#, чтобы определить способ адресации, необходимый для следующего цикла.

Попеременно избираются два варианта размера физической шины данных: 32 бита или 16 битов. Обязательно ближе к концу цикла шины состояние входного сигнала BS16# (размер шины 16) анализируется с целью установления размера физической шины данных, необходимого в текущем цикле. Высокий уровень сигнала BS16# указывает на 32-разрядный размер, активный уровень (низкий) BS16# указывает на 16-разрядный размер. Если указан 16-разрядный размер шины, то 80386 автоматически реагирует на это соответствующим образом, чтобы завершить передачу по 16-разрядной шине данных. В зависимости от размера и расположения операнда может потребоваться второй 16-разрядный цикл шины. Подробно об этом см. табл. 6-7. Когда необходимо, 80386 выполняет дополнительный 16-разрядный цикл шины, используя линии D0-D15 для передачи разрядов D16-D31.

Для завершения цикла чтения или цикла записи, также как и любого другого цикла шины, требуется подтверждение цикла, устанавливаемое на входе READY#. До получения подтверждения процессор вводит в цикл шины состояние ожидания, чтобы соответствовать быстродействию внешнего устройства. Внешнее устройство, распознавшее свой адрес и декодировавшее тип цикла шины, формирует в соответствующий момент активный уровень сигнала READY#.

Сигнал READY# анализируется во втором состоянии цикла шины. Если в это же время внешняя аппаратура подтверждает цикл шины установкой активного уровня READY#, то цикл шины завершается, как показано на рис.6-11.

Если во втором состоянии шины сигнал READY# остается в неактивном состоянии, как показано на рис.6-12, цикл шины дополняется еще одним состоянием (состоянием ожидания), и сигнал READY# будет снова анализироваться в конце каждого такого состояния ожидания. Так будет продолжиться неограниченно до тех пор, пока цикл не получит подтверждения по линии READY#.

Когда процессор получает подтверждение текущего цикла, он завершает его. Когда подтверждается цикл чтения, 80386 "защелкивает" информацию, сформированную к этому времени на выводах шины данных процессора. Когда подтверждается цикл записи, 80386 поддерживает достоверное значение записываемых данных в течение первой фазы следующего цикла шины, чтобы обеспечить необходимое значение времени удержания записываемых данных.

6.4.3.2 Неконвейерная адресация

Любой цикл шины может быть выполнен в режиме неконвейерной адресации. Для примера, на рис.6-11 показана последовательность циклов чтения и записи в режиме неконвейерной адресации. Из рис.6-11 следует, что самые короткие циклы, возможные в режиме неконвейерной адресации, состоят каждый из двух состояний шины. Состояния обозначены как T1 и T2. В первой фазе состояния T1 выдаются достоверные значения сигналов адреса и сигналов определения типа цикла шины, и одновременно устанавливается активный уровень сигнала строба адреса

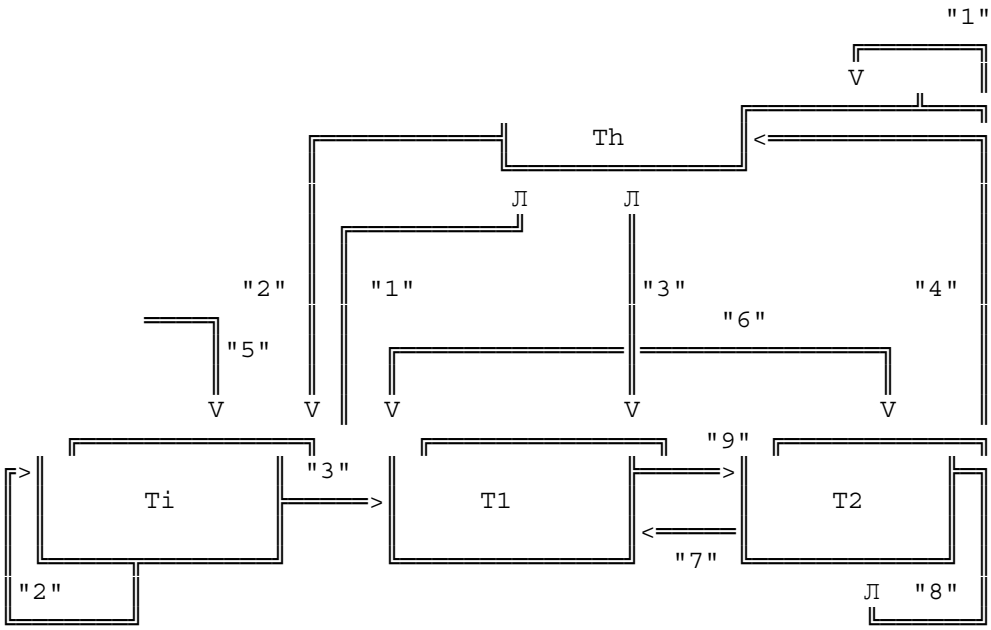
(ADS#), сигнализирующий о готовности вышеперечисленных сигналов.

В течение циклов чтения или записи шина данных фкнкционирует как описано ниже. В цикле чтения 80386 переключает свою шину данных таким образом, чтобы принять сигналы данных от адресованного внешнего устройства. В цикле записи сигналы данных передаются процессором 80386, начиная со второй фазы состояния T1 и кончая первой фазой состояния шины, которое будет установлено сразу после получения подтверждения цикла.

Рис.6-12 иллюстрирует циклы шины в режиме неконвейерной адресации, причем циклы 2 и 3 дополнены одним состоянием ожидания. В циклах 2 и 3 сигнал READY# оказался неустановленным в активное состояние T2. В этих циклах активный уровень сигнала READY# устанавливается в конце второго из состояний T2.

Когда не используется конвейерная адресация, достоверные значения адреса и типа цикла шины сохраняются в течение всех состояний ожидания. Когда цикл дополняется состояниями ожидания, и необходимо обеспечить режим неконвейерной адресации, неактивный (высокий) уровень сигнала NA# должен устанавливаться в течение каждого из состояний T2, исключая самое последнее состояние T2 цикла, как показано на рис.6-12 в циклах 2 и 3. Если активный уровень NA# окажется установленным в состояние T2 (но не в последнем T2), то следующим состоянием, вместо состояния T2 (для неконвейерной адресации), будет состояние T2i (для конвейерной адресации) или T2p (для конвейерной адресации).

Рис.6-13 наиболее полно иллюстрирует картину состояний шины и переходов из состояния в состояние для случая, когда конвейерная адресация не используется. Показанные переходы шины между 4-мя возможными состояниями: T1, T2, Ti и Th. Циклы шины содержат состояния T1 и T2, причем T2 может повторяться (состояния ожидания). Кроме этого, шина может находиться в нерабочем состоянии, т.е. в состоянии Ti, или в состоянии подтверждения захвата шины, т.е. в состоянии Th.



"1"-HOLD активизирован

"2"-HOLD неактивизирован * нет внутреннего запроса
"3"-HOLD неактивизирован * есть внутренний запрос, ожидающий обслуживание
"4"-READY# активизирован * HOLD активизирован
"5"-RESET активизирован
"6"-READY# активизирован * HOLD неактивизирован * нет внутреннего запроса
"7"-READY# активизирован * HOLD неактивизирован * есть внутренний запрос, ожидающий обслуживание
"8"-READY# неактивизирован * NA# неактивизирован
"9"-постоянно

Состояния шины :

T1-первое состояние неконвейерного цикла шины (80386 выдает новый адрес и устанавливает активный уровень ADS#).

T2-последующие состояния цикла шины, когда при просмотре сигнала NA# в текущем цикле шины он оказывается в неактивном состоянии.

Ti-нерабочее состояние.

Th-состояние подтверждения захвата шины (80386 устанавливает активный уровень HLDA).

Самый короткий цикл шины состоит из двух состояний: T1 и T2. Четыре основные состояния шины описывают функционирование шины, когда не используется конвейерная адресация. Эти состояния распространяются на оба размера шины: 32 бита и 16 бит, т.е. справедливы для любого значения BS16#. Если при активном уровне сигнала BS16# требуется выполнение второго 16-разрядного цикла, последний выполняется перед формированием процессором сигнала подтверждения захвата шины.

Рис.6-13. Состояния шины 80386 (когда конвейерная адресация не применяется)

Для случая, когда конвейерная адресация не используется, диаграмма состояния шины такая, как показана на рис.6-13. В нерабочем состоянии шина находится в Ti. Циклы шины всегда начинаются с T1. T1 всегда предшествует состоянию T2. Если цикл шины не подтвержден в течение T2 и уровень NA# при этом неактивный, состояние T2 повторяется. Когда цикл подтвержден в течение T2, то за этим последует состояние T1 следующего цикла шины, если имеется ожидающий обслуживания внутренний запрос шины (запрос на захват шины самим процессором), или состояние Ti, если такого запроса нет, или состояние Th, если установлен активный уровень входного сигнала HOLD.

Диаграмма состояния шины на рис.6-13 справедлива при любом значении сигнала BS16#. Если 80386 выполнит внутренние переключения, необходимые для установки 16-разрядного размера шины, то эти переключения не повлияют на состояния внешней шины. Если для выполнения передачи по 16-разрядной шине требуется дополнительный 16-разрядный цикл шины, он также будет выполняться в соответствии с переходами состояний, показанными на рис.6-13.

В режиме конвейерной адресации в 80386 могут иметь еще три типа состояния шины, не показанные на рис.6-13. На рис.6-20 в параграфе 6.4.3.4 "Конвейерная адресация" показана

более подробная диаграмма состояния шины, включающая циклы в режиме конвейерной адресации.

Состояния шины:

T1 - первое состояние неконвейерного цикла шины (80386 выдает новый адрес и устанавливает активный уровень ADS#);

T2 - последующие состояния цикла шины, когда при просмотре сигнала NA# в текущем цикле шины он оказывается в неактивном состоянии;

Ti - нерабочее состояние;

Th - состояние подтверждения захвата шины (80386 устанавливает активный уровень HLDA).

Самый короткий цикл шины состоит из двух состояний: T1 и T2.

Четыре основные состояния шины описывают функционирование шины, когда не используется конвейерная адресация. Эти состояния распространяются на оба размера шины: 32 бит и 16 бит, т.е. справедливы для любого значения BS16#. Если при активном уровне сигнала BS16# требуется выполнение второго 16-разрядного цикла, последний выполняется перед формированием процессором сигнала подтверждения захвата шины.

6.4.3.3 Режим неконвейерной адресации при изменении размера шины данных

Ширина физической магистрали данных для любого неконвейерного цикла шины может составлять или 32 разряда, или 16 разрядов. В начале цикла шины процессор ведет себя так, как если бы ширина шины данных составляла 32 бита. Когда цикл шины подтверждается установкой активного уровня сигнала READY# в конце состояния T2, то анализируемый в этот момент уровень сигнала BS16# окажется в неактивном состоянии, то размер физической шины данных принимается равным 32 разрядам. Если же наблюдается активный уровень BS16#, то размер шины принимается равным 16 разрядам.

Когда установлен активный уровень BS16#, и для выполнения одной передачи требуется два 16-разрядных цикла, то активный уровень сигнала BS16# должен быть установлен и во втором цикле. Иначе 16-разрядный размер шины не будет сохранен во втором цикле. Также, как и любой другой цикл шины, второй 16-разрядный цикл должен быть подтвержден установкой активного уровня READY#.

Когда требуется второй 16-разрядный цикл для выполнения одной передачи по 16-разрядной шине, то адреса, формируемые для двух 16-разрядных циклов шины, тесно взаимосвязаны. Эти адреса идентичны за исключением разрядов BE0# и BE1#, которые всегда переключаются в неактивное состояние (высокий уровень) во втором цикле, так как сигналы данных D0-D15 были уже переданы в первом 16-битном цикле.

На рис.6-14 и 6-15 показаны передачи, при которых установка активного уровня BS16# требует второго 16-битного цикла для выполнения передачи всего операнда. Рис.6-14 иллюстрирует циклы без состояний ожидания. Рис.6-15 иллюстрирует циклы с одним состоянием ожидания. Отметим, что в цикле 1 на рис.6-15, в течение которого устанавливается активный уровень сигнала BS16#, сигнал NA# должен быть обязательно переключен в неактивный уровень в состоянии(ях) T2, предшествующем(их) последнему состоянию T2. Это необходимо для того, чтобы в финальном состоянии T2 процессор воспринял установленный активный уровень BS16# в режиме неконвейерной адресации.

6.4.3.4 Конвейерная адресация

В режиме конвейерной адресации адрес и тип цикла для следующего цикла шины, который будет обслуживать ждущий обработки,внутреннии запрос процессора, запрашиваются еще до того, как будет получено подтверждение текущего цикла по линии READY#. Когда следующий адрес подготовлен и выдан, 80386 устанавливает активный уровень сигнала ADS#. Протокол режима конвейерной адресации строится на основе совмещения циклов и с помощью входного сигнала NA#.

Когда выполняется цикл шины и текущий адрес должен иметь достоверное значение в течение по меньшей мере одного полного состояния шины, значение входного сигнала NA# анализируется в конце каждой первой фазы состояния до тех пор, пока цикл шины не получит подтверждение. В течение неконвейерных циклов шины, следовательно, NA# анализируется в конце первой фазы в каждом состоянии T2. Примером может служить Цикл 2 на рис.6-16, в течение которого NA# анализируется в конце первой фазы каждого T2 (NA# был установлен в активный уровень один раз в течение первого состояния T2 и не оказывает дальнейшего воздействия на выполнение этого цикла шины).

Если процессор при просмотре NA# обнаружит активный уровень этого сигнала, то 80386 освобождается, чтобы выдать адрес и тип цикла следующего цикла, и установить активный уровень сигнала ADS#, как только в процессоре появится ожидающий обслуживания внутренний запрос шины. Процессор может выдать следующий адрес уже в следующем состоянии шины, независимо от того получил ли в этот момент подтверждение текущий цикл или не получил.

Что касается режима конвейерной адресации, то в этом режиме 80386 имеет следующие особенности:

1) для того, чтобы процессор воспринял активный уровень сигнала NA#, сигнал BS16# должен быть переключен в неактивный уровень на время просмотра сигнала NA# (см. рис.6-16 Циклы 3 и 4; рис.6-17 Циклы 2-4);

В том случае, если сигналы NA# и BS16# окажутся оба активными в течение последнего периода T2 цикла шины, приоритетом будет обладать активный сигнал BS16#. Следовательно, если оба сигнала активны, то текущий размер шины принимается равным 16 разрядам, а следующий адрес будет неконвейерным. Схематично рис.6-18 показывает внутреннюю логику 80386, обеспечивающую эти особенности.

2) следующий адрес может появиться в состоянии шины, следующем сразу после момента обнаружения активного уровня NA# (см. рис.6-16 или 6-17);

В этом случае сразу шина перейдет в состояние T2p. Однако, если в этот момент отсутствует ожидающий обслуживания внутренний запрос шины, то следующий адрес не будет установлен сразу после активизации NA#, и вместо состояния T2p шина перейдет в состояние T2i (см. рис.6-19 Цикл 3). При условии, что текущий цикл шины еще не получил подтверждение по линии READY#, шина перейдет в состояние T2p как только 80386 выдает следующий адрес. Внешние устройства, поэтому, должны следить за состоянием выходного сигнала ADS#, подтверждающим выдачу на шину достоверного следующего адреса.

3) в том случае, когда 80386 при просмотре обнаружит активный уровень NA#, 80386 сам принимает решение на обслуживание самого приоритетного внутреннего запроса шины, ожидающего обслуживания. Процессор не сможет больше выполнить другую 16-разрядную передачу по тому же адресу, даже если BS16# будет установлен в активный уровень внешним устройством, так как после восприятия процессором активного сигнала NA# теку-

щий размер шины принимается равным 32-м разрядам;

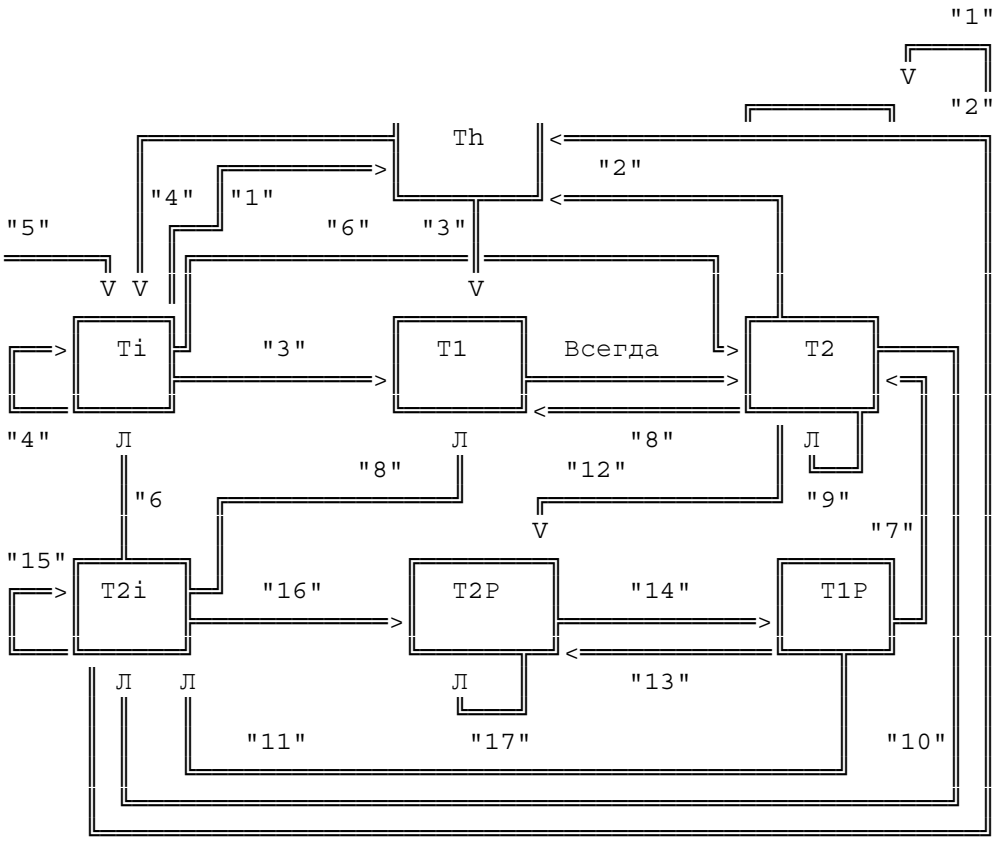
Следовательно, если процессор опознал активный сигнал NA# в течение цикла шины, то после этого сигнал BS16# игнорируется в этом цикле шины (см. рис.6-16, 6-17, 6-19). Таким образом, нельзя активизировать сигнал NA# в тех циклах шины, в которых 16-разрядный размер шины должен быть задан установкой активного уровня сигнала BS16#. См. 6.4.3.6 "Изменение размера шины в режиме конвейерной адресации".

4) любой адрес, достоверное значение которого подтверждено выходным импульсным сигналом 80386 ADS#, будет сохраняться на адресных выводах в течение по меньшей мере двух периодов тактовой частоты процессора. 80386 не может выдавать новый адрес чаще, чем каждые два периода тактовой частоты процессора (см. рис.6-16, 6-17, 6-19);

5) из всех сигналов, необходимых для следующего цикла шины, процессор выдает только адрес и тип цикла шины;

Уровень совмещения в режиме конвейерной адресации не более, чем один цикл шины (см. рис.6-19 Цикл 1).

Полная диаграмма переходов состояний шины, включая функционирование в режиме конвейерной адресации, дана на рис.6-20. Отметим, что эта диаграмма включает диаграмму, справедливую только для режима неконвейерной адресации, и дополнительные три состояния шины для режима конвейерной адресации.



"1"-HOLD активизирован

"2"-READY# активизирован * HOLD активизирован
"3"-HOLD неактивизирован * есть внутренний запрос, ожидающий обслуживание
"4"-HOLD неактивизирован * нет внутреннего запроса
"5"-RESET активизирован
"6"-READY# активизирован * HOLD неактивизирован * нет внутреннего запроса
"7"-NA# неактивизирован
"8"-READY# активизирован * HOLD неактивизирован * есть внутренний запрос, ожидающий обслуживание
"9"-READY# неактивизирован * NA# неактивизирован
"10"-(нет внутреннего запроса * HOLD активизирован) * NA# активизирован * READY# неактивизирован
"11"-NA# активизирован * (HOLD активизирован * нет внутреннего запроса)
"12"-READY# неактивизирован * NA# активизирован * HOLD неактивизирован * есть внутренний запрос, ожидающий обслуживание
"13"-NA# активизирован * HOLD неактивизирован * есть внутренний запрос, ожидающий обслуживание
"14"-READY# активизирован
"15"-READY# неактивизирован * (нет внутреннего запроса * HOLD активизирован)
"16"-READY# неактивизирован * есть запрос, ожидающий обслуживание * HOLD неактивизирован
"17"-READY# неактивизирован

Состояния шины :

T1-первое состояние неконвейерного цикла шины (80386 выдает новый адрес и устанавливает активный уровень ADS#).

T2-последующие состояния цикла шины, когда при просмотре сигнала NA# в текущем цикле шины он оказывается в неактивном состоянии.

T2i-последующие состояния цикла шины, имеющие место когда обнаружен активный уровень NA# в текущем цикле шины, но еще отсутствует внутренний запрос шины, ожидающий обслуживание (80386 не выдаст новый адрес или не установит активный уровень ADS#).

T2P-последующие состояния цикла шины, имеющие место когда в текущем цикле шины обнаружен активный уровень NA#, и имеется внутренний, ожидающий обслуживание, запрос шины (80386 выдаст новый адрес и активизирует сигнал ADS#).

T1P-первое состояние конвейерного цикла шины.

Ti-нерабочее состояние.

Th-состояние подтверждения захвата шины (80386 устанавливает активный уровень HLDA).

Установка активного уровня NA# для конвейерной адресации может повлечь за собой одно из трех типов состояний шины: T2i,T2P и T1P.

В режиме конвейерной адресации самый короткий цикл шины состоит из состояний T1P и T2P.

Рис.6-20. Диаграмма всех возможных состояний шины (включая режим конвейерной адресации)

Самый короткий цикл шины в режиме конвейерной адресации состоит только из двух состояний шины, T1p и T2p (напомним, что для режима неконвейерной адресации такими состояниями являются состояния T1 и T2). T1p является первым состоянием шины конвейерного цикла.

6.4.3.5 Инициализация и поддержание режима конвейерной адресации

Пользуясь диаграммой состояний на рис.6-20, проследим переходы шины из нерабочего состояния Ti, в начало конвейерного цикла шины T1p. Цикл шины, первый после нерабочего состояния шины Ti, должен начинаться с состояния T1, следовательно этот цикл будет неконвейерным. Однако, если будет установлен активный уровень NA#, и первый цикл шины закончится в состоянии T2p, то следующий цикл шины будет конвейерным (адрес для следующего цикла шины выдается в состоянии T2p первого цикла). Кратчайший путь от нерабочего состояния к циклу шины с конвейерной адресацией показан ниже:

Ti,Ti,Ti	T1-T2-T2p	T1p-T2p
нерабочее состояние	неконвейерный цикл	конвейерный цикл

T1-T2-T2p - состояние цикла шины, в течение корого устанавливается адрес (конвейерный) для следующего цикла шины, начинающегося с состояния T1p. Переход к конвейерному циклу из состояния подтверждения захвата шины осуществляется аналогично и показан ниже:

Th,Th,Th	T1-T2-T2p	T1p*T2p
состояние подтверждения	неконвейерный цикл	конвейерный цикл захвата

Переход к ковейерной адресации показан на рис.6-17 Цикл 1. Цикл 1 используется для перехода в режим конвейерной адресации для выполнения последовательности конвейерных циклов 2, 3 и 4. В соответствующий момент устанавливается активный уровень NA# для того, чтобы выбрать конвейерный адрес для циклов 2, 3 и 4.

Когда выполняется цикл шины, и достоверное значение текущего адреса удерживается в течение одного состояния шины, состояние входа NA# анализируется в конце каждой первой фазы до тех пор, пока этот цикл не получит подтверждение. Следовательно, в Цикле 1 на рис.6-17 процессор начинает анализ NA# в состоянии T2. Как только в текущем цикле NA# оказывается установленным в активный уровень, 80386 освобождается, чтобы выдать на шину новый адрес и тип цикла до начала следующего состояния шины. Например, в Цикле 1 обеспечивает переход в режим конвейерной адресации, так как он начинается с состояния T1, но заканчивается состоянием T2p. Поскольку адрес для Цикла 2 устанавливается еще до начала Цикла 2, последний называется конвейерным циклом шины и начинается с состояния T1p. Цикл 2 начнется, как только активный сигнал READY# завершит Цикл 1. Примерами переходных циклов шины являются Цикл 1 на рис.6-17 и Цикл 2 на рис.6-16. На рис.6-17 показан переход в течение цикла шины, первого после нерабочего состояния шины, это самый кратчайший из возможных переходов в режим конвейерной адресации. Цикл 2 на рис.6-16 иллюстрирует переходной цикл шины, имеющий место внутри последовательности рабочих циклов шины. В любом случае переходные циклы осуществляются аналогично независимо от момента их появления: переходной цикл состоит по меньшей мере из состояний T1, T2 (в этот момент вы устанавливаете активный сигнал NA#), T2p (при условии, что 80386 уже имеет ожидающий обслуживания внутренний запрос шины, это условие выполняется почти всегда). Сос-

тояния T2p повторяются, если цикл дополняется состояниями ожидания.

Отметим три состояния (T1, T2 и T2p), комбинация которых требуется только в цикле шины, выполняющем переход из режима неконвейерной адресации в режим конвейерной адресации, например, Цикл 1 на рис.6-17. Циклы 2, 3 и 4 на рис.6-17 показывают, что в режиме конвейерной адресации могут выполняться циклы шины из двух состояний каждый, включающие только состояния T1p и T2p.

Когда выполняется конвейерный цикл шины, режим конвейерной адресации поддерживается путем установки активного сигнала NA# и определением того, что 80386 устанавливает состояние T2p шины в текущем цикле шины. Текущий цикл шины должен заканчиваться состоянием T2p для того, чтобы режим конвейерной адресации был сохранен и в следующем цикле. Состояние T2p идентифицируется установкой активного сигнала ADS#. На рис.6-16 и 6-17 режим конвейерной адресации заканчивается после Цикла 4, так как последним состоянием Цикла 4 является состояние T2i. Это означает, что 80386 не имел внутреннего запроса шины перед получением подтверждения Цикла 4. Если цикл заканчивается состоянием T2 или T2i, то следующий цикл будет неконвейерным.

В действительности, конвейерный адрес почти всегда устанавливается сразу после обнаружения активного уровня NA#. Это происходит потому, что при отсутствии любого другого запроса внутренний запрос предварительной выборки команды почти всегда ожидает обслуживания до тех пор, пока занят дешифратор команд и полностью заполнена очередь предварительно выбранных команд. Следовательно, конвейерный адрес устанавливается для длинных цепочек циклов шины, если шина доступна, и в каждом цикле шины оказывается установленным активный уровень сигнала NA#.

6.4.3.6 Конвейерная адресация при изменении размера шины данных

Наличие сигнала BS16# обеспечивает простое соединение с 16-разрядными шинами данных. Когда установлен активный уровень BS16#, схема шинного интерфейса 80386 выполняет соответствующие переключения, чтобы осуществить передачу, используя 16-разрядную шину данных, соединенную с линиями D0-D16.

Однако, при одновременном использовании сигналов NA# и BS16# имеет место некоторое взаимное влияние этих сигналов друг на друга. Это взаимное влияние проявляется тогда, когда требуются многократные циклы шины для передачи 32-разрядных операндов по 16-разрядной шине. Если операнду требуются обе 16-разрядные половины 32-разрядной шины, то в соответствии с этим требованием 80386 должен выполнить второй цикл шины для того, чтобы полностью передать весь операнд. Именно это требование приводит к конфликтной ситуации при использовании сигнала NA#.

Когда NA# оказывается установленным в активный уровень, 80386 дает себе разрешение на обработку следующего внутреннего, ожидающего обслуживания запроса шины и выдает на шину следующий подготовленный внутри адрес. Следовательно, активизация NA# делает невозможным повторную выборку в следующем цикле шины текущего адреса по линиям A2-A31, как это может потребоваться, когда сигнал BS16# активизирован внешним устройством.

Для разрешения этого конфликта схема 80386 разработана таким образом, чтобы удовлетворять следующим двум условиям:

1). Для разрешения конфликта 80386 разработан таким образом, чтобы игнорировать сигнал BS16# в текущем цикле шины, если NA# уже оказался установленным в текущем цикле. Если NA# оказался установленным, то текущий размер шины данных принимается равным 32 разрядам.

2). Также для разрешения конфликта в том случае, если и NA# и BS16# оказались установленными в один и тот же момент времени, активный BS16# обладает более высокими по сравнению с активным NA# приоритетом, и 80386 функционируется так, как если бы в этот момент сигнал NA# оказался неактивным. Внутренняя схема 80386, схематично показанная на рис.6-18 работает таким образом, чтобы сигнал BS16# воспринимался активным и сигнал NA# воспринимался неактивным, если оба входных сигнала активизированы внешними устройствами в один и тот же момент просмотра этих сигналов.

6.4.4 Циклы подтверждения прерывания(INTA)

В ответ на запрос прерывания, поступивший на вход INTR, когда прерывания разрешены, 80386 выполнит два цикла подтверждения прерывания. Эти циклы шины аналогичны циклам чтения, в которых имеющий место вид деятельности шины соответствует сигналам определения типа цикла шины, и каждый цикл продолжается до тех пор, пока процессор не получит подтверждение, наблюдая за сигналом READY#.

В зависимости от значения адресного разряда A2 различаются первый и второй циклы подтверждения прерывания. Адрес байта, выдаваемый в первом цикле подтверждения прерывания, равен 4 (A31-A3 низкие, A2 высокий, BE3#-BE1# высокие и BE0# низкий). Адрес, выдаваемый во втором цикле подтверждения прерывания, равен 0 (A31-A2 низкие, BE3#-BE1# высокие, BE0# низкий).

Активный уровень сигнала LOCK# устанавливается с начала первого цикла подтверждения прерывания и до конца второго цикла подтверждения прерывания. Четыре нерабочих состояния шины, Ti, вставляются процессором между двумя циклами подтверждения прерывания, чтобы обеспечить время блокированного нерабочего состояния шины ("мертвое" время) по меньшей мере длительностью 160 нс, что позволит в будущем ввести модификации скорости 80386, достигающие 24 МГц (при этом внешняя частота CLK2 должна достигать 48 МГц), что в свою очередь обеспечит совместимость с временным параметром TRHRL контроллера прерываний 8259A.

В течение обоих циклов подтверждения прерывания линии D0-D31 отключены.В конце первого цикла подтверждения прерывания данных для чтения не имеется. В конце второго цикла подтверждения прерывания 80386 считает внешний вектор прерывания по линиям D0-D7 шины данных. Вектор указывает определенный номер прерывания (от 0 до 255), требующего обслуживания.

6.4.5 Цикл индикации останова

80386 останавливается в результате выполнения инструкции HALT. Для сигнализирования входа процессора в состояние останова выполняется цикл индикации останова. Цикл индикации останова идентифицируется определенной комбинацией сигналов типа цикла шины, указанной в разделе 6.2.5 Сигналы определения типа цикла шины, и адресом байта, равным 2. BE0# и BE2# при этом служат только для различения цикла индикации останова от цикла индикации выключения, в котором выдается адрес, равный 0. В течение цикла останова данные, передаваемые по D0-D31,

неопределены.

Остановленный 80386 возобновляет функционирование, когда устанавливается активный уровень сигнала INTR (если прерывания разрешены), или сигнала NMI, или сигнала RESET.

6.4.6 Цикл индикации выключения

80386 выключается в результате появления ошибки защиты памяти при попытке обработать двойную ошибку. Для сигнализирования перехода процессора в выключенное состояние выполняется цикл индикации выключения. Цикл индикации выключения идентифицируется определенной комбинацией сигналов определения типа цикла шины, указанной в разделе 6.2.5 Сигналы определения типа цикла шины, и адресом байта, равным 0. Сигналы BE0# и BE2# при этом служат только для отличения цикла индикации выключения от цикла индикации останова, в котором выдается адрес, равный 2. В течение цикла выключения данные, выдаваемые на линии D0-D31, неопределены. Цикл индикации выключения должен быть подтвержден установкой активного уровня сигнала READY#.

Выключенный 80386 возобновляет функционирование, когда устанавливается активный уровень сигнала NMI или RESET.

6.5 Дополнительные сведения о функционировании

6.6.1 Вход и выход в/из состояния подтверждения захвата шины

Состояние подтверждения захвата шины, Th, вводится в ответ на установку активного уровня входного сигнала HOLD. В состоянии подтверждения захвата шины 80386 выключает все выходные или двунаправленные сигналы, исключая сигнал HLDA. Активный уровень сигнала HLDA поддерживается все время, пока 80386 находится в состоянии подтверждения захвата шины. В состоянии подтверждения захвата шины все входы, за исключением HOLD и RESET, игнорируются (исключение составляет также один положительный фронт сигнала NMI, который запоминается для последующей его обработки, когда сигнал HOLD перейдет в неактивное состояние).

Состояние Th может быть введено после нерабочего состояния шины, как на рис.6-25, или после подтверждения текущего физического цикла шины, если при этом уровень сигнала LOCK# неактивный, как на рис.6-26 и 5-27. Если установка активного сигнала BS16# требует выполнения второго 16-разрядного цикла шины для завершения передачи физического операнда, то этот цикл выполняется до подтверждения сигнала HOLD, хотя диаграммы состояний на рис.6-13 и 5-20 не отражают этой особенности.

Выход из состояния Th осуществляется в ответ на снятие активного уровня входного сигнала HOLD. Следующим состоянием будет состояние Ti, как на рис.6-25, если не имеется ожидающего обслуживания внутреннего запроса шины. Если же имеется внутренний, ожидающий решения запрос шины, то следующим состоянием шины будет состояние T1, как показано на рис.6-26 и 6-27.

Выход из состояния Th осуществляется также в ответ на установку активного уровня сигнала RESET.

Если в течение состояния Th появится положительный фронт на чувствительном к фронту входе NMI, то это событие запоминается как немаскируемое прерывание 2 и обслуживается после выхода процессора из состояния Th кроме случая, когда до выхода из Th произойдет сброс 80386.

6.6.2 Сброс в состоянии подтверждения захвата шины

Активный сигнал RESET обладает более высоким приоритетом по сравнению с активным сигналом HOLD. Следовательно, в ответ на установку активного уровня на входе RESET осуществляется выход из состояния Th. Если сигнал RESET активизируется во время действия сигнала HOLD, то 80386 установит свои выводы в определенные состояния в соответствии с табл.6-3 "Состояния выводов в течение действия сигнала RESET" и выполнит обычную процедуру внутреннего сброса.

Если активный уровень сигнала HOLD остается установленным и после снятия активного сигнала RESET, то 80386 перейдет в состояние подтверждения захвата шины до того, как выполнит свой первый цикл шины, но при условии, что HOLD все еще остается активным в момент, когда 80386 в другом случае приступил бы к выполнению своего первого цикла шины. Если HOLD остается активным после снятия сигнала RESET, то вход BUSY# все равно анализируется как обычно, чтобы определить требуется ли самотестирование, сигнал ERROR# также при этом анализируется как обычно, чтобы определить, какой из двух воэможных сопроцессоров присутствует в системе (или когда сопроцессора вобще нет).

6.6.3 Функционирование шины в течение и после действия сигнала RESET

RESET является самым приоритетным входным сигналом, при установке активного уровня RESET прерывается любая деятельность процессора. Выполняемый цикл шины может быть прерванным на любой стадии, а нерабочие состояния или состояния подтверждения захвата шины прекращаются при установке состояния сброса.

RESET должен поддерживаться в активном состоянии в течение по меньшей мере 15 периодов частоты CLK2, чтобы он был уверенно принят всеми схемами 80386, и по меньшей мере в течение 78 периодов CLK2, если выполняется самотестирование 80386, запрос на которое анализируется во время отрицательного фронта RESET.

Активные импульсы RESET длительностью меньше 15 периодов CLK2 могут быть не восприняты.

Активные импульсы RESET длительностью меньше 78 периодов CLK2, за которыми следует самотестирование, могут привести к тому, что тест-структура выдаст сообщение о неисправности, когда в действительности неисправности не существует. Дополнительное расширение импульса RESET необходимо для получения достоверных результатов самодиагностирования.

При условии, что отрицательный фронт RESET отвечает требованиям, предъявляемым к времени установки t25 и времени удержания t26, этот фронт определит фазу внутренней тактовой частоты процессора, как показано на рис.6-28 и рис.7-7.

Самодиагностирование 80386 можно запустить, если поддерживать сигнал BUSY# на низком уровне в момент снятия сигнала RESET, как показано на рис.6-28. Для выполнения всей процедуры самодиагностирования требуется [(2^20)+ приблизительно 60] периодов CLK2. Результаты тестирования не влияют на продолжительность самодиагностирования.

Даже если после тестирования тест-структура указывает на наличие неисправности, 80386 все равно перейдет к выполнению процедуры, которая должна была следовать за сбросом 80386.

После отрицательного фронта RESET (и после самотестиро-

вания, если в нем была необходимость) 80386 выполнит последовательность внутренней инициализации за время, приблизительно равное 350-450 периодам CLK2. Во время инициализации, между двадцатым периодом CLK2 и первым циклом шины (который последует за инициализацией), 80386 анализирует состояние входа ERROR#, чтобы отличить случай присутсвия в системе сопроцессора 80387 от случая, когда присутствует 80287 или в системе вобще нет сопропроцессора. Различие между последними двумя случаями (в системе присутствует 80287 или в системе нет сопроцессора) задается программно.

6.6 Сигнатура самотестирования

По завершению самотестирования (если самотестирование было запрошено путем поддержания низкого уровня сигнала BUSY# во время отрицательного фронта сигнала RESET), если не было обнаружено неисправностей 80386, то значение каждого из регистров AX и DX будет равно 0000H. Это справедливо для всех модификаций 80386. Ненулевые значения регистров AX или DX после самотестирования указывает на то, что какой-то блок 80386 неисправен.

6.7 Идентификаторы типа и модификации

Чтобы помочь пользователям 80386, 80386 после сброса поддерживает идентификатор типа и идентификатор модификации соответственно в регистрах BH и BL. BH содержит 03H для идентификации типа 80386. BL содержит беззнаковое двоичное число, соответствующее версии данного типа процессора. Хронология изменения идентификатора модификации (версии) 80386 в BL такова: он начинается с нуля и изменяется (обычно увеличивается) при изменениях данного типа процессора, предназначенных для усовершенствования данного типа процессора по сравнению с предыдущими версиями.

Эти особенности предназначены для того, чтобы помочь пользователям 80386 в их практической деятельности. Однако не гарантируется, что значение идентификатора версии будет изменяться с каждым изменением версии или что изменения этого значения в зависимости от содержания или цели версии или в зависимости от материалов, требующих изменения, будут следовать строго по непрерывной числовой последовательности. Фирма Intel поступает с этими характеристиками данного типа процессора только по своему усмотрению.

Табл.6-10.

История идентификаторов типа и версии

Содержание изменения 80386	Идентификатор типа	Идентификатор версии	Содержание изменения 80386	Идентификатор типа	Идентификатор версии