Основы микропроцессорной техники

Авторы: Ю.В. Новиков, П.К. Скоробогатов

Оглавление

Глава 1. Философия микропроцессорной техники 2

1. Лекция: Философия микропроцессорной техники 2

1.1. Что такое микропроцессор? 4

1.2. Шинная структура связей 6

1.3. Режимы работы микропроцессорной системы 10

1.4. Архитектура микропроцессорных систем 13

1.5. Типы микропроцессорных систем 15

Глава 2. Организация обмена информацией 17

2. Лекция: Шины микропроцессорной системы и циклы обмена 17

2.1. Шины микропроцессорной системы 18

2.2. Циклы обмена информацией 20

2.3. Прохождение сигналов по магистрали 27

3. Лекция: Функции устройств магистрали 29

2.4. Функции устройств магистрали 29

4. Лекция: Адресация операндов 40

3.1. Адресация операндов 40

3.2. Регистры процессора 46

5. Лекция: Система команд процессора 48

3.3. Система команд процессора 48

3.4. Быстродействие процессора 54

6. Лекция: Процессорное ядро и память микроконтроллеров 57

4.1. Классификация и структура микроконтроллеров 57

4.2. Процессорное ядро микроконтроллера 58

4.3. Память программ и данных МК 61

7. Лекция: Организация связи микроконтроллера с внешней средой и временем 64

4.4. Порты ввода/вывода 64

4.5. Таймеры и процессоры событий 65

4.6. Модуль прерываний МК 71

8. Лекция: Вспомогательные аппаратные средства микроконтроллера 71

4.7. Минимизация энергопотребления в системах на основе МК 71

4.8. Тактовые генераторы МК 73

4.9. Аппаратные средства обеспечения надежной работы МК 74

4.10. Дополнительные модули МК 77

Глава 5. Однокристальные микроконтроллеры серии PIC 80

9. Лекция: Аппаратные средства микроконтроллеров серии PIC 80

5.1. Основные особенности микроконтроллеров серии PIC 80
5.2. Микроконтроллеры подгруппы PIC16F8X 83
11. Лекция: Особенности разработки цифровых устройств на основе микроконтроллеров 103
6.1. Разработка микропроцессорной системы на основе микроконтроллера 103
Глава 7. Организация персонального компьютера 109
13. Лекция: Архитектура и процессоры персональных компьютеров 109
7.1. Архитектура персонального компьютера 109
7.2. Процессоры персональных компьютеров 113

Глава 1. Философия микропроцессорной техники

1. Лекция: Философия микропроцессорной техники

В этой лекции рассказывается о базовой терминологии микропроцессорной техники, о принципах организации микропроцессорных систем, о структуре связей, режимах работы и об основных типах микропроцессорных систем.

В этой главе рассматриваются базовые концепции, которые лежат в основе любой микропроцессорной системы — от простейшего микроконтроллера до сложного компьютера. Именно в этом смысле здесь используется термин "философия".

Для начала несколько основных определений.

- **Электронная система** — в данном случае это любой электронный узел, блок, прибор или комплекс, производящий обработку информации.
- **Задача** — это набор функций, выполнение которых требуется от электронной системы.
- **Быстродействие** — это показатель скорости выполнения электронной системой ее функций.
- **Гибкость** — это способность системы подстраиваться под различные задачи.
- **Избыточность** — это показатель степени соответствия возможностей системы решаемой данной системой задаче.
- **Интерфейс** — соглашение об обмене информацией, правила обмена информацией, подразумевающие электрическую, логическую и конструктивную совместимость устройств, участвующих в обмене. Другое название — **сопряжение**.

Микропроцессорная система может рассматриваться как частный случай электронной системы, предназначенной для обработки входных сигналов и выдачи выходных сигналов (рис. 1.1). В качестве входных и выходных сигналов при этом могут использоваться аналоговые сигналы, одиночные цифровые сигналы, цифровые коды, последовательности цифровых кодов. Внутри системы может производиться хранение, накопление сигналов (или информации), но суть от этого не меняется. Если система цифровая (а микропроцессорные системы относятся к разряду цифровых), то входные аналоговые сигналы преобразуются в последовательности кодов выборок с помощью АЦП, а выходные аналоговые сигналы формируются из последовательности кодов выборок с помощью ЦАП. Обработка и хранение информации производятся в цифровом виде.

Характерная особенность традиционной цифровой системы состоит в том, что алгоритмы обработки и хранения информации в ней жестко связаны со схемотехникой системы. То есть изменение этих алгоритмов возможно только путем изменения структуры системы, замены электронных узлов, входящих в систему, и/или связей между ними. Например, если нам нужна дополнительная операция

суммирования, то необходимо добавить в структуру системы лишний сумматор. Или если нужна дополнительная функция хранения кода в течение одного такта, то мы должны добавить в структуру еще один регистр. Естественно, это практически невозможно сделать в процессе эксплуатации, обязательно нужен новый производственный цикл проектирования, изготовления, отладки всей системы. Именно поэтому традиционная цифровая система часто называется системой на "жесткой логике".

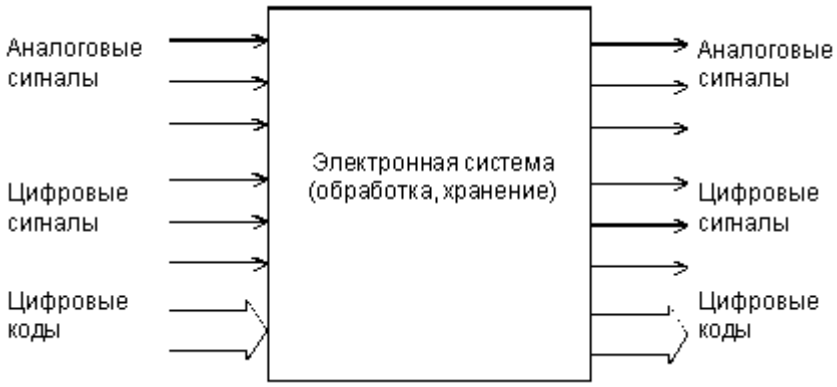

Рис. 1.1. Электронная система.

Любая система на "жесткой логике" обязательно представляет собой специализированную систему, настроенную исключительно на одну задачу или (реже) на несколько близких, заранее известных задач. Это имеет свои бесспорные преимущества.

Во-первых, специализированная система (в отличие от универсальной) никогда не имеет аппаратурной избыточности, то есть каждый ее элемент обязательно работает в полную силу (конечно, если эта система грамотно спроектирована).

Во-вторых, именно специализированная система может обеспечить максимально высокое быстродействие, так как скорость выполнения алгоритмов обработки информации определяется в ней только быстродействием отдельных логических элементов и выбранной схемой путей прохождения информации. А именно логические элементы всегда обладают максимальным на данный момент быстродействием.

Но в то же время большим недостатком цифровой системы на "жесткой логике" является то, что для каждой новой задачи ее надо проектировать и изготавливать заново. Это процесс длительный, дорогостоящий, требующий высокой квалификации исполнителей. А если решаемая задача вдруг изменяется, то вся аппаратура должна быть полностью заменена. В нашем быстро меняющемся мире это довольно расточительно.

Путь преодоления этого недостатка довольно очевиден: надо построить такую систему, которая могла бы легко адаптироваться под любую задачу, перестраиваться с одного алгоритма работы на другой без изменения аппаратуры. И задавать тот или иной алгоритм мы тогда могли бы путем ввода в систему некой дополнительной управляющей информации, **программы** работы системы (рис. 1.2). Тогда система станет универсальной, или **программируемой**, не жесткой, а гибкой. Именно это и обеспечивает микропроцессорная система.

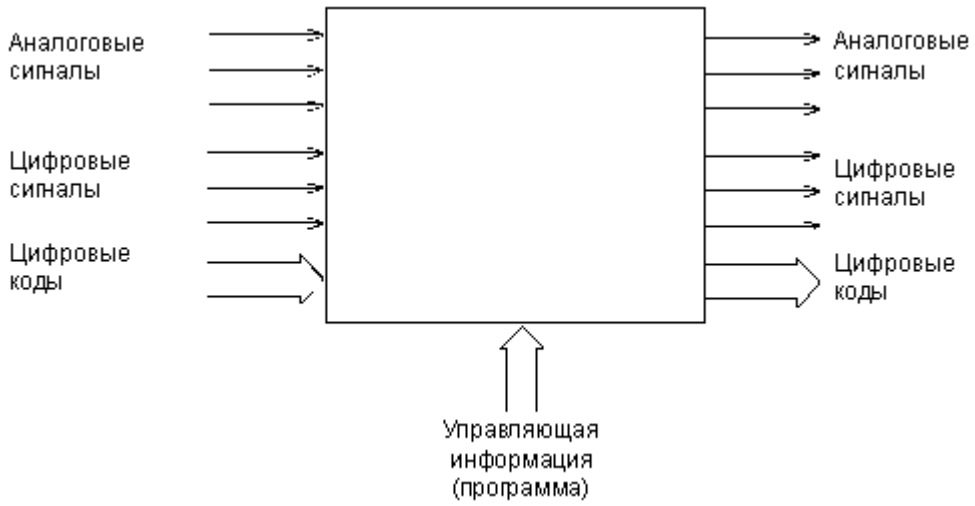

Рис. 1.2. Программируемая (она же универсальная) электронная система.

Но любая универсальность обязательно приводит к избыточности. Ведь решение максимально трудной задачи требует гораздо больше средств, чем решение максимально простой задачи. Поэтому сложность универсальной системы должна быть такой, чтобы обеспечивать решение самой трудной задачи, а при решении простой задачи система будет работать далеко не в полную силу, будет использовать не все свои ресурсы. И чем проще решаемая задача, тем больше избыточность, и тем менее оправданной становится универсальность. Избыточность ведет к увеличению стоимости системы, снижению ее надежности, увеличению потребляемой мощности и т.д.

Кроме того, универсальность, как правило, приводит к существенному снижению быстродействия. Оптимизировать универсальную систему так, чтобы каждая новая задача решалась максимально быстро, попросту невозможно. Общее правило таково: чем больше универсальность, гибкость, тем меньше быстродействие. Более того, для универсальных систем не существует таких задач (пусть даже и самых простых), которые бы они решали с максимально возможным быстродействием. За все приходится платить.

Таким образом, можно сделать следующий вывод. Системы на "жесткой логике" хороши там, где решаемая задача не меняется длительное время, где требуется самое высокое быстродействие, где алгоритмы обработки информации предельно просты. А универсальные, программируемые системы хороши там, где часто меняются решаемые задачи, где высокое быстродействие не слишком важно, где алгоритмы обработки информации сложные. То есть любая система хороша на своем месте.

Однако за последние десятилетия быстродействие универсальных (микропроцессорных) систем сильно выросло (на несколько порядков). К тому же большой объем выпуска микросхем для этих систем привел к резкому снижению их стоимости. В результате область применения систем на "жесткой логике" резко сузилась. Более того, высокими темпами развиваются сейчас программируемые системы, предназначенные для решения одной задачи или нескольких близких задач. Они удачно совмещают в себе как достоинства систем на "жесткой логике", так и программируемых систем, обеспечивая сочетание достаточно высокого быстродействия и необходимой гибкости. Так что вытеснение "жесткой логики" продолжается.

1.1. Что такое микропроцессор?

Ядром любой микропроцессорной системы является микропроцессор или просто процессор (от английского processor). Перевести на русский язык это слово правильнее всего как "обработчик", так как именно микропроцессор — это тот узел, блок, который производит всю обработку информации внутри микропроцессорной системы. Остальные узлы выполняют всего лишь вспомогательные функции: хранение информации (в том числе и управляющей информации, то есть программы), связи с внешними устройствами, связи с пользователем и т.д. Процессор заменяет практически всю "же-

сткую логику", которая понадобилась бы в случае традиционной цифровой системы. Он выполняет арифметические функции (сложение, умножение и т.д.), логические функции (сдвиг, сравнение, маскирование кодов и т.д.), временное хранение кодов (во внутренних регистрах), пересылку кодов между узлами микропроцессорной системы и многое другое. Количество таких элементарных операций, выполняемых процессором, может достигать нескольких сотен. Процессор можно сравнить с мозгом системы.

Но при этом надо учитывать, что все свои операции процессор выполняет **последовательно**, то есть одну за другой, по очереди. Конечно, существуют процессоры с параллельным выполнением некоторых операций, встречаются также микропроцессорные системы, в которых несколько процессоров работают над одной задачей параллельно, но это редкие исключения. С одной стороны, последовательное выполнение операций — несомненное достоинство, так как позволяет с помощью всего одного процессора выполнять любые, самые сложные алгоритмы обработки информации. Но, с другой стороны, последовательное выполнение операций приводит к тому, что время выполнения алгоритма зависит от его сложности. Простые алгоритмы выполняются быстрее сложных. То есть микропроцессорная система способна сделать все, но работает она не слишком быстро, ведь все информационные потоки приходится пропускать через один-единственный узел — микропроцессор (рис. 1.3). В традиционной цифровой системе можно легко организовать параллельную обработку всех потоков информации, правда, ценой усложнения схемы.

Рис. 1.3. Информационные потоки в микропроцессорной системе.

Итак, микропроцессор способен выполнять множество операций. Но откуда он узнает, какую операцию ему надо выполнять в данный момент? Именно это определяется управляющей информацией, программой. Программа представляет собой набор **команд (инструкций)**, то есть цифровых кодов, расшифровав которые, процессор узнает, что ему надо делать. Программа от начала и до конца составляется человеком, программистом, а процессор выступает в роли послушного исполнителя этой программы, никакой инициативы он не проявляет (если, конечно, исправен). Поэтому сравнение процессора с мозгом не слишком корректно. Он всего лишь исполнитель того алгоритма, который заранее составил для него человек. Любое отклонение от этого алгоритма может быть вызвано только неисправностью процессора или каких-нибудь других узлов микропроцессорной системы.

Все команды, выполняемые процессором, образуют **систему команд** процессора. Структура и объем системы команд процессора определяют его быстродействие, гибкость, удобство использования. Всего команд у процессора может быть от нескольких десятков до нескольких сотен. Система команд может быть рассчитана на узкий круг решаемых задач (у специализированных процессоров) или на максимально широкий круг задач (у универсальных процессоров). Коды команд могут иметь различное количество разрядов (занимать от одного до нескольких байт). Каждая команда имеет свое время выполнения, поэтому время выполнения всей программы зависит не только от количества команд в программе, но и от того, какие именно команды используются.

Для выполнения команд в структуру процессора входят внутренние регистры, арифметико-логическое устройство (АЛУ, ALU — Arithmetic Logic Unit) , мультиплексоры, буферы, регистры и другие узлы. Работа всех узлов синхронизируется общим внешним тактовым сигналом процессора. То есть процессор представляет собой довольно сложное цифровое устройство (рис. 1.4).

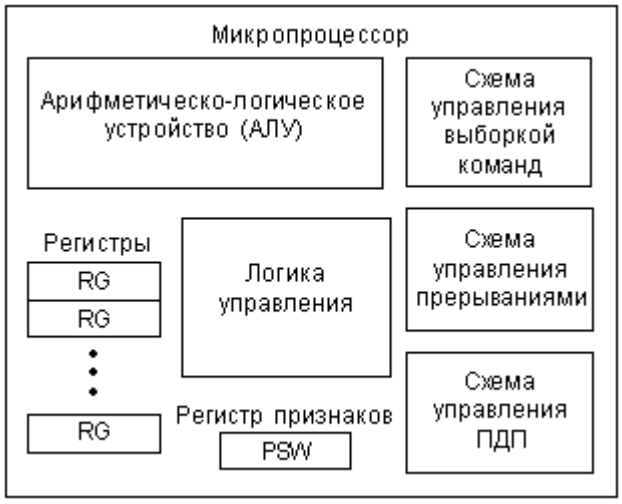

Рис. 1.4. Пример структуры простейшего процессора.

Впрочем, для разработчика микропроцессорных систем информация о тонкостях внутренней структуры процессора не слишком важна. Разработчик должен рассматривать процессор как "черный ящик", который в ответ на входные и управляющие коды производит ту или иную операцию и выдает выходные сигналы. Разработчику необходимо знать систему команд, режимы работы процессора, а также правила взаимодействия процессора с внешним миром или, как их еще называют, протоколы обмена информацией. О внутренней структуре процессора надо знать только то, что необходимо для выбора той или иной команды, того или иного режима работы.

1.2. Шинная структура связей

Для достижения максимальной универсальности и упрощения протоколов обмена информацией в микропроцессорных системах применяется так называемая шинная структура связей между отдельными устройствами, входящими в систему. Суть шинной структуры связей сводится к следующему.

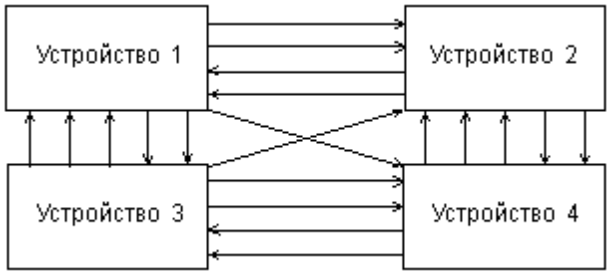

Рис. 1.5. Классическая структура связей.

При классической структуре связей (рис. 1.5) все сигналы и коды между устройствами передаются по отдельным линиям связи. Каждое устройство, входящее в систему, передает свои сигналы и коды независимо от других устройств. При этом в системе получается очень много линий связи и разных протоколов обмена информацией.

При шинной структуре связей (рис. 1.6) все сигналы между устройствами передаются по одним и тем же линиям связи, но в разное время (это называется мультиплексированной передачей). Причем передача по всем линиям связи может осуществляться в обоих направлениях (так называемая двунаправленная передача). В результате количество линий связи существенно сокращается, а правила обмена (протоколы) упрощаются. Группа линий связи, по которым передаются сигналы или коды как раз и называется шиной (англ. bus).

Понятно, что при шинной структуре связей легко осуществляется пересылка всех информационных потоков в нужном направлении, например, их можно пропустить через один процессор, что очень важно для микропроцессорной системы. Однако при шинной структуре связей вся информация передается по линиям связи последовательно во времени, по очереди, что снижает быстродействие системы по сравнению с классической структурой связей.

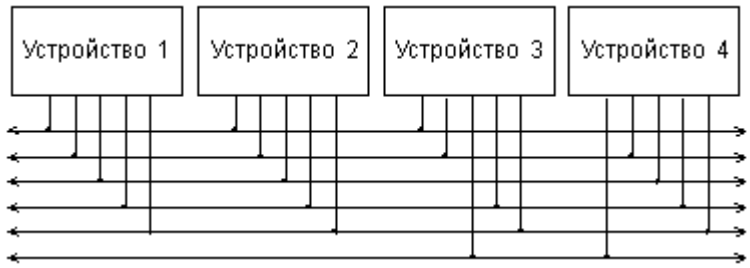

Рис. 1.6. Шинная структура связей.

Большое достоинство шинной структуры связей состоит в том, что все устройства, подключенные к шине, должны принимать и передавать информацию по одним и тем же правилам (протоколам обмена информацией по шине). Соответственно, все узлы, отвечающие за обмен с шиной в этих устройствах, должны быть единообразны, унифицированы.

Существенный недостаток шинной структуры связан с тем, что все устройства подключаются к каждой линии связи параллельно. Поэтому любая неисправность любого устройства может вывести из строя всю систему, если она портит линию связи. По этой же причине отладка системы с шинной структурой связей довольно сложна и обычно требует специального оборудования.

В системах с шинной структурой связей применяют все три существующие разновидности выходных каскадов цифровых микросхем:

- стандартный выход или выход с двумя состояниями (обозначается 2С, 2S, реже ТТЛ, TTL);
- выход с открытым коллектором (обозначается ОК, ОС);
- выход с тремя состояниями или (что то же самое) с возможностью отключения (обозначается 3С, 3S).

Упрощенно эти три типа выходных каскадов могут быть представлены в виде схем на рис. 1.7.

У выхода 2С два ключа замыкаются по очереди, что соответствует уровням логической единицы (верхний ключ замкнут) и логического нуля (нижний ключ замкнут). У выхода ОК замкнутый ключ формирует уровень логического нуля, разомкнутый — логической единицы. У выхода 3С ключи могут замыкаться по очереди (как в случае 2С), а могут размыкаться одновременно, образуя третье, высокоимпедансное, состояние. Переход в третье состояние (Z-состояние) управляется сигналом на специальном входе EZ.

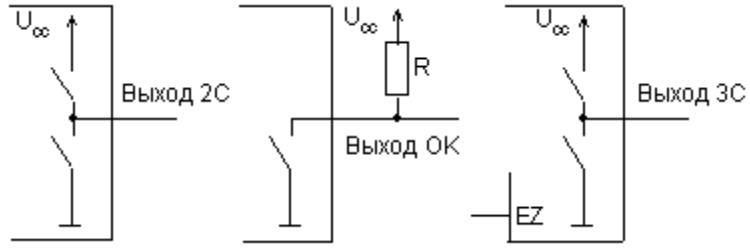

Рис. 1.7. Три типа выходов цифровых микросхем.

Выходные каскады типов 3С и ОК позволяют объединять несколько выходов микросхем для получения мультиплексированных (рис. 1.8) или двунаправленных (рис. 1.9) линий.

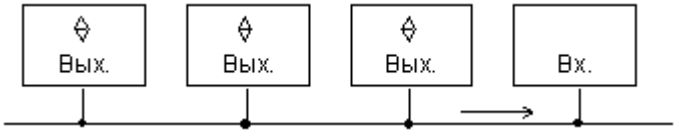

Рис. 1.8. Мультиплексированная линия.

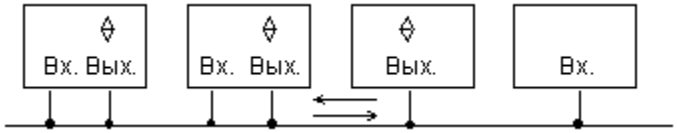

Рис. 1.9. Двунаправленная линия.

При этом в случае выходов 3С необходимо обеспечить, чтобы на линии всегда работал только один активный выход, а все остальные выходы находились бы в это время в третьем состоянии, иначе возможны конфликты. Объединенные выходы ОК могут работать все одновременно, без всяких конфликтов.

Типичная структура микропроцессорной системы приведена на рис. 1.10. Она включает в себя три основных типа устройств:

- процессор;
- память, включающую оперативную память (ОЗУ, RAM — Random Access Memory) и постоянную память (ПЗУ, ROM —Read Only Memory), которая служит для хранения данных и программ;
- устройства ввода/вывода (УВВ, I/O — Input/Output Devices), служащие для связи микропроцессорной системы с внешними устройствами, для приема (ввода, чтения, Read) входных сигналов и выдачи (вывода, записи, Write) выходных сигналов.

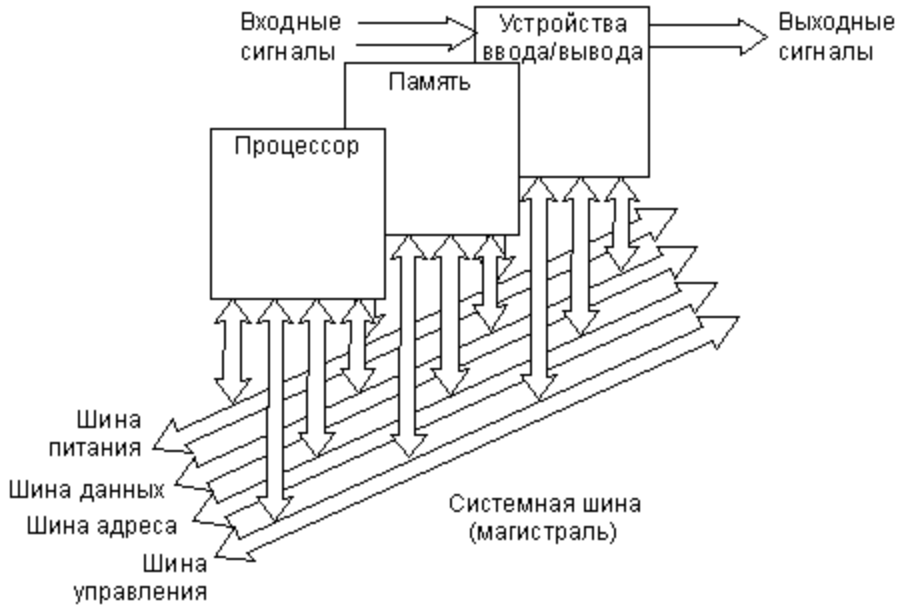

Рис. 1.10. Структура микропроцессорной системы.

Все устройства микропроцессорной системы объединяются общей системной шиной (она же называется еще **системной магистралью** или **каналом**). Системная магистраль включает в себя четыре основные шины нижнего уровня:

- шина адреса (Address Bus);

- шина данных (Data Bus);
- шина управления (Control Bus);
- шина питания (Power Bus).

Шина адреса служит для определения адреса (номера) устройства, с которым процессор обменивается информацией в данный момент. Каждому устройству (кроме процессора), каждой ячейке памяти в микропроцессорной системе присваивается собственный адрес. Когда код какого-то адреса выставляется процессором на шине адреса, устройство, которому этот адрес приписан, понимает, что ему предстоит обмен информацией. Шина адреса может быть однонаправленной или двунаправленной.

Шина данных — это основная шина, которая используется для передачи информационных кодов между всеми устройствами микропроцессорной системы. Обычно в пересылке информации участвует процессор, который передает код данных в какое-то устройство или в ячейку памяти или же принимает код данных из какого-то устройства или из ячейки памяти. Но возможна также и передача информации между устройствами без участия процессора. Шина данных всегда двунаправленная.

Шина управления в отличие от шины адреса и шины данных состоит из отдельных управляющих сигналов. Каждый из этих сигналов во время обмена информацией имеет свою функцию. Некоторые сигналы служат для стробирования передаваемых или принимаемых данных (то есть определяют моменты времени, когда информационный код выставлен на шину данных). Другие управляющие сигналы могут использоваться для подтверждения приема данных, для сброса всех устройств в исходное состояние, для тактирования всех устройств и т.д. Линии шины управления могут быть однонаправленными или двунаправленными.

Наконец, шина питания предназначена не для пересылки информационных сигналов, а для питания системы. Она состоит из линий питания и общего провода. В микропроцессорной системе может быть один источник питания (чаще +5 В) или несколько источников питания (обычно еще –5 В, +12 В и –12 В). Каждому напряжению питания соответствует своя линия связи. Все устройства подключены к этим линиям параллельно.

Если в микропроцессорную систему надо ввести входной код (или входной сигнал), то процессор по шине адреса обращается к нужному устройству ввода/вывода и принимает по шине данных входную информацию. Если из микропроцессорной системы надо вывести выходной код (или выходной сигнал), то процессор обращается по шине адреса к нужному устройству ввода/вывода и передает ему по шине данных выходную информацию.

Если информация должна пройти сложную многоступенчатую обработку, то процессор может хранить промежуточные результаты в системной оперативной памяти. Для обращения к любой ячейке памяти процессор выставляет ее адрес на шину адреса и передает в нее информационный код по шине данных или же принимает из нее информационный код по шине данных. В памяти (оперативной и постоянной) находятся также и управляющие коды (команды выполняемой процессором программы), которые процессор также читает по шине данных с адресацией по шине адреса. Постоянная память используется в основном для хранения программы начального пуска микропроцессорной системы, которая выполняется каждый раз после включения питания. Информация в нее заносится изготовителем раз и навсегда.

Таким образом, в микропроцессорной системе все информационные коды и коды команд передаются по шинам последовательно, по очереди. Это определяет сравнительно невысокое быстродействие микропроцессорной системы. Оно ограничено обычно даже не быстродействием процессора (которое тоже очень важно) и не скоростью обмена по системной шине (магистрали), а именно последовательным характером передачи информации по системной шине (магистрали).

Важно учитывать, что устройства ввода/вывода чаще всего представляют собой устройства на "жесткой логике". На них может быть возложена часть функций, выполняемых микропроцессорной системой. Поэтому у разработчика всегда имеется возможность перераспределять функции системы между аппаратной и программной реализациями оптимальным образом. Аппаратная реализация ус-

коряет выполнение функции, но имеет недостаточную гибкость. Программная реализация значительно медленнее, но обеспечивает высокую гибкость. Аппаратная реализация функций увеличивает стоимость системы и ее энергопотребление, программная — не увеличивает. Чаще всего применяется комбинирование аппаратных и программных функций.

Иногда устройства ввода/вывода имеют в своем составе процессор, то есть представляют собой небольшую специализированную микропроцессорную систему. Это позволяет переложить часть программных функций на устройства ввода/вывода, разгрузив центральный процессор системы.

1.3. Режимы работы микропроцессорной системы

Как уже отмечалось, микропроцессорная система обеспечивает большую гибкость работы, она способна настраиваться на любую задачу. Гибкость эта обусловлена прежде всего тем, что функции, выполняемые системой, определяются программой (программным обеспечением, software), которую выполняет процессор. Аппаратура (аппаратное обеспечение, hardware) остается неизменной при любой задаче. Записывая в память системы программу, можно заставить микропроцессорную систему выполнять любую задачу, поддерживаемую данной аппаратурой. К тому же шинная организация связей микропроцессорной системы позволяет довольно легко заменять аппаратные модули, например, заменять память на новую большего объема или более высокого быстродействия, добавлять или модернизировать устройства ввода/вывода, наконец, заменять процессор на более мощный. Это также позволяет увеличить гибкость системы, продлить ее жизнь при любом изменении требований к ней.

Но гибкость микропроцессорной системы определяется не только этим. Настраиваться на задачу помогает еще и выбор режима работы системы, то есть режима обмена информацией по системной магистрали (шине).

Практически любая развитая микропроцессорная система (в том числе и компьютер) поддерживает три основных режима обмена по магистрали:

- программный обмен информацией;
- обмен с использованием прерываний (Interrupts);
- обмен с использованием прямого доступа к памяти (ПДП, DMA — Direct Memory Access).

Программный обмен информацией является основным в любой микропроцессорной системе. Он предусмотрен всегда, без него невозможны другие режимы обмена. В этом режиме процессор является единоличным хозяином (или задатчиком, Master) системной магистрали. Все операции (циклы) обмена информацией в данном случае инициируются только процессором, все они выполняются строго в порядке, предписанном исполняемой программой.

Процессор читает (выбирает) из памяти коды команд и исполняет их, читая данные из памяти или из устройства ввода/вывода, обрабатывая их, записывая данные в память или передавая их в устройство ввода/вывода. Путь процессора по программе может быть линейным, циклическим, может содержать переходы (прыжки), но он всегда непрерывен и полностью находится под контролем процессора. Ни на какие внешние события, не связанные с программой, процессор не реагирует (рис. 1.11). Все сигналы на магистрали в данном случае контролируются процессором.

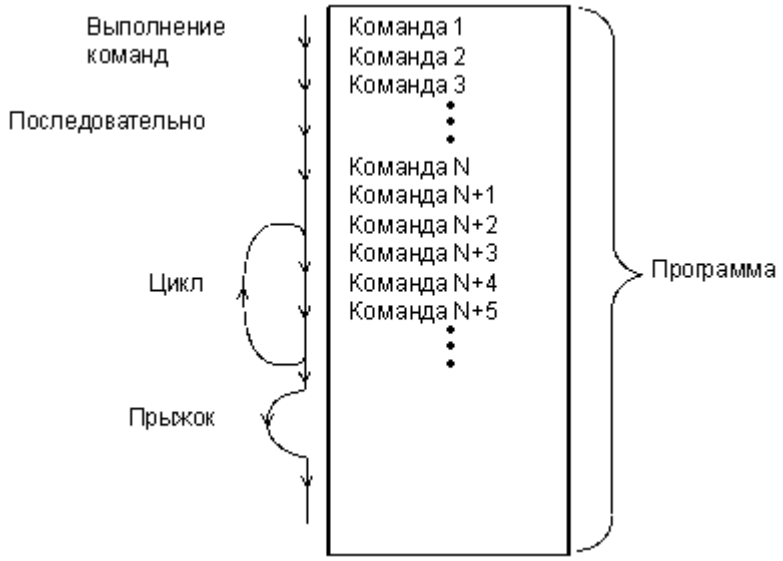

Рис. 1.11. Программный обмен информацией.

Обмен по прерываниям используется тогда, когда необходима реакция микропроцессорной системы на какое-то внешнее событие, на приход внешнего сигнала. В случае компьютера внешним событием может быть, например, нажатие на клавишу клавиатуры или приход по локальной сети пакета данных. Компьютер должен реагировать на это, соответственно, выводом символа на экран или же чтением и обработкой принятого по сети пакета.

В общем случае организовать реакцию на внешнее событие можно тремя различными путями:

- с помощью постоянного программного контроля факта наступления события (так называемый метод опроса флага или polling);
- с помощью прерывания, то есть насильственного перевода процессора с выполнения текущей программы на выполнение экстренно необходимой программы;
- с помощью прямого доступа к памяти, то есть без участия процессора при его отключении от системной магистрали.

Проиллюстрировать эти три способа можно следующим простым примером. Допустим, вы готовите себе завтрак, поставив на плиту кипятиться молоко. Естественно, на закипание молока надо реагировать, причем срочно. Как это организовать? Первый путь — постоянно следить за молоком, но тогда вы ничего другого не сможете делать. Правильнее будет регулярно поглядывать на молоко, делая одновременно что-то другое. Это программный режим с опросом флага. Второй путь — установить на кастрюлю с молоком датчик, который подаст звуковой сигнал при закипании молока, и спокойно заниматься другими делами. Услышав сигнал, вы выключите молоко. Правда, возможно, вам придется сначала закончить то, что вы начали делать, так что ваша реакция будет медленнее, чем в первом случае. Наконец, третий путь состоит в том, чтобы соединить датчик на кастрюле с управлением плитой так, чтобы при закипании молока горелка была выключена без вашего участия (правда, аналогия с ПДП здесь не очень точная, так как в данном случае на момент выполнения действия вас не отвлекают от работы).

Первый случай с опросом флага реализуется в микропроцессорной системе постоянным чтением информации процессором из устройства ввода/вывода, связанного с тем внешним устройством, на поведение которого необходимо срочно реагировать.

Во втором случае в режиме прерывания процессор, получив запрос прерывания от внешнего устройства (часто называемый IRQ — Interrupt ReQuest), заканчивает выполнение текущей команды и переходит к программе обработки прерывания. Закончив выполнение программы обработки прерывания, он возвращается к прерванной программе с той точки, где его прервали (рис. 1.12).

Здесь важно то, что вся работа, как и в случае программного режима, осуществляется самим процессором, внешнее событие просто временно отвлекает его. Реакция на внешнее событие по прерыванию в общем случае медленнее, чем при программном режиме. Как и в случае программного обмена, здесь все сигналы на магистрали выставляются процессором, то есть он полностью контролирует магистраль. Для обслуживания прерываний в систему иногда вводится специальный модуль контроллера прерываний, но он в обмене информацией не участвует. Его задача состоит в том, чтобы упростить работу процессора с внешними запросами прерываний. Этот контроллер обычно программно управляется процессором по системной магистрали.

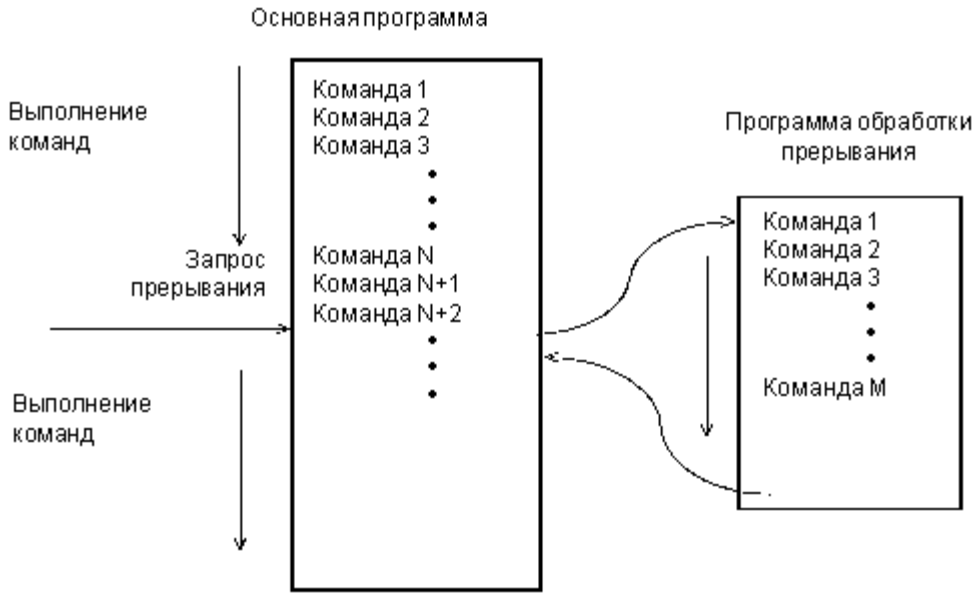

Рис. 1.12. Обслуживание прерывания.

Естественно, никакого ускорения работы системы прерывание не дает. Его применение позволяет только отказаться от постоянного опроса флага внешнего события и временно, до наступления внешнего события, занять процессор выполнением каких-то других задач.

Прямой доступ к памяти (ПДП, DMA) — это режим, принципиально отличающийся от двух ранее рассмотренных режимов тем, что обмен по системной шине идет без участия процессора. Внешнее устройство, требующее обслуживания, сигнализирует процессору, что режим ПДП необходим, в ответ на это процессор заканчивает выполнение текущей команды и отключается от всех шин, сигнализируя запросившему устройству, что обмен в режиме ПДП можно начинать.

Операция ПДП сводится к пересылке информации из устройства ввода/вывода в память или же из памяти в устройство ввода/вывода. Когда пересылка информации будет закончена, процессор вновь возвращается к прерванной программе, продолжая ее с той точки, где его прервали (рис. 1.13). Это похоже на режим обслуживания прерываний, но в данном случае процессор не участвует в обмене. Как и в случае прерываний, реакция на внешнее событие при ПДП существенно медленнее, чем при программном режиме.

Понятно, что в этом случае требуется введение в систему дополнительного устройства (контроллера ПДП), которое будет осуществлять полноценный обмен по системной магистрали без всякого участия процессора. Причем процессор предварительно должен сообщить этому контроллеру ПДП, откуда ему следует брать информацию и/или куда ее следует помещать. Контроллер ПДП может считаться специализированным процессором, который отличается тем, что сам не участвует в обмене, не принимает в себя информацию и не выдает ее (рис. 1.14).

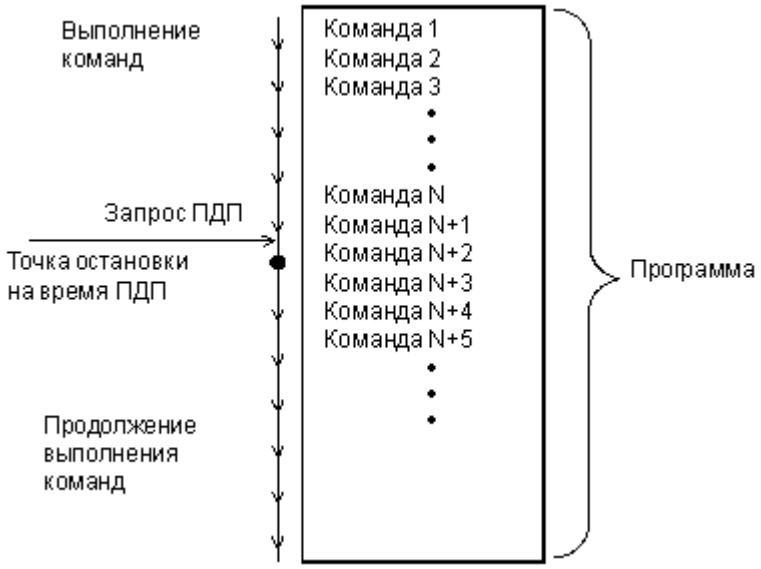

Рис. 1.13. Обслуживание ПДП.

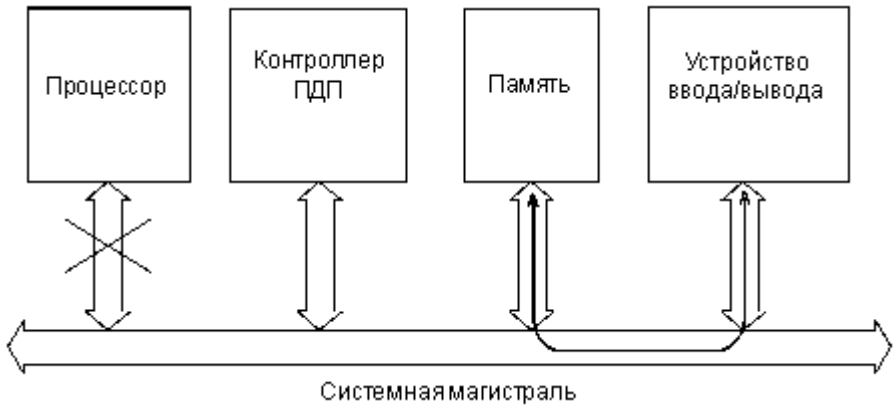

Рис. 1.14. Информационные потоки в режиме ПДП.

В принципе контроллер ПДП может входить в состав устройства ввода/вывода, которому необходим режим ПДП или даже в состав нескольких устройств ввода/вывода. Теоретически обмен с помощью прямого доступа к памяти может обеспечить более высокую скорость передачи информации, чем программный обмен, так как процессор передает данные медленнее, чем специализированный контроллер ПДП. Однако на практике это преимущество реализуется далеко не всегда. Скорость обмена в режиме ПДП обычно ограничена возможностями магистрали. К тому же необходимость программного задания режимов контроллера ПДП может свести на нет выигрыш от более высокой скорости пересылки данных в режиме ПДП. Поэтому режим ПДП применяется редко.

Если в системе уже имеется самостоятельный контроллер ПДП, то это может в ряде случаев существенно упростить аппаратуру устройств ввода/вывода, работающих в режиме ПДП. В этом, пожалуй, состоит единственное бесспорное преимущество режима ПДП.

1.4. Архитектура микропроцессорных систем

До сих пор мы рассматривали только один тип архитектуры микропроцессорных систем — архитектуру с общей, единой шиной для данных и команд (одношинную, или **принстонскую**, **фон-неймановскую** архитектуру). Соответственно, в составе системы в этом случае присутствует одна общая память, как для данных, так и для команд (рис. 1.15).

Рис. 1.15. Архитектура с общей шиной данных и команд.

Но существует также и альтернативный тип архитектуры микропроцессорной системы — это архитектура с раздельными шинами данных и команд (двухшинная, или гарвардская, архитектура). Эта архитектура предполагает наличие в системе отдельной памяти для данных и отдельной памяти для команд (рис. 1.16). Обмен процессора с каждым из двух типов памяти происходит по своей шине.

Архитектура с общей шиной распространена гораздо больше, она применяется, например, в персональных компьютерах и в сложных микрокомпьютерах. Архитектура с раздельными шинами применяется в основном в однокристальных микроконтроллерах.

Рассмотрим некоторые достоинства и недостатки обоих архитектурных решений.

Архитектура с общей шиной (принстонская, фон-неймановская) проще, она не требует от процессора одновременного обслуживания двух шин, контроля обмена по двум шинам сразу. Наличие единой памяти данных и команд позволяет гибко распределять ее объем между кодами данных и команд. Например, в некоторых случаях нужна большая и сложная программа, а данных в памяти надо хранить не слишком много. В других случаях, наоборот, программа требуется простая, но необходимы большие объемы хранимых данных. Перераспределение памяти не вызывает никаких проблем, главное — чтобы программа и данные вместе помещались в памяти системы. Как правило, в системах с такой архитектурой память бывает довольно большого объема (до десятков и сотен мегабайт). Это позволяет решать самые сложные задачи.

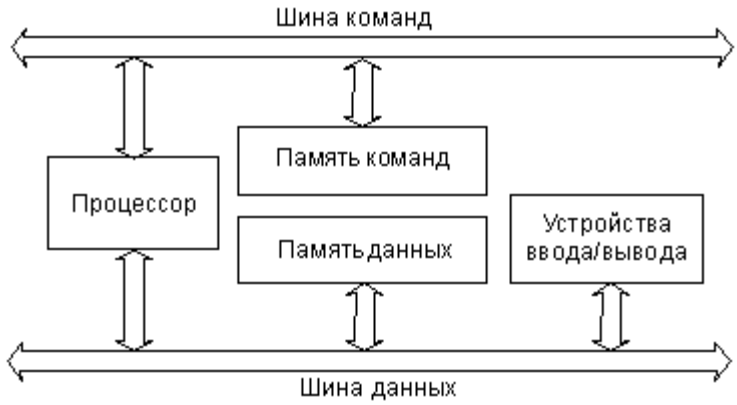

Рис. 1.16. Архитектура с раздельными шинами данных и команд.

Архитектура с раздельными шинами данных и команд сложнее, она заставляет процессор работать одновременно с двумя потоками кодов, обслуживать обмен по двум шинам одновременно. Программа может размещаться только в памяти команд, данные — только в памяти данных. Такая узкая специализация ограничивает круг задач, решаемых системой, так как не дает возможности гибкого перераспределения памяти. Память данных и память команд в этом случае имеют не слишком большой объем, поэтому применение систем с данной архитектурой ограничивается обычно не слишком сложными задачами.

В чем же преимущество архитектуры с двумя шинами (гарвардской)? В первую очередь, в быстродействии.

Дело в том, что при единственной шине команд и данных процессор вынужден по одной этой шине принимать данные (из памяти или устройства ввода/вывода) и передавать данные (в память или в устройство ввода/вывода), а также читать команды из памяти. Естественно, одновременно эти пересылки кодов по магистрали происходить не могут, они должны производиться по очереди. Современные процессоры способны совместить во времени выполнение команд и проведение циклов обмена по системной шине. Использование конвейерных технологий и быстрой кэш-памяти позволяет им ускорить процесс взаимодействия со сравнительно медленной системной памятью. Повышение тактовой частоты и совершенствование структуры процессоров дают возможность сократить время выполнения команд. Но дальнейшее увеличение быстродействия системы возможно только при совмещении пересылки данных и чтения команд, то есть при переходе к архитектуре с двумя шинами.

В случае двухшинной архитектуры обмен по обеим шинам может быть независимым, параллельным во времени. Соответственно, структуры шин (количество разрядов кода адреса и кода данных, порядок и скорость обмена информацией и т.д.) могут быть выбраны оптимально для той задачи, которая решается каждой шиной. Поэтому при прочих равных условиях переход на двухшинную архитектуру ускоряет работу микропроцессорной системы, хотя и требует дополнительных затрат на аппаратуру, усложнения структуры процессора. Память данных в этом случае имеет свое распределение адресов, а память команд — свое.

Проще всего преимущества двухшинной архитектуры реализуются внутри одной микросхемы. В этом случае можно также существенно уменьшить влияние недостатков этой архитектуры. Поэтому основное ее применение — в микроконтроллерах, от которых не требуется решения слишком сложных задач, но зато необходимо максимальное быстродействие при заданной тактовой частоте.

1.5. Типы микропроцессорных систем

Диапазон применения микропроцессорной техники сейчас очень широк, требования к микропроцессорным системам предъявляются самые разные. Поэтому сформировалось несколько типов микропроцессорных систем, различающихся мощностью, универсальностью, быстродействием и структурными отличиями. Основные типы следующие:

- микроконтроллеры — наиболее простой тип микропроцессорных систем, в которых все или большинство узлов системы выполнены в виде одной микросхемы;
- контроллеры — управляющие микропроцессорные системы, выполненные в виде отдельных модулей;
- микрокомпьютеры — более мощные микропроцессорные системы с развитыми средствами сопряжения с внешними устройствами.
- компьютеры (в том числе персональные) — самые мощные и наиболее универсальные микропроцессорные системы.

Четкую границу между этими типами иногда провести довольно сложно. Быстродействие всех типов микропроцессоров постоянно растет, и нередки ситуации, когда новый микроконтроллер оказывается быстрее, например, устаревшего персонального компьютера. Но кое-какие принципиальные отличия все-таки имеются.

Микроконтроллеры представляют собой универсальные устройства, которые практически всегда используются не сами по себе, а в составе более сложных устройств, в том числе и контроллеров. Системная шина микроконтроллера скрыта от пользователя внутри микросхемы. Возможности подключения внешних устройств к микроконтроллеру ограничены. Устройства на микроконтроллерах обычно предназначены для решения одной задачи.

Контроллеры, как правило, создаются для решения какой-то отдельной задачи или группы близких задач. Они обычно не имеют возможностей подключения дополнительных узлов и устройств, на-

пример, большой памяти, средств ввода/вывода. Их системная шина чаще всего недоступна пользователю. Структура контроллера проста и оптимизирована под максимальное быстродействие. В большинстве случаев выполняемые программы хранятся в постоянной памяти и не меняются. Конструктивно контроллеры выпускаются в одноплатном варианте.

Микрокомпьютеры отличаются от контроллеров более открытой структурой, они допускают подключение к системной шине нескольких дополнительных устройств. Производятся микрокомпьютеры в каркасе, корпусе с разъемами системной магистрали, доступными пользователю. Микрокомпьютеры могут иметь средства хранения информации на магнитных носителях (например, магнитные диски) и довольно развитые средства связи с пользователем (видеомонитор, клавиатура). Микрокомпьютеры рассчитаны на широкий круг задач, но в отличие от контроллеров, к каждой новой задаче его надо приспосабливать заново. Выполняемые микрокомпьютером программы можно легко менять.

Наконец, компьютеры и самые распространенные из них — персональные компьютеры — это самые универсальные из микропроцессорных систем. Они обязательно предусматривают возможность модернизации, а также широкие возможности подключения новых устройств. Их системная шина, конечно, доступна пользователю. Кроме того, внешние устройства могут подключаться к компьютеру через несколько встроенных портов связи (количество портов доходит иногда до 10). Компьютер всегда имеет сильно развитые средства связи с пользователем, средства длительного хранения информации большого объема, средства связи с другими компьютерами по информационным сетям. Области применения компьютеров могут быть самыми разными: математические расчеты, обслуживание доступа к базам данных, управление работой сложных электронных систем, компьютерные игры, подготовка документов и т.д.

Любую задачу в принципе можно выполнить с помощью каждого из перечисленных типов микропроцессорных систем. Но при выборе типа надо по возможности избегать избыточности и предусматривать необходимую для данной задачи гибкость системы.

В настоящее время при разработке новых микропроцессорных систем чаще всего выбирают путь использования микроконтроллеров (примерно в 80% случаев). При этом микроконтроллеры применяются или самостоятельно, с минимальной дополнительной аппаратурой, или в составе более сложных контроллеров с развитыми средствами ввода/вывода.

Классические микропроцессорные системы на базе микросхем процессоров и микропроцессорных комплектов выпускаются сейчас довольно редко, в первую очередь, из-за сложности процесса разработки и отладки этих систем. Данный тип микропроцессорных систем выбирают в основном тогда, когда микроконтроллеры не могут обеспечить требуемых характеристик.

Наконец, заметное место занимают сейчас микропроцессорные системы на основе персонального компьютера. Разработчику в этом случае нужно только оснастить персональный компьютер дополнительными устройствами сопряжения, а ядро микропроцессорной системы уже готово. Персональный компьютер имеет развитые средства программирования, что существенно упрощает задачу разработчика. К тому же он может обеспечить самые сложные алгоритмы обработки информации. Основные недостатки персонального компьютера — большие размеры корпуса и аппаратурная избыточность для простых задач. Недостатком является и неприспособленность большинства персональных компьютеров к работе в сложных условиях (запыленность, высокая влажность, вибрации, высокие температуры и т.д.). Однако выпускаются и специальные персональные компьютеры, приспособленные к различным условиям эксплуатации.

Глава 2. Организация обмена информацией

2. Лекция: Шины микропроцессорной системы и циклы обмена

В этой лекции речь идет об обмене информацией по шинам микропроцессорных систем, о циклах обмена информацией и их фазах, о принципах синхронизации обмена, принципах организации прерываний и ПДП.

Самое главное, что должен знать разработчик микропроцессорных систем — это принципы организации обмена информацией по шинам таких систем. Без этого невозможно разработать аппаратную часть системы, а без аппаратной части не будет работать никакое программное обеспечение.

За более чем 30 лет, прошедших с момента появления первых микропроцессоров, были выработаны определенные правила обмена, которым следуют и разработчики новых микропроцессорных систем. Правила эти не слишком сложны, но твердо знать и неукоснительно соблюдать их для успешной работы необходимо. Как показала практика, принципы организации обмена по шинам гораздо важнее, чем особенности конкретных микропроцессоров. Стандартные системные магистрали живут гораздо дольше, чем тот или иной процессор. Разработчики новых процессоров ориентируются на уже существующие стандарты магистрали. Более того, некоторые системы на основе совершенно разных процессоров используют одну и ту же системную магистраль. То есть магистраль оказывается самым главным системообразующим фактором в микропроцессорных системах.

Обмен информацией в микропроцессорных системах происходит в циклах обмена информацией. Под циклом обмена информацией понимается временной интервал, в течение которого происходит выполнение одной элементарной операции обмена по шине. Например, пересылка кода данных из процессора в память или же пересылка кода данных из устройства ввода/вывода в процессор. В пределах одного цикла также может передаваться и несколько кодов данных, даже целый массив данных, но это встречается реже.

Циклы обмена информацией делятся на два основных типа:

- Цикл записи (вывода), в котором процессор записывает (выводит) информацию;
- Цикл чтения (ввода), в котором процессор читает (вводит) информацию.

В некоторых микропроцессорных системах существует также цикл "чтение-модификация-запись" или же "ввод-пауза-вывод". В этих циклах процессор сначала читает информацию из памяти или устройства ввода/вывода, затем как-то преобразует ее и снова записывает по тому же адресу. Например, процессор может прочитать код из ячейки памяти, увеличить его на единицу и снова записать в эту же ячейку памяти. Наличие или отсутствие данного типа цикла связано с особенностями используемого процессора.

Особое место занимают циклы прямого доступа к памяти (если режим ПДП в системе предусмотрен) и циклы запроса и предоставления прерывания (если прерывания в системе есть). Когда в дальнейшем речь пойдет о таких циклах, это будет специально оговорено.

Во время каждого цикла устройства, участвующие в обмене информацией, передают друг другу информационные и управляющие сигналы в строго установленном порядке или, как еще говорят, в соответствии с принятым **протоколом обмена информацией**.

Длительность цикла обмена может быть постоянной или переменной, но она всегда включает в себя несколько периодов сигнала тактовой частоты системы. То есть даже в идеальном случае частота чтения информации процессором и частота записи информации оказываются в несколько раз меньше тактовой частоты системы.

Чтение кодов команд из памяти системы также производится с помощью циклов чтения. Поэтому в случае одношинной архитектуры на системной магистрали чередуются циклы чтения команд и циклы пересылки (чтения и записи) данных, но протоколы обмена остаются неизменными независимо от того, что передается — данные или команды. В случае двухшинной архитектуры циклы чтения команд и записи или чтения данных разделяются по разным шинам и могут выполняться одновременно.

2.1. Шины микропроцессорной системы

Прежде чем переходить к особенностям циклов обмена, остановимся подробнее на составе и назначении различных шин микропроцессорной системы.

Как уже упоминалось, в системную магистраль (системную шину) микропроцессорной системы входит три основные информационные шины: адреса, данных и управления.

Шина данных — это основная шина, ради которой и создается вся система. Количество ее разрядов (линий связи) определяет скорость и эффективность информационного обмена, а также максимально возможное количество команд.

Шина данных всегда двунаправленная, так как предполагает передачу информации в обоих направлениях. Наиболее часто встречающийся тип выходного каскада для линий этой шины — выход с тремя состояниями.

Обычно шина данных имеет 8, 16, 32 или 64 разряда. Понятно, что за один цикл обмена по 64-разрядной шине может передаваться 8 байт информации, а по 8-разрядной — только один байт. Разрядность шины данных определяет и разрядность всей магистрали. Например, когда говорят о 32-разрядной системной магистрали, подразумевается, что она имеет 32-разрядную шину данных.

Шина адреса — вторая по важности шина, которая определяет максимально возможную сложность микропроцессорной системы, то есть допустимый объем памяти и, следовательно, максимально возможный размер программы и максимально возможный объем запоминаемых данных. Количество адресов, обеспечиваемых шиной адреса, определяется как 2^N, где N — количество разрядов. Например, 16-разрядная шина адреса обеспечивает 65 536 адресов. Разрядность шины адреса обычно кратна 4 и может достигать 32 и даже 64. Шина адреса может быть однонаправленной (когда магистралью всегда управляет только процессор) или двунаправленной (когда процессор может временно передавать управление магистралью другому устройству, например контроллеру ПДП). Наиболее часто используются типы выходных каскадов с тремя состояниями или обычные ТТЛ (с двумя состояниями).

Как в шине данных, так и в шине адреса может использоваться **положительная логика** или **отрицательная логика**. При положительной логике высокий уровень напряжения соответствует логической единице на соответствующей линии связи, низкий — логическому нулю. При отрицательной логике — наоборот. В большинстве случаев уровни сигналов на шинах — ТТЛ.

Для снижения общего количества линий связи магистрали часто применяется мультиплексирование шин адреса и данных. То есть одни и те же линии связи используются в разные моменты времени для передачи как адреса, так и данных (в начале цикла — адрес, в конце цикла — данные). Для фиксации этих моментов (стробирования) служат специальные сигналы на шине управления. Понятно, что мультиплексированная шина адреса/данных обеспечивает меньшую скорость обмена, требует более длительного цикла обмена (рис. 2.1). По типу шины адреса и шины данных все магистрали также делятся на мультиплексированные и немультиплексированные.

Рис. 2.1. Мультиплексирование шин адреса и данных.

В некоторых мультиплексированных магистралях после одного кода адреса передается несколько кодов данных (массив данных). Это позволяет существенно повысить быстродействие магистрали. Иногда в магистралях применяется частичное мультиплексирование, то есть часть разрядов данных передается по немультиплексированным линиям, а другая часть — по мультиплексированным с адресом линиям.

Шина управления — это вспомогательная шина, управляющие сигналы на которой определяют тип текущего цикла и фиксируют моменты времени, соответствующие разным частям или стадиям цикла. Кроме того, управляющие сигналы обеспечивают согласование работы процессора (или другого хозяина магистрали, задатчика, master) с работой памяти или устройства ввода/вывода (устройства-исполнителя, slave). Управляющие сигналы также обслуживают запрос и предоставление прерываний, запрос и предоставление прямого доступа.

Сигналы шины управления могут передаваться как в положительной логике (реже), так и в отрицательной логике (чаще). Линии шины управления могут быть как однонаправленными, так и двунаправленными. Типы выходных каскадов могут быть самыми разными: с двумя состояниями (для однонаправленных линий), с тремя состояниями (для двунаправленных линий), с открытым коллектором (для двунаправленных и мультиплексированных линий).

Самые главные управляющие сигналы — это стробы обмена, то есть сигналы, формируемые процессором и определяющие моменты времени, в которые производится пересылка данных по шине данных, обмен данными. Чаще всего в магистрали используются два различных строба обмена:

- Строб записи (вывода), который определяет момент времени, когда устройство-исполнитель может принимать данные, выставленные процессором на шину данных;
- Строб чтения (ввода), который определяет момент времени, когда устройство-исполнитель должно выдать на шину данных код данных, который будет прочитан процессором.

При этом большое значение имеет то, как процессор заканчивает обмен в пределах цикла, в какой момент он снимает свой строб обмена. Возможны два пути решения (рис. 2.2):

- При синхронном обмене процессор заканчивает обмен данными самостоятельно, через раз и навсегда установленный временной интервал выдержки ($t_{выд}$), то есть без учета интересов устройства-исполнителя;
- При асинхронном обмене процессор заканчивает обмен только тогда, когда устройство-исполнитель подтверждает выполнение операции специальным сигналом (так называемый режим handshake — рукопожатие).

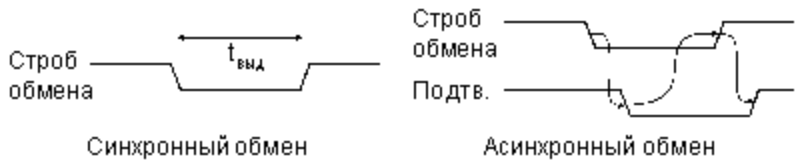

Рис. 2.2. Синхронный обмен и асинхронный обмен.

Достоинства синхронного обмена — более простой протокол обмена, меньшее количество управляющих сигналов. Недостатки — отсутствие гарантии, что исполнитель выполнил требуемую операцию, а также высокие требования к быстродействию исполнителя.

Достоинства асинхронного обмена — более надежная пересылка данных, возможность работы с самыми разными по быстродействию исполнителями. Недостаток — необходимость формирования сигнала подтверждения всеми исполнителями, то есть дополнительные аппаратурные затраты.

Какой тип обмена быстрее, синхронный или асинхронный? Ответ на этот вопрос неоднозначен. С одной стороны, при асинхронном обмене требуется какое-то время на выработку, передачу дополнительного сигнала и на его обработку процессором. С другой стороны, при синхронном обмене приходится искусственно увеличивать длительность строба обмена для соответствия требованиям большего числа исполнителей, чтобы они успевали обмениваться информацией в темпе процессора. Поэтому иногда в магистрали предусматривают возможность как синхронного, так и асинхронного обмена, причем синхронный обмен является основным и довольно быстрым, а асинхронный применяется только для медленных исполнителей.

По используемому типу обмена магистрали микропроцессорных систем также делятся на синхронные и асинхронные.

2.2. Циклы обмена информацией

2.2.1. Циклы программного обмена

Рассмотрим для примера два довольно типичных случая программного обмена по магистрали микропроцессорной системы.

Первый пример — это обмен по мультиплексированной асинхронной магистрали Q-bus, предложенной фирмой DEC и широко применявшейся в микрокомпьютерах и промышленных контроллерах. Упрощенные временные диаграммы циклов чтения (ввода) и записи (вывода) по этой магистрали приведены на рис. 2.3 и 2.4.

Отметим, что в дальнейшем тексте знак "минус" перед названием сигнала говорит о том, что активный уровень сигнала низкий, пассивный — высокий, то есть сигнал отрицательный. Если минуса перед названием сигнала нет, то сигнал положительный, его низкий уровень пассивный, а высокий — активный.

На шине адреса/данных (AD) в начале цикла обмена (в фазе адреса) процессор (задатчик) выставляет код адреса. На этой шине используется отрицательная логика. Средний уровень сигналов на шине AD обозначает, что состояния сигналов на шине в данные временные интервалы не важны. Для стробирования адреса используется отрицательный синхросигнал -SYNC, выставляемый также процессором. Его передний (отрицательный) фронт соответствует действительности кода адреса на шине AD. Фаза адреса одинакова в обоих циклах записи и чтения.

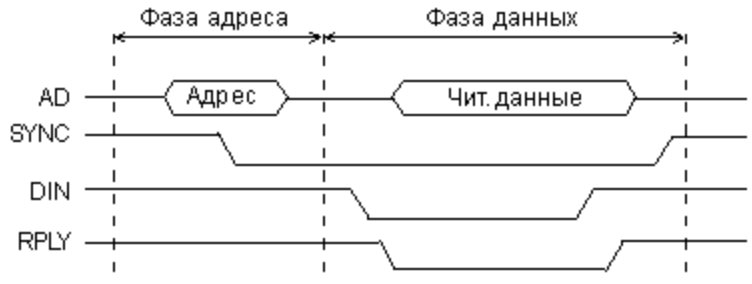

Рис. 2.3. Цикл чтения на магистрали Q-bus.

Получив (распознав) свой код адреса, устройство ввода/вывода или память (исполнитель) готовится к проведению обмена. Через некоторое время после начала (отрицательного фронта) сигнала -SYNC процессор снимает адрес и начинает фазу данных.

Рис. 2.4. Цикл записи на магистрали Q-bus.

В фазе данных цикла чтения (рис. 2.3) процессор выставляет сигнал строба чтения данных -DIN, в ответ на который устройство, к которому обращается процессор (исполнитель), должно выставить свой код данных (читаемые данные). Одновременно это устройство должно подтвердить выполнение операции сигналом подтверждения обмена -RPLY.

Для сигнала -RPLY используется тип выходного каскада ОК, чтобы не было конфликтов между устройствами-исполнителями. Процессор, получив сигнал -RPLY, заканчивает цикл обмена. Для этого он снимает сигнал -DIN и сигнал -SYNC. Устройство-исполнитель в ответ на снятие сигнала -DIN должно снять код данных с шины AD и закончить сигнал подтверждения -RPLY. После этого процессор снимает сигнал -SYNC.

В фазе данных цикла записи (рис. 2.4) процессор выставляет на шину AD код записываемых данных и сопровождает его отрицательным сигналом строба записи данных -DOUT. Устройство-исполнитель должно по этому сигналу принять данные от процессора и сформировать сигнал подтверждения обмена -RPLY. Процессор, получив сигнал -RPLY, заканчивает цикл обмена. Для этого он снимает код данных с шины AD и сигнал -DOUT. Устройство-исполнитель в ответ на снятие сигнала -DOUT должно закончить сигнал подтверждения -RPLY. После этого процессор снимает сигнал -SYNC.

То есть на данной магистрали адрес передается синхронно (без подтверждения его получения исполнителем), а данные передаются асинхронно, с обязательным подтверждением их выдачи или приема исполнителем. Отсутствие сигнала подтверждения -RPLY в течение заданного времени воспринимается процессором как аварийная ситуация. В принципе возможна и асинхронная передача адреса, что увеличивает надежность обмена, хотя может снижать его скорость.

Помимо циклов чтения и записи на магистрали Q-bus используются также и циклы типа "ввод-пауза-вывод" ("чтение-модификация-запись"). Упрощенная временная диаграмма этого цикла представлена на рис. 2.5.

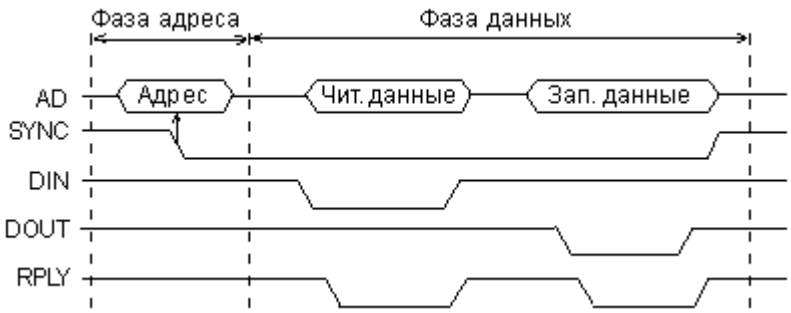

Рис. 2.5. Цикл "ввод-пауза-вывод" на магистрали Q-bus.

В этом цикле адресная фаза производится точно так же, как и в циклах чтения (ввода) и записи (вывода). Но в фазе данных процессор производит сначала чтение из заданного в адресной фазе адреса, а потом запись в тот же самый адрес. Для чтения используется строб чтения -DIN, а для записи – строб записи -DOUT. В ответ на сигнал -DIN устройство-исполнитель выдает свои данные на шину AD, а по сигналу -DOUT – принимает данные с шины AD. Как и в циклах чтения и записи, устройство-исполнитель подтверждает выполнение каждой операции сигналом подтверждения -RPLY. Понятно, что цикл "ввод-пауза-вывод" требует больше времени, чем каждый из циклов чтения или записи, но меньше времени, чем два последовательно произведенных цикла чтения и записи (так как для него нужна только одна адресная фаза). Сигнал -SYNC вырабатывается процессором в начале цикла "ввод-пауза-вывод" и держится до окончания всего цикла.

В качестве второго примера рассмотрим циклы обмена на синхронной немультиплексированной магистрали ISA (Industrial Standard Architecture), предложенной фирмой IBM и широко используемой в персональных компьютерах. Упрощенные циклы записи в устройство ввода/вывода и чтения из устройства ввода/вывода приведены на рис. 2.6 и 2.7.

Оба цикла начинаются с выставления процессором (задатчиком) кода адреса на шину адреса SA (логика на этой шине положительная). Адрес остается на шине SA до конца цикла. Фаза адреса, одинаковая для обоих циклов, заканчивается с началом строба обмена данными -IOR или -IOW. В течение фазы адреса устройство-исполнитель должно принять код адреса и распознать или не распознать его. Если адрес распознан, исполнитель готовится к обмену.

В фазе данных цикла чтения (рис. 2.6) процессор выставляет отрицательный сигнал чтения данных из устройства ввода/вывода -IOR. В ответ на него устройство-исполнитель должно выдать на шину данных SD свой код данных (читаемые данные). Логика на шине данных положительная. Через установленное время строб обмена -IOR снимается процессором, после чего снимается также и код адреса с шины SA. Цикл заканчивается без учета быстродействия исполнителя.

Рис. 2.6. Цикл чтения из УВВ на магистрали ISA.

Рис. 2.7. Цикл записи в УВВ на магистрали ISA.

Но так происходит только в случае основного, синхронного обмена. Кроме него на магистрали ISA также предусмотрена возможность асинхронного обмена. Для этого применяется сигнал готовности канала (магистрали) I/O CH RDY. Тип выходного каскада для данного сигнала — ОК, для предот-

вращения конфликтов между устройствами-исполнителями. При синхронном обмене сигнал I/O CH RDY всегда положительный. Но медленное устройство-исполнитель, не успевающее работать в темпе процессора, может этот сигнал снять, то есть сделать нулевым сразу после начала строба обмена. Тогда процессор до того момента, пока сигнал I/O CH RDY не станет снова положительным, приостанавливает завершение цикла, продлевает строб обмена. Конечно, слишком большая длительность этого сигнала рассматривается как аварийная ситуация. Для простоты понимания можно считать, что устройство-исполнитель формирует в данном случае отрицательный сигнал неготовности завершить обмен. На время этого сигнала обмен на магистрали приостанавливается.

Принципиальное отличие асинхронного обмена по магистрали ISA от асинхронного обмена по магистрали Q-bus состоит в следующем. Если в случае Q-bus сигнал подтверждения обязателен, и его должен формировать каждый исполнитель, то в случае ISA сигнал о неготовности исполнитель может не формировать, если он успевает работать в темпе процессора. Зато в случае Q-bus к концу цикла обмена процессор всегда уверен, что устройство-исполнитель выполнило требуемую операцию, а в случае ISA такой уверенности нет.

В фазе данных цикла записи по магистрали ISA (рис. 2.7) процессор выставляет на шину данных SD код записываемых данных и сопровождает их стробом записи данных в устройство ввода/вывода -IOW. Получив этот сигнал, устройство-исполнитель должно принять с шины SD код записываемых данных. Если оно не успевает сделать это в темпе процессора, то оно может снять на нужное время сигнал I/O CH RDY после получения переднего фронта сигнала -IOW. Тогда процессор приостановит окончание цикла записи.

Рассмотренные примеры, конечно, не раскрывают всех тонкостей обмена по упомянутым магистралям. Они всего лишь иллюстрируют главные принципы обмена по ним.

2.2.2. Циклы обмена по прерываниям

Циклы обмена в режиме прерываний строятся по тем же принципам, что и циклы программного обмена, но имеют ряд специфических особенностей.

Прерывания в микропроцессорных системах бывают двух основных типов:

- векторные прерывания, которые требуют проведения цикла чтения по магистрали;
- радиальные прерывания, которые не требуют никакого цикла обмена по магистрали.

Дело в том, что прерываний в микропроцессорной системе обычно бывает много. Поэтому процессору необходима информация о номере (или, как еще говорят, об адресе вектора) конкретного прерывания. Эта информация может быть передана процессору двумя путями.

При векторном прерывании код номера прерывания передается процессору тем устройством ввода/вывода, которое данное прерывание запросило. Для этого процессор проводит цикл чтения по магистрали, и по шине данных получает код номера прерывания. Шина адреса в данном цикле обычно не используется, так как устройство, запросившее прерывание, и так знает, что процессор будет обращаться именно к нему. В этом случае в магистрали достаточно всего одной линии запроса прерывания для всех устройств ввода/вывода. Так организованы прерывания, например, в магистрали Q-bus.

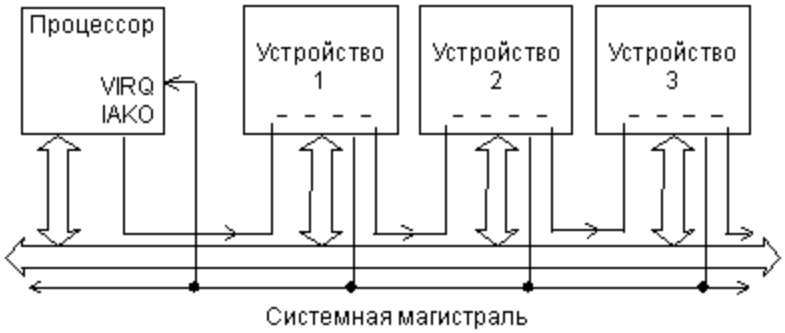

Рис. 2.8. Сигналы запроса и предоставления прерывания в магистрали Q-bus.

Схема распространения сигналов, участвующих в прерываниях на магистрали Q-bus, показана на рис. 2.8. Упрощенная временная диаграмма цикла запроса и предоставления магистрали представлена на рис. 2.9.

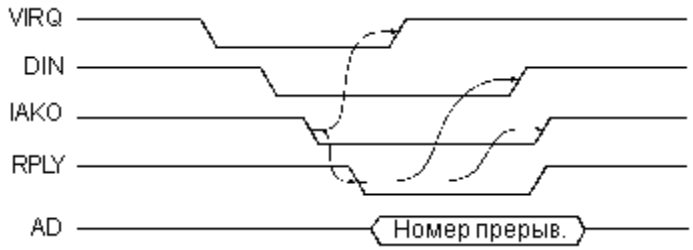

Рис. 2.9. Цикл запроса/предоставления векторного прерывания на магистрали Q-bus.

Запрос прерывания осуществляется отрицательным сигналом -VIRQ, который может формироваться каждым из устройств, запрашивающих прерывание. Тип выходного каскада для этого сигнала — ОК, чтобы избежать конфликтов между запрашивающими прерывания устройствами. Получив сигнал -VIRQ, процессор предоставляет прерывание (закончив предварительно выполнение текущей команды). Для этого он выставляет сигнал чтения данных -DIN и сигнал предоставления прерывания IAKO. Этот сигнал IAKO последовательно проходит через все устройства, которые могут запрашивать прерывания. Если устройство запросило прерывание, то оно не пропускает через себя этот сигнал. В результате получается, что если прерывания одновременно запросили два или более устройств, то сигнал предоставления прерывания получит только одно устройство, а именно то, которое ближе к процессору. Такой механизм разрешения конфликтов называется иногда географическим приоритетом (или цепочечным приоритетом, Daisy Chain). Получив сигнал IAKO, устройство, запросившее прерывание, должно снять свой сигнал -VIRQ.

Затем процессор проводит цикл безадресного чтения номера прерывания. В ответ на полученные сигналы -DIN и IAKO устройство, которому предоставлено прерывание, должно выдать на шину адреса/данных AD код номера прерывания (адреса вектора прерывания) и выставить сигнал подтверждения -RPLY. Процессор читает код номера прерывания и заканчивает цикл безадресного чтения снятием сигналов -DIN и IAKO.

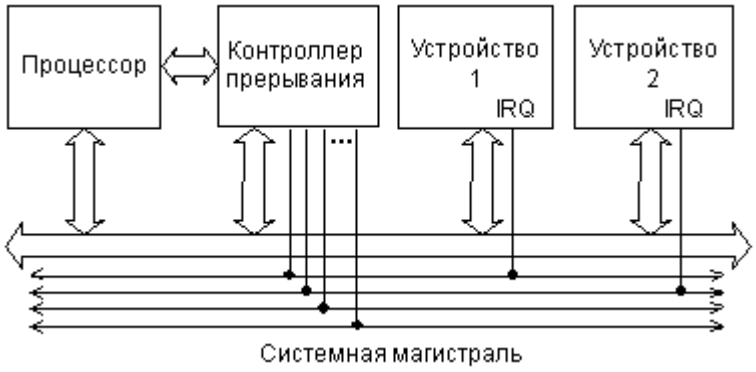

Рис. 2.10. Структура связей для организации радиальных прерываний на магистрали ISA.

При радиальном прерывании в магистрали имеется столько линий запроса прерывания, сколько всего может быть разных прерываний. То есть каждое устройство ввода/вывода, желающее использовать прерывание, подает сигнал запроса прерывания по своей отдельной линии. Процессор узнает о номере прерывания по номеру линии, по которой пришел сигнал запроса прерывания. Никаких циклов обмена по магистрали при этом не требуется. В случае радиальных прерываний в систему обычно включается дополнительная микросхема контроллера прерываний, обрабатывающая сигналы запросов прерываний. Именно так организованы прерывания, например, в магистрали ISA.

Упрощенная структура связей между устройствами, участвующими в обмене по прерываниям, на магистрали ISA показана на рис. 2.10. Процессор общается с контроллером прерываний как по магистрали (чтобы задать ему режимы работы), так и вне магистрали (при обработке запросов на прерывание). Сигналы запросов прерываний IRQ распределяются между всеми устройствами магистрали. На каждую линию IRQ приходится одно устройство. Тип выходного каскада для этих линий — 2С, так как конфликты здесь не предусмотрены. Запросом прерывания является передний, положительный фронт сигнала IRQ. При одновременном поступлении сигналов IRQ от нескольких устройств порядок их обслуживания определяется контроллером прерываний.

Какой тип прерываний лучше — векторный или радиальный?

Векторные прерывания обеспечивают системе большую гибкость, в системе их может быть очень много. Но зато они требуют дополнительных аппаратурных узлов во всех устройствах, запрашивающих прерывания, для обслуживания циклов безадресного чтения.

Радиальных прерываний в системе обычно не очень много (от 1 до 16). При этом типе прерываний, как правило, требуется введение в систему специального контроллера прерываний. Каждое радиальное прерывание требует введения дополнительной линии в шину управления системной магистрали. Но работать с радиальными прерываниями проще, так как все сводится только к выработке единственного сигнала IRQ, и никаких циклов обмена по магистрали не требуется.

2.2.3. Циклы обмена в режиме ПДП

Циклы обмена в режиме прямого доступа к памяти выполняются по тем же правилам, что и циклы программного обмена, и циклы предоставления прерываний.

Прежде чем начать обмен в режиме ПДП, устройство, которому необходим ПДП, должно запросить ПДП и получить его. Процедура запроса и предоставления ПДП очень похожа на процедуру запроса и предоставления прерывания. В обоих случаях устройство, требующее обслуживания, посылает сигнал запроса процессору. Однако в случае ПДП процессор обязательно должен предоставить ПДП запросившему устройству с помощью специальных сигналов, так как на время ПДП процессор отключается от магистрали. А при радиальных прерываниях предоставления прерывания от процессора не требуется.

На магистрали Q-bus запрос и предоставление ПДП организуются подобно запросу и предоставлению прерывания. Упрощенная структура связей устройств, участвующих в ПДП, показана на рис. 2.11. Временная диаграмма запроса/предоставления ПДП очень близка к временной диаграмме запроса/предоставления прерывания (см. рис. 2.9).

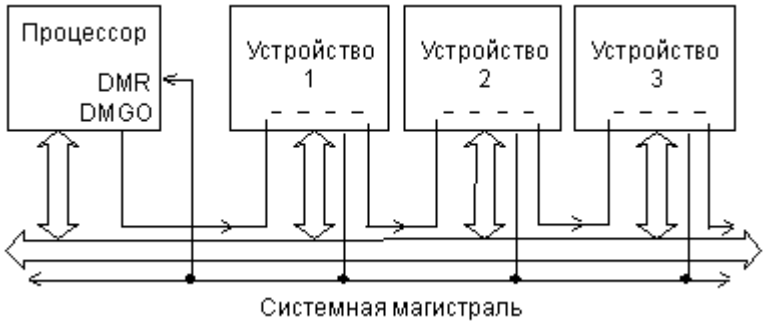

Рис. 2.11. Структура связей запроса/предоставления ПДП на магистрали Q-bus.

Сигнал запроса ПДП, называемый -DMR, передается всеми устройствами, нуждающимися в ПДП, по одной линии магистрали. Тип выходного каскада на этой линии — ОК. Процессор, получив сигнал -DMR, выдает сигнал предоставления ПДП DMGO, аналогичный сигналу IAKO. Этот сигнал также проходит через все устройства последовательно, в результате чего ПДП получает только то устройство, которое находится ближе к процессору (географический приоритет). А затем устройство, получившее ПДП, проводит циклы обмена по магистрали, аналогично циклам программного обмена. В циклах ПДП информация читается из памяти и записывается в устройство ввода/вывода, или наоборот — читается из устройства ввода/вывода и передается в память.

На магистрали ISA запрос/предоставление ПДП очень напоминает организацию радиальных прерываний (рис. 2.12). Точно так же в системе существует контроллер ПДП, к которому сходятся сигналы запроса ПДП, называемые DRQ, и от которого расходятся сигналы предоставления ПДП, называемые -DACK. К каждому каналу ПДП (пара сигналов DRQ и -DACK) подключается только одно устройство, запрашивающее ПДП. Тип выходных каскадов для этих сигналов —2С. Устройство, нуждающееся в ПДП, посылает сигнал запроса DRQ и получает в ответ сигнал предоставления -DACK. После этого контроллер ПДП проводит циклы обмена по магистрали между устройством ввода/вывода и памятью.

Упрощенная временная диаграмма циклов ПДП на магистрали ISA показана на рис. 2.13.

На магистрали ISA используются раздельные стробы записи в память (-MEMW) и записи в устройства ввода/вывода (-IOW), а также раздельные стробы чтения из памяти (-MEMR) и чтения из устройств ввода/вывода (-IOR). Это позволяет за один цикл обмена ПДП читать информацию из памяти и записывать ее в устройство ввода/вывода или же читать информацию из устройства ввода/вывода и записывать ее в память. При этом на шине адреса выставляется адрес памяти, а адрес устройства ввода/вывода заменяется одним-единственным сигналом AEN. Естественно, в цикле обмена в режиме ПДП участвует только то устройство ввода/вывода, которое предварительно запросило ПДП и которому ПДП было предоставлено. Поэтому никаких конфликтов между устройствами ввода/вывода из-за такой упрощенной адресации не возникает.

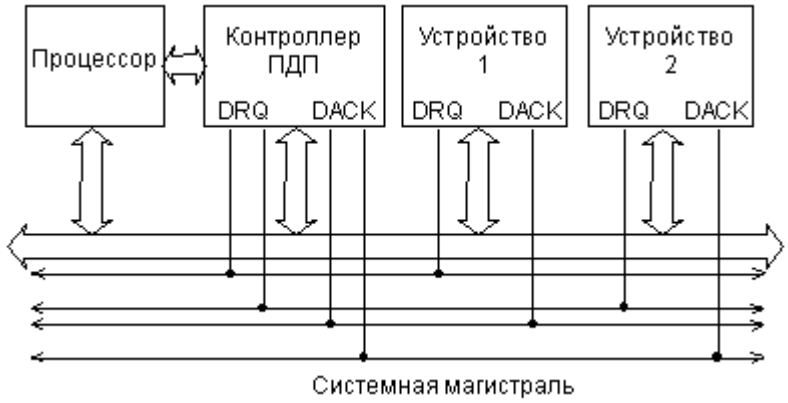

Рис. 2.12. Структура связей запроса/предоставления ПДП на магистрали ISA.

Рис. 2.13. Цикл ПДП на магистрали ISA.

2.3. Прохождение сигналов по магистрали

При организации обмена по магистралям и шинам разработчику необходимо учитывать несколько важных моментов, связанных как с особенностью распространения сигналов по шинам, так и с самой природой шин. В противном случае микропроцессорная система может попросту не работать или работать неустойчиво, хотя вся логика цифровых устройств, входящих в систему, будет спроектирована безошибочно.

В случае, когда системная шина (магистраль) микропроцессорной системы является внешней, а не скрыта внутри микросхемы, необходимо учитывать особенности распространения сигналов по длинным линиям. Хотя в большинстве случаев длина магистрали не слишком велика, не превышает 1—2 десятков сантиметров, это все равно оказывает большое влияние на синхронизацию обмена.

На прохождение сигналов по магистрали влияют следующие факторы:

- конечная величина задержки распространения сигналов по линиям магистрали;
- различие задержек распространения сигналов по разным линиям шины;
- неодновременное выставление сигналов на линии шины;
- искажение фронтов сигналов, проходящих по линиям магистрали;
- отражение сигналов от концов линий связи (рис. 2.14).

Рис. 2.14. Прохождение сигналов по шине.

Для учета всех этих факторов разработчики стандартных магистралей обмена и стандартных протоколов обмена всегда закладывают необходимые задержки между сигналами, участвующими в обмене. Кроме того, задержки между сигналами выбираются таким образом, чтобы устройство, которому адресован тот или иной сигнал, имело достаточно времени для его обработки. Если разрабатывается новая магистраль, все это тоже надо учитывать.

Поэтому пытаться "модернизировать" какой-то стандартный протокол и ускорять обмен по магистрали путем уменьшения задержек, предусмотренных стандартом, очень опасно. Точно так же опасно, не изменяя протокола обмена, пытаться увеличить длину магистрали, увеличивая тем самым задержки распространения сигналов по линиям и шинам. Особенно чувствительны к такого рода "модернизациям" синхронные магистрали, в которых не предусмотрено обязательное подтверждение выполнения каждой операции.

Например, длительность фазы адреса в цикле обмена выбирается таким образом. В течение адресной фазы все сигналы всех разрядов кода адреса, пусть даже и сформированные процессором не одновременно, должны дойти до устройства-исполнителя по своим проводам шины. А устройство-исполнитель должно этот код адреса принять и обработать (то есть отличить свой адрес от чужого). Естественно, для гарантии в длительность адресной фазы еще добавляется небольшая дополнительная задержка.

Точно так же длительность фазы данных в цикле чтения должна выбираться такой, чтобы устройство-исполнитель успело получить строб чтения и выдать код читаемых данных на шину данных. Затем этот код должен успеть дойти до процессора и процессор должен успеть его прочитать. После чего процессор снимает сигнал строба чтения, этот задний фронт сигнала доходит с задержкой до устройства-исполнителя, которое также с задержкой снимает свой код данных. Аналогично и в цикле записи.

Для улучшения формы сигналов, распространяющихся по магистрали, иногда применяют оконечные согласователи (терминаторы) на концах линий магистрали. Особенно важно их применение в случае, когда допустимая длина магистрали превышает несколько метров. Например, в случае магистрали Q-bus применяются два типа согласователей: 120-омный и 250-омный (рис. 2.15).

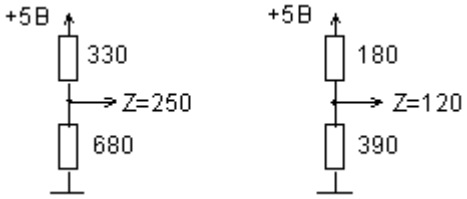

Рис. 2.15. Оконечные согласователи на магистрали Q-bus.

Включение согласователей предъявляет дополнительные требования к нагрузочной способности передатчиков, работающих на линии магистрали. В магистрали ISA подобные согласователи не ис-

пользуются, хотя к некоторым линиям присоединены резисторы, соединенные другим своим выводом с шиной питания (прежде всего это линии, тип выходного каскада для которых — ОК).

В любом случае выходные каскады передатчиков, работающих на линии магистрали, должны обеспечивать высокие выходные токи, так как к магистрали может подключаться несколько устройств, каждое из которых потребляет входной ток. Типичные величины требуемых выходных токов магистральных передатчиков находятся в пределах 20—30 мА. В то же время входные токи магистральных приемников должны быть малыми, чтобы не перегружать передатчики. Типичные величины допустимых входных токов магистральных приемников лежат в пределах 0,2—0,8 мА.

3. Лекция: Функции устройств магистрали

В этой лекции рассказывается о функциях основных устройств микропроцессорной системы: процессора, памяти, устройств ввода-вывода, о принципах их устройства и подключения к магистрали.

2.4. Функции устройств магистрали

Рассмотрим теперь, как взаимодействуют на магистрали основные устройства микропроцессорной системы: процессор, память (оперативная и постоянная), устройства ввода/вывода.

2.4.1. Функции процессора

Процессор (рис. 2.16) обычно представляет собой отдельную микросхему или же часть микросхемы (в случае микроконтроллера). В прежние годы процессор иногда выполнялся на комплектах из нескольких микросхем, но сейчас от такого подхода уже практически отказались. Микросхема процессора обязательно имеет выводы трех шин: шины адреса, шины данных и шины управления. Иногда некоторые сигналы и шины мультиплексируются, чтобы уменьшить количество выводов микросхемы процессора.

Важнейшие характеристики процессора — это количество разрядов его шины данных, количество разрядов его шины адреса и количество управляющих сигналов в шине управления. Разрядность шины данных определяет скорость работы системы. Разрядность шины адреса определяет допустимую сложность системы. Количество линий управления определяет разнообразие режимов обмена и эффективность обмена процессора с другими устройствами системы.

Кроме выводов для сигналов трех основных шин процессор всегда имеет вывод (или два вывода) для подключения внешнего тактового сигнала или кварцевого резонатора (CLK), так как процессор всегда представляет собой тактируемое устройство. Чем больше тактовая частота процессора, тем он быстрее работает, то есть тем быстрее выполняет команды. Впрочем, быстродействие процессора определяется не только тактовой частотой, но и особенностями его структуры. Современные процессоры выполняют большинство команд за один такт и имеют средства для параллельного выполнения нескольких команд. Тактовая частота процессора не связана прямо и жестко со скоростью обмена по магистрали, так как скорость обмена по магистрали ограничена задержками распространения сигналов и искажениями сигналов на магистрали. То есть тактовая частота процессора определяет только его внутреннее быстродействие, а не внешнее. Иногда тактовая частота процессора имеет нижний и верхний пределы. При превышении верхнего предела частоты возможно перегревание процессора, а также сбои, причем, что самое неприятное, возникающие не всегда и нерегулярно. Так что с изменением этой частоты надо быть очень осторожным.

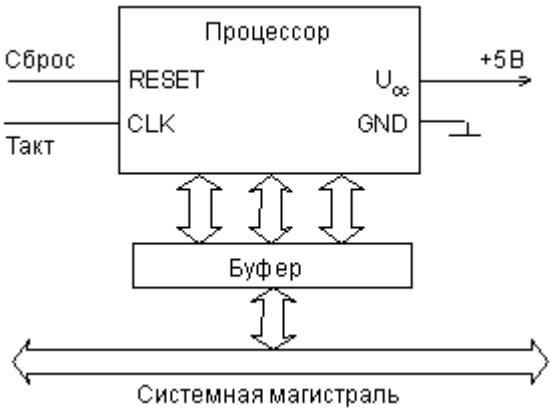

Рис. 2.16. Схема включения процессора.

Еще один важный сигнал, который имеется в каждом процессоре, — это сигнал начального сброса RESET. При включении питания, при аварийной ситуации или зависании процессора подача этого сигнала приводит к инициализации процессора, заставляет его приступить к выполнению программы начального запуска. Аварийная ситуация может быть вызвана помехами по цепям питания и "земли", сбоями в работе памяти, внешними ионизирующими излучениями и еще множеством причин. В результате процессор может потерять контроль над выполняемой программой и остановиться в каком-то адресе. Для выхода из этого состояния как раз и используется сигнал начального сброса. Этот же вход начального сброса может использоваться для оповещения процессора о том, что напряжение питания стало ниже установленного предела. В таком случае процессор переходит к выполнению программы сохранения важных данных. По сути, этот вход представляет собой особую разновидность радиального прерывания.

Иногда у микросхемы процессора имеется еще один-два входа радиальных прерываний для обработки особых ситуаций (например, для прерывания от внешнего таймера).

Шина питания современного процессора обычно имеет одно напряжение питания (+5В или +3,3В) и общий провод ("землю"). Первые процессоры нередко требовали нескольких напряжений питания. В некоторых процессорах предусмотрен режим пониженного энергопотребления. Вообще, современные микросхемы процессоров, особенно с высокими тактовыми частотами, потребляют довольно большую мощность. В результате для поддержания нормальной рабочей температуры корпуса на них нередко приходится устанавливать радиаторы, вентиляторы или даже специальные микрохолодильники.

Для подключения процессора к магистрали используются буферные микросхемы, обеспечивающие, если необходимо, демультиплексирование сигналов и электрическое буферирование сигналов магистрали. Иногда протоколы обмена по системной магистрали и по шинам процессора не совпадают между собой, тогда буферные микросхемы еще и согласуют эти протоколы друг с другом. Иногда в микропроцессорной системе используется несколько магистралей (системных и локальных), тогда для каждой из магистралей применяется свой буферный узел. Такая структура характерна, например, для персональных компьютеров.

После включения питания процессор переходит в первый адрес программы начального пуска и выполняет эту программу. Данная программа предварительно записана в постоянную (энергонезависимую) память. После завершения программы начального пуска процессор начинает выполнять основную программу, находящуюся в постоянной или оперативной памяти, для чего выбирает по очереди все команды. От этой программы процессор могут отвлекать внешние прерывания или запросы на ПДП. Команды из памяти процессор выбирает с помощью циклов чтения по магистрали. При необходимости процессор записывает данные в память или в устройства ввода/вывода с помощью циклов записи или же читает данные из памяти или из устройств ввода/вывода с помощью циклов чтения.

Таким образом, основные функции любого процессора следующие:

- выборка (чтение) выполняемых команд;
- ввод (чтение) данных из памяти или устройства ввода/вывода;
- вывод (запись) данных в память или в устройства ввода/вывода;
- обработка данных (операндов), в том числе арифметические операции над ними;
- адресация памяти, то есть задание адреса памяти, с которым будет производиться обмен;
- обработка прерываний и режима прямого доступа.

Упрощенно структуру микропроцессора можно представить в следующем виде (рис. 2.17).

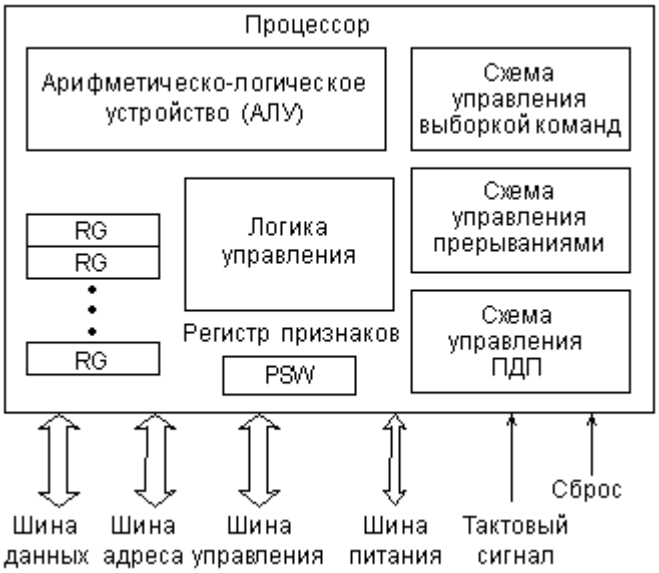

Рис. 2.17. Внутренняя структура микропроцессора.

Основные функции показанных узлов следующие.

Схема управления выборкой команд выполняет чтение команд из памяти и их дешифрацию. В первых микропроцессорах было невозможно одновременное выполнение предыдущей команды и выборка следующей команды, так как процессор не мог совмещать эти операции. Но уже в 16-разрядных процессорах появляется так называемый конвейер (очередь) команд, позволяющий выбирать несколько следующих команд, пока выполняется предыдущая. Два процесса идут параллельно, что ускоряет работу процессора. Конвейер представляет собой небольшую внутреннюю память процессора, в которую при малейшей возможности (при освобождении внешней шины) записывается несколько команд, следующих за исполняемой. Читаются эти команды процессором в том же порядке, что и записываются в конвейер (это память типа FIFO, First In — First Out, первый вошел — первый вышел). Правда, если выполняемая команда предполагает переход не на следующую ячейку памяти, а на удаленную (с меньшим или большим адресом), конвейер не помогает, и его приходится сбрасывать. Но такие команды встречаются в программах сравнительно редко.

Развитием идеи конвейера стало использование внутренней кэш-памяти процессора, которая заполняется командами, пока процессор занят выполнением предыдущих команд. Чем больше объем кэш-памяти, тем меньше вероятность того, что ее содержимое придется сбросить при команде перехода. Понятно, что обрабатывать команды, находящиеся во внутренней памяти, процессор может гораздо быстрее, чем те, которые расположены во внешней памяти. В кэш-памяти могут храниться и данные, которые обрабатываются в данный момент, это также ускоряет работу. Для большего ускорения выборки команд в современных процессорах применяют совмещение выборки и дешифрации, одновременную дешифрацию нескольких команд, несколько параллельных конвейеров команд, предсказание команд переходов и некоторые другие методы.

Арифметико-логическое устройство (или АЛУ, ALU) предназначено для обработки информации в соответствии с полученной процессором командой. Примерами обработки могут служить логические операции (типа логического "И", "ИЛИ", "Исключающего ИЛИ" и т.д.) то есть побитные операции

над операндами, а также арифметические операции (типа сложения, вычитания, умножения, деления и т.д.). Над какими кодами производится операция, куда помещается ее результат — определяется выполняемой командой. Если команда сводится всего лишь к пересылке данных без их обработки, то АЛУ не участвует в ее выполнении.

Быстродействие АЛУ во многом определяет производительность процессора. Причем важна не только частота тактового сигнала, которым тактируется АЛУ, но и количество тактов, необходимое для выполнения той или иной команды. Для повышения производительности разработчики стремятся довести время выполнения команды до одного такта, а также обеспечить работу АЛУ на возможно более высокой частоте. Один из путей решения этой задачи состоит в уменьшении количества выполняемых АЛУ команд, создание процессоров с уменьшенным набором команд (так называемые RISC-процессоры). Другой путь повышения производительности процессора — использование нескольких параллельно работающих АЛУ.

Что касается операций над числами с плавающей точкой и других специальных сложных операций, то в системах на базе первых процессоров их реализовали последовательностью более простых команд, специальными подпрограммами, однако затем были разработаны специальные вычислители — математические сопроцессоры, которые заменяли основной процессор на время выполнения таких команд. В современных микропроцессорах математические сопроцессоры входят в структуру как составная часть.

Регистры процессора представляют собой по сути ячейки очень быстрой памяти и служат для временного хранения различных кодов: данных, адресов, служебных кодов. Операции с этими кодами выполняются предельно быстро, поэтому, в общем случае, чем больше внутренних регистров, тем лучше. Кроме того, на быстродействие процессора сильно влияет разрядность регистров. Именно разрядность регистров и АЛУ называется **внутренней разрядностью процессора**, которая может не совпадать с **внешней разрядностью**.

По отношению к назначению внутренних регистров существует два основных подхода. Первого придерживается, например, компания Intel, которая каждому регистру отводит строго определенную функцию. С одной стороны, это упрощает организацию процессора и уменьшает время выполнения команды, но с другой — снижает гибкость, а иногда и замедляет работу программы. Например, некоторые арифметические операции и обмен с устройствами ввода/вывода проводятся только через один регистр — **аккумулятор**, в результате чего при выполнении некоторых процедур может потребоваться несколько дополнительных пересылок между регистрами. Второй подход состоит в том, чтобы все (или почти все) регистры сделать равноправными, как , например, в 16-разрядных процессорах T-11 фирмы DEC. При этом достигается высокая гибкость, но необходимо усложнение структуры процессора. Существуют и промежуточные решения, в частности, в процессоре MC68000 фирмы Motorola половина регистров использовалась для данных, и они были взаимозаменяемы, а другая половина — для адресов, и они также взаимозаменяемы.

Регистр признаков (регистр состояния) занимает особое место, хотя он также является внутренним регистром процессора. Содержащаяся в нем информация — это не данные, не адрес, а слово состояния процессора (ССП, PSW — Processor Status Word). Каждый бит этого слова (флаг) содержит информацию о результате предыдущей команды. Например, есть бит нулевого результата, который устанавливается в том случае, когда результат выполнения предыдущей команды — нуль, и очищается в том случае, когда результат выполнения команды отличен от нуля. Эти биты (флаги) используются командами условных переходов, например, командой перехода в случае нулевого результата. В этом же регистре иногда содержатся флаги управления, определяющие режим выполнения некоторых команд.

Схема управления прерываниями обрабатывает поступающий на процессор запрос прерывания, определяет адрес начала программы обработки прерывания (адрес вектора прерывания), обеспечивает переход к этой программе после выполнения текущей команды и сохранения в памяти (в стеке) текущего состояния регистров процессора. По окончании программы обработки прерывания процес-

сор возвращается к прерванной программе с восстановленными из памяти (из стека) значениями внутренних регистров. Подробнее о стеке будет рассказано в следующем разделе.

Схема управления прямым доступом к памяти служит для временного отключения процессора от внешних шин и приостановки работы процессора на время предоставления прямого доступа запросившему его устройству.

Логика управления организует взаимодействие всех узлов процессора, перенаправляет данные, синхронизирует работу процессора с внешними сигналами, а также реализует процедуры ввода и вывода информации.

Таким образом, в ходе работы процессора схема выборки команд выбирает последовательно команды из памяти, затем эти команды выполняются, причем в случае необходимости обработки данных подключается АЛУ. На входы АЛУ могут подаваться обрабатываемые данные из памяти или из внутренних регистров. Во внутренних регистрах хранятся также коды адресов обрабатываемых данных, расположенных в памяти. Результат обработки в АЛУ изменяет состояние регистра признаков и записывается во внутренний регистр или в память (как источник, так и приемник данных указывается в составе кода команды). При необходимости информация может переписываться из памяти (или из устройства ввода/вывода) во внутренний регистр или из внутреннего регистра в память (или в устройство ввода/вывода).

Внутренние регистры любого микропроцессора обязательно выполняют две служебные функции:

- определяют адрес в памяти, где находится выполняемая в данный момент команда (функция **счетчика команд** или **указателя команд**);
- определяют текущий адрес стека (функция указателя стека).

В разных процессорах для каждой из этих функций может отводиться один или два внутренних регистра. Эти два указателя отличаются от других не только своим специфическим, служебным, системным назначением, но и особым способом изменения содержимого. Их содержимое программы могут менять только в случае крайней необходимости, так как любая ошибка при этом грозит нарушением работы компьютера, зависанием и порчей содержимого памяти.

Содержимое указателя (счетчика) команд изменяется следующим образом. В начале работы системы (при включении питания) в него заносится раз и навсегда установленное значение. Это первый адрес программы начального запуска. Затем после выборки из памяти каждой следующей команды значение указателя команд автоматически увеличивается (инкрементируется) на единицу (или на два в зависимости от формата команд и типа процессора). То есть следующая команда будет выбираться из следующего по порядку адреса памяти. При выполнении команд перехода, нарушающих последовательный перебор адресов памяти, в указатель команд принудительно записывается новое значение — новый адрес в памяти, начиная с которого адреса команд опять же будут перебираться последовательно. Такая же смена содержимого указателя команд производится при вызове подпрограммы и возврате из нее или при начале обработки прерывания и после его окончания.

О стеке будет подробнее рассказано в следующем разделе.

2.4.2. Функции памяти

Память микропроцессорной системы выполняет функцию временного или постоянного хранения данных и команд. Объем памяти определяет допустимую сложность выполняемых системой алгоритмов, а также в некоторой степени и скорость работы системы в целом. Модули памяти выполняются на микросхемах памяти (оперативной или постоянной). Все чаще в составе микропроцессорных систем используется флэш-память (англ. — flash memory), которая представляет собой энергонезависимую память с возможностью многократной перезаписи содержимого.

Информация в памяти хранится в ячейках, количество разрядов которых равно количеству разрядов шины данных процессора. Обычно оно кратно восьми (например, 8, 16, 32, 64). Допустимое количество ячеек памяти определяется количеством разрядов шины адреса как 2^N, где N — количество разрядов шины адреса. Чаще всего объем памяти измеряется в байтах независимо от разрядности ячейки памяти. Используются также следующие более крупные единицы объема памяти: килобайт — 2^{10} или 1024 байта (обозначается Кбайт), мегабайт — 2^{20} или 1 048 576 байт (обозначается Мбайт), гигабайт — 2^{30} байт (обозначается Гбайт), терабайт — 2^{40} (обозначается Тбайт) Например, если память имеет 65 536 ячеек, каждая из которых 16-разрядная, то говорят, что память имеет объем 128 Кбайт. Совокупность ячеек памяти называется обычно **пространством памяти** системы.

Для подключения модуля памяти к системной магистрали используются блоки сопряжения, которые включают в себя дешифратор (селектор) адреса, схему обработки управляющих сигналов магистрали и буферы данных (рис. 2.18).

Оперативная память общается с системной магистралью в циклах чтения и записи, постоянная память — только в циклах чтения. Обычно в составе системы имеется несколько модулей памяти, каждый из которых работает в своей области пространства памяти. Селектор адреса как раз и определяет, какая область адресов пространства памяти отведена данному модулю памяти. Схема управления вырабатывает в нужные моменты сигналы разрешения работы памяти (CS) и сигналы разрешения записи в память (WR). Буферы данных передают данные от памяти к магистрали или от магистрали к памяти.

В пространстве памяти микропроцессорной системы обычно выделяются несколько особых областей, которые выполняют специальные функции.

Память программы начального запуска всегда выполняется на ПЗУ или флэш-памяти. Именно с этой области процессор начинает работу после включения питания и после сброса его с помощью сигнала RESET.

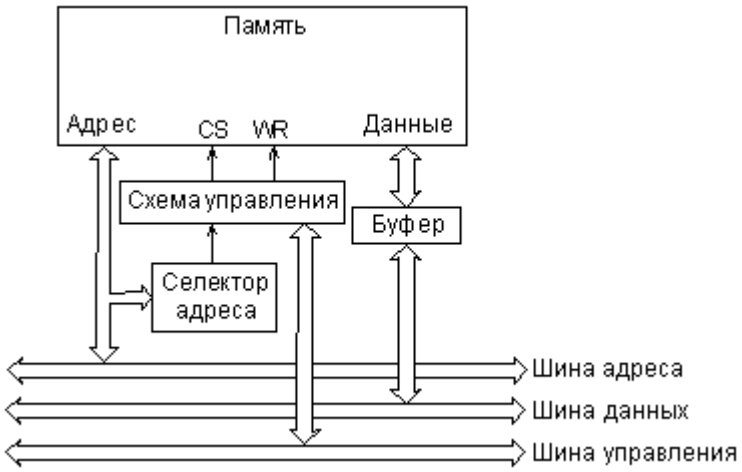

Рис. 2.18. Структура модуля памяти.

Память для стека или стек (Stack) — это часть оперативной памяти, предназначенная для временного хранения данных в режиме LIFO (Last In — First Out).

Особенность стека по сравнению с другой оперативной памятью — это заданный и неизменяемый способ адресации. При записи любого числа (кода) в стек число записывается по адресу, определяемому как содержимое регистра указателя стека, предварительно уменьшенное (декрементированное) на единицу (или на два, если 16-разрядные слова расположены в памяти по четным адресам). При чтении из стека число читается из адреса, определяемого содержимым указателя стека, после чего это содержимое указателя стека увеличивается (инкрементируется) на единицу (или на два). В результате получается, что число, записанное последним, будет прочитано первым, а число, записан-

ное первым, будет прочитано последним. Такая память называется LIFO или памятью магазинного типа (например, в магазине автомата патрон, установленный последним, будет извлечен первым).

Принцип действия стека показан на рис. 2.19 (адреса ячеек памяти выбраны условно).

Пусть, например, текущее состояние указателя стека 1000008, и в него надо записать два числа (слова). Первое слово будет записано по адресу 1000006 (перед записью указатель стека уменьшится на два). Второе — по адресу 1000004. После записи содержимое указателя стека — 1000004. Если затем прочитать из стека два слова, то первым будет прочитано слово из адреса 1000004, а после чтения указатель стека станет равным 1000006. Вторым будет прочитано слово из адреса 1000006, а указатель стека станет равным 1000008. Все вернулось к исходному состоянию. Первое записанное слово читается вторым, а второе — первым.

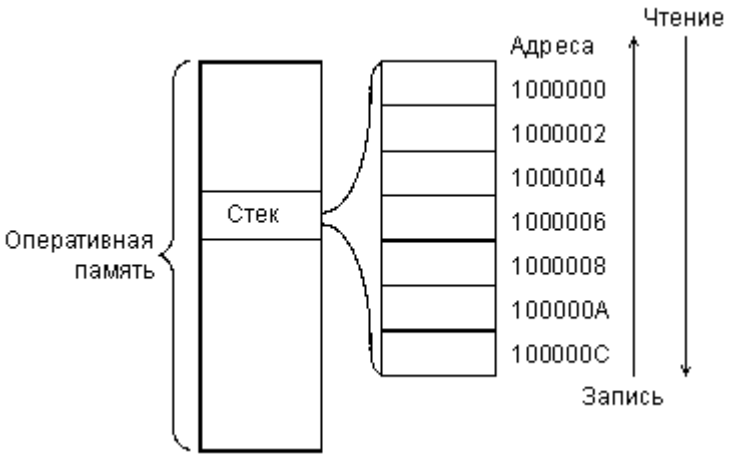

Рис. 2.19. Принцип работы стека.

Необходимость такой адресации становится очевидной в случае многократно вложенных подпрограмм. Пусть, например, выполняется основная программа, и из нее вызывается подпрограмма 1. Если нам надо сохранить значения данных и внутренних регистров основной программы на время выполнения подпрограммы, мы перед вызовом подпрограммы сохраним их в стеке (запишем в стек), а после ее окончания извлечем (прочитаем) их из стека. Если же из подпрограммы 1 вызывается подпрограмма 2, то ту же самую операцию мы проделаем с данными и содержимым внутренних регистров подпрограммы 1. Понятно, что внутри подпрограммы 2 крайними в стеке (читаемыми в первую очередь) будут данные из подпрограммы 1, а данные из основной программы будут глубже. При этом в случае чтения из стека автоматически будет соблюдаться нужный порядок читаемой информации. То же самое будет и в случае, когда таких уровней вложения подпрограмм гораздо больше. То есть то, что надо хранить подольше, прячется поглубже, а то, что скоро может потребоваться — с краю.

В системе команд любого процессора для обмена информацией со стеком предусмотрены специальные команды записи в стек (PUSH) и чтения из стека (POP). В стеке можно прятать не только содержимое всех внутренних регистров процессоров, но и содержимое регистра признаков (слово состояния процессора, PSW). Это позволяет, например, при возвращении из подпрограммы контролировать результат последней команды, выполненной непосредственно перед вызовом этой подпрограммы. Можно также хранить в стеке и данные, для того чтобы удобнее было передавать их между программами и подпрограммами. В общем случае, чем больше область памяти, отведенная под стек, тем больше свободы у программиста и тем более сложные программы могут выполняться.

Следующая специальная область памяти — это **таблица векторов прерываний**.

Вообще, понятие прерывания довольно многозначно. Под прерыванием в общем случае понимается не только обслуживание запроса внешнего устройства, но и любое нарушение последовательной работы процессора. Например, может быть предусмотрено прерывание по факту некорректного вы-

полнения арифметической операции типа деления на ноль. Или же прерывание может быть программным, когда в программе используется команда перехода на какую-то подпрограмму, из которой затем последует возврат в основную программу. В последнем случае общее с истинным прерыванием только то, как осуществляется переход на подпрограмму и возврат из нее.

Любое прерывание обрабатывается через таблицу векторов (указателей) прерываний. В этой таблице в простейшем случае находятся адреса начала программ обработки прерываний, которые и называются векторами. Длина таблицы может быть довольно большой (до нескольких сот элементов). Обычно таблица векторов прерываний располагается в начале пространства памяти (в ячейках памяти с малыми адресами). Адрес каждого вектора (или адрес начального элемента каждого вектора) представляет собой номер прерывания.

В случае аппаратных прерываний номер прерывания или задается устройством, запросившим прерывание (при векторных прерываниях), или же задается номером линии запроса прерываний (при радиальных прерываниях). Процессор, получив аппаратное прерывание, заканчивает выполнение текущей команды и обращается к памяти в область таблицы векторов прерываний, в ту ее строку, которая определяется номером запрошенного прерывания. Затем процессор читает содержимое этой строки (код вектора прерывания) и переходит в адрес памяти, задаваемый этим вектором. Начиная с этого адреса в памяти должна располагаться программа обработки прерывания с данным номером. В конце программы обработки прерываний обязательно должна располагаться команда выхода из прерывания, выполнив которую, процессор возвращается к выполнению прерванной основной программы. Параметры процессора на время выполнения программы обработки прерывания сохраняются в стеке.

Пусть, например, процессор (рис.2.20) выполнял основную программу и команду, находящуюся в адресе памяти 5000 (условно). В этот момент он получил запрос прерывания с номером (адресом вектора) 4. Процессор заканчивает выполнение команды из адреса 5000. Затем он сохраняет в стеке текущее значение счетчика команд (5001) и текущее значение PSW. После этого процессор читает из адреса 4 памяти код вектора прерывания. Пусть этот код равен 6000. Процессор переходит в адрес памяти 6000 и приступает к выполнению программы обработки прерывания, начинающейся с этого адреса. Пусть эта программа заканчивается в адресе 6100. Дойдя до этого адреса, процессор возвращается к выполнению прерванной программы. Для этого он извлекает из стека значение адреса (5001), на котором его прервали, и бывшее в тот момент PSW. Затем процессор читает команду из адреса 5001 и дальше последовательно выполняет команды основной программы.

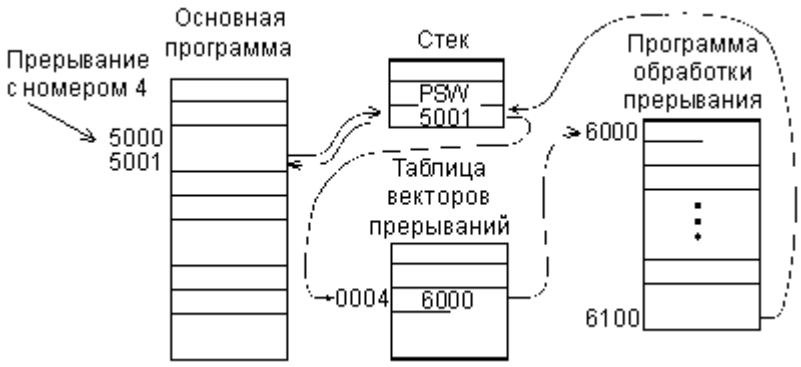

Рис. 2.20. Упрощенный алгоритм обработки прерывания.

Прерывание в случае аварийной ситуации обрабатывается точно так же, только адрес вектора прерывания (номер строки в таблице векторов) жестко привязан к данному типу аварийной ситуации.

Программное прерывание тоже обслуживается через таблицу векторов прерываний, но номер прерывания указывается в составе команды, вызывающей прерывание.

Такая сложная, на первый взгляд, организация прерываний позволяет программисту легко менять программы обработки прерываний, располагать их в любой области памяти, делать их любого размера и любой сложности.

Во время выполнения программы обработки прерывания может поступить новый запрос на прерывание. В этом случае он обрабатывается точно так же, как описано, но основной программой считается прерванная программа обработки предыдущего прерывания. Это называется многократным вложением прерываний. Механизм стека позволяет без проблем обслуживать это многократное вложение, так как первым из стека извлекается тот код, который был сохранен последним, то есть возврат из обработки данного прерывания происходит в программу обработки предыдущего прерывания.

Отметим, что в более сложных случаях в таблице векторов прерываний могут находиться не адреса начала программ обработки прерываний, а так называемые дескрипторы (описатели) прерываний. Но конечным результатом обработки этого дескриптора все равно будет адрес начала программы обработки прерываний.

Наконец, еще одна специальная область памяти микропроцессорной системы — это **память устройств, подключенных к системной шине**. Такое решение встречается нечасто, но иногда оно очень удобно. То есть процессор получает возможность обращаться к внутренней памяти устройств ввода/вывода или каких-то еще подключенных к системной шине устройств, как к своей собственной системной памяти. Обычно окно в пространстве памяти, выделяемое для этого, не слишком большое.

Все остальные части пространства памяти, как правило, имеют универсальное назначение. В них могут располагаться как данные, так и программы (конечно, в случае одношинной архитектуры). Иногда это пространство памяти используется как единое целое, без всяких границ. А иногда пространство памяти делится на сегменты с программно изменяемым адресом начала сегмента и с установленным размером сегмента. Оба подхода имеют свои плюсы и минусы. Например, использование сегментов позволяет защитить область программ или данных, но зато границы сегментов могут затруднять размещение больших программ и массивов данных.

В заключение остановимся на проблеме разделения адресов памяти и адресов устройств ввода/вывода. Существует два основных подхода к решению этой проблемы:

- выделение в общем адресном пространстве системы специальной области адресов для устройств ввода/вывода;
- полное разделение адресных пространств памяти и устройств ввода/вывода.

Первый подход хорош тем, что при обращении к устройствам ввода/вывода процессор может использовать те же команды, которые служат для взаимодействия с памятью. Но адресное пространство памяти должно быть уменьшено на величину адресного пространства устройств ввода/вывода. Например, при 16-разрядной шине адреса всего может быть 64К адресов. Из них 56К адресов отводится под адресное пространство памяти, а 8К адресов — под адресное пространство устройств ввода/вывода.

Преимущество второго подхода состоит в том, что память занимает все адресное пространство микропроцессорной системы. Для общения с устройствами ввода/вывода применяются специальные команды и специальные стробы обмена на магистрали. Именно так сделано, например, в персональных компьютерах. Но возможности взаимодействия с устройствами ввода/вывода в данном случае существенно ограничены по сравнению с возможностями общения с памятью.

2.4.3. Функции устройств ввода/вывода

Устройства ввода/вывода обмениваются информацией с магистралью по тем же принципам, что и память. Наиболее существенное отличие с точки зрения организации обмена состоит в том, что модуль памяти имеет в адресном пространстве системы много адресов (до нескольких десятков мил-

лионов), а устройство ввода/вывода обычно имеет немного адресов (обычно до десяти), а иногда и всего один адрес.

Но модули памяти системы обмениваются информацией только с магистралью, с процессором, а устройства ввода/вывода взаимодействуют еще и с внешними устройствами, цифровыми или аналоговыми. Поэтому разнообразие устройств ввода/вывода неизмеримо больше, чем модулей памяти. Часто используются еще и другие названия для устройств ввода/вывода: устройства сопряжения, контроллеры, карты расширения, интерфейсные модули и т.д.

Объединяют все устройства ввода/вывода общие принципы обмена с магистралью и, соответственно, общие принципы организации узлов, которые осуществляют сопряжение с магистралью. Упрощенная структура устройства ввода/вывода (точнее, его интерфейсной части) приведена на рис. 2.21. Как и в случае модуля памяти, она обязательно содержит схему селектора адреса, схему управления для обработки стробов обмена и буферы данных.

Самые простейшие устройства ввода/вывода выдают на внешнее устройство код данных в параллельном формате и принимают из внешнего устройства код данных в параллельном формате. Такие устройства ввода/вывода часто называют параллельными портами ввода/вывода. Они наиболее универсальны, то есть удовлетворяют потребности сопряжения с большим числом внешних устройств, поэтому их часто вводят в состав микропроцессорной системы в качестве стандартных устройств. Параллельные порты обычно имеются в составе микроконтроллеров. Именно через параллельные порты микроконтроллер связывается с внешним миром.

Входной порт (порт ввода) в простейшем случае представляет собой параллельный регистр, в который процессор может записывать информацию. Выходной порт (порт вывода) обычно представляет собой просто однонаправленный буфер, через который процессор может читать информацию от внешнего устройства. Именно такие порты показаны для примера на рис. 2.21. Порт может быть и двунаправленным (входным/выходным). В этом случае процессор пишет информацию во внешнее устройство и читает информацию из внешнего устройства по одному и тому же адресу в адресном пространстве системы. Входные и выходные линии для связи с внешним устройством при этом могут быть объединены поразрядно, образуя двунаправленные линии.

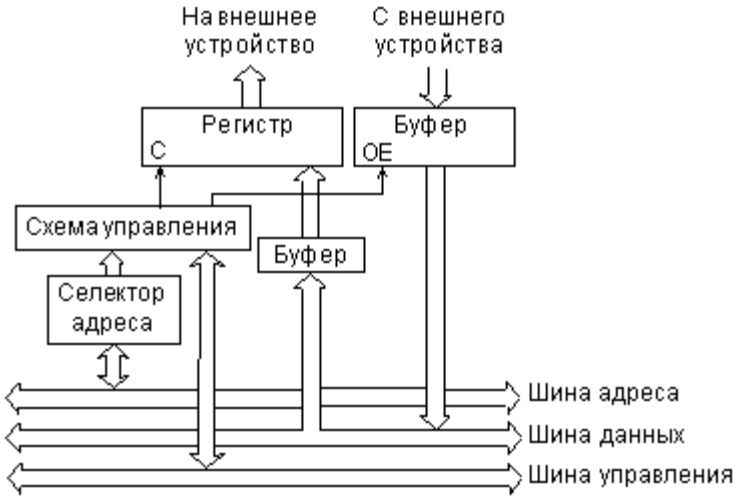

Рис. 2.21. Структура простейшего устройства ввода/вывода.

При обращении со стороны магистрали селектор адреса распознает адрес, приписанный данному устройству ввода/вывода. Схема управления выдает внутренние стробы обмена в ответ на магистральные стробы обмена. Входной буфер данных обеспечивает электрическое согласование шины данных с этим устройством (буфер может и отсутствовать). Данные из шины данных записываются в регистр по сигналу C и выдаются на внешнее устройство. Выходной буфер данных передает входные данные с внешнего устройства на шину данных магистрали в цикле чтения из порта.

Более сложные устройства ввода/вывода (устройства сопряжения) имеют в своем составе внутреннюю буферную оперативную память и даже могут иметь микроконтроллер, на который возложено выполнение функций обмена с внешним устройством.

Каждому устройству ввода/вывода отводится свой адрес в адресном пространстве микропроцессорной системы. Дублирование адресов должно быть исключено, за этим должны следить разработчик и пользователь микропроцессорной системы.

Устройства ввода/вывода помимо программного обмена могут также поддерживать режим обмена по прерываниям. В этом случае они преобразуют поступающий от внешнего устройства сигнал запроса на прерывание в сигнал запроса прерывания, необходимый для данной магистрали (или в последовательность сигналов при векторном прерывании). Если нужно использовать режим ПДП, устройство ввода/вывода должно выдать сигнал запроса ПДП на магистраль и обеспечить работу в циклах ПДП, принятых для данной магистрали.

В составе микропроцессорных систем, как правило, выделяются три специальные группы устройств ввода/вывода:

- устройства интерфейса пользователя (ввода информации пользователем и вывода информации для пользователя);
- устройства ввода/вывода для длительного хранения информации;
- таймерные устройства.

К устройствам ввода для интерфейса пользователя относятся контроллеры клавиатуры, тумблеров, отдельных кнопок, мыши, трекбола, джойстика и т.д. К устройствам вывода для интерфейса пользователя относятся контроллеры светодиодных индикаторов, табло, жидкокристаллических, плазменных и электронно-лучевых экранов и т.д. В простейших случаях управляющих контроллеров или микроконтроллеров эти средства могут отсутствовать. В сложных микропроцессорных системах они есть обязательно. Роль внешнего устройства в данном случае играет человек.

Устройства ввода/вывода для длительного хранения информации обеспечивают сопряжение микропроцессорной системы с дисководами (компакт-дисков или магнитных дисков), а также с накопителями на магнитной ленте. Применение таких устройств существенно увеличивает возможности микропроцессорной системы в отношении хранения выполняемых программ и накопления массивов данных. В простейших контроллерах эти устройства отсутствуют.

Таймерные устройства отличаются от других устройств ввода/вывода тем, что они могут не иметь внешних выводов для подключения к внешним устройствам. Эти устройства предназначены для того, чтобы микропроцессорная система могла выдерживать заданные временные интервалы, следить за реальным временем, считать импульсы и т.д. В основе любого таймера лежит кварцевый тактовый генератор и многоразрядные двоичные счетчики, которые могут перезапускать друг друга. Процессор может записывать в таймер коэффициенты деления тактовой частоты, количество отсчитываемых импульсов, задавать режим работы счетчиков таймера, а читает процессор выходные коды счетчиков. В принципе выполнить практически все функции таймера можно и программным путем, поэтому иногда таймеры в системе отсутствуют. Но включение в систему таймера позволяет решать более сложные задачи и строить более эффективные алгоритмы.

Еще один важный класс устройств ввода/вывода — это устройства для подключения к информационным сетям (локальным и глобальным). Эти устройства распространены не так широко, как устройства трех перечисленных ранее групп, но их значение с каждым годом становится все больше. Сейчас средства связи с информационными сетями вводятся иногда даже в простые контроллеры.

Иногда устройства ввода/вывода обеспечивают сопряжение с внешними устройствами с помощью аналоговых сигналов. Это бывает очень удобно, поэтому в состав некоторых микроконтроллеров даже вводят внутренние ЦАП и АЦП.

4. Лекция: Адресация операндов

В этой лекции рассказывается о функциях основных узлов процессора, о методах адресации операндов и о регистрах процессора.

Основная функция любого процессора, ради которой он и создается, — это выполнение команд. Система команд, выполняемых процессором, представляет собой нечто подобное таблице истинности логических элементов или таблице режимов работы более сложных логических микросхем. То есть она определяет логику работы процессора и его реакцию на те или иные комбинации внешних событий.

Написание программ для микропроцессорной системы — важнейший и часто наиболее трудоемкий этап разработки такой системы. А для создания эффективных программ необходимо иметь хотя бы самое общее представление о системе команд используемого процессора. Самые компактные и быстрые программы и подпрограммы создаются на языке Ассемблер, использование которого без знания системы команд абсолютно невозможно, ведь язык Ассемблер представляет собой символьную запись цифровых кодов машинного языка, кодов команд процессора. Конечно, для разработки программного обеспечения существуют всевозможные программные средства. Пользоваться ими обычно можно и без знания системы команд процессора. Чаще всего применяются языки программирования высокого уровня, такие как Паскаль и Си. Однако знание системы команд и языка Ассемблер позволяет в несколько раз повысить эффективность некоторых наиболее важных частей программного обеспечения любой микропроцессорной системы — от микроконтроллера до персонального компьютера.

Именно поэтому в данной главе мы рассмотрим основные типы команд, имеющиеся у большинства процессоров, и особенности их применения.

Каждая команда, выбираемая (читаемая) из памяти процессором, определяет алгоритм поведения процессора на ближайшие несколько тактов. Код команды говорит о том, какую операцию предстоит выполнить процессору и с какими операндами (то есть кодами данных), где взять исходную информацию для выполнения команды и куда поместить результат (если необходимо). Код команды может занимать от одного до нескольких байт, причем процессор узнает о том, сколько байт команды ему надо читать, из первого прочитанного им байта или слова. В процессоре код команды расшифровывается и преобразуется в набор микроопераций, выполняемых отдельными узлами процессора. Но разработчику микропроцессорных систем это знание не слишком важно, ему важен только результат выполнения той или иной команды.

3.1. Адресация операндов

Большая часть команд процессора работает с кодами данных (операндами). Одни команды требуют входных операндов (одного или двух), другие выдают выходные операнды (чаще один операнд). Входные операнды называются еще операндами-источниками, а выходные называются операндами-приемниками. Все эти коды операндов (входные и выходные) должны где-то располагаться. Они могут находиться во внутренних регистрах процессора (наиболее удобный и быстрый вариант). Они могут располагаться в системной памяти (самый распространенный вариант). Наконец, они могут находиться в устройствах ввода/вывода (наиболее редкий случай). Определение места положения операндов производится кодом команды. Причем существуют разные методы, с помощью которых код команды может определить, откуда брать входной операнд и куда помещать выходной операнд. Эти методы называются методами адресации. Эффективность выбранных методов адресации во многом определяет эффективность работы всего процессора в целом.

3.1.1. Методы адресации

Количество методов адресации в различных процессорах может быть от 4 до 16. Рассмотрим несколько типичных методов адресации операндов, используемых сейчас в большинстве микропроцессоров.

Непосредственная адресация (рис. 3.1) предполагает, что операнд (входной) находится в памяти непосредственно за кодом команды. Операнд обычно представляет собой константу, которую надо куда-то переслать, к чему-то прибавить и т.д. Например, команда может состоять в том, чтобы прибавить число 6 к содержимому какого-то внутреннего регистра процессора. Это число 6 будет располагаться в памяти, внутри программы в адресе, следующем за кодом данной команды сложения.

Рис. 3.1. Непосредственная адресация.

Прямая (она же абсолютная) адресация (рис. 3.2) предполагает, что операнд (входной или выходной) находится в памяти по адресу, код которого находится внутри программы сразу же за кодом команды. Например, команда может состоять в том, чтобы очистить (сделать нулевым) содержимое ячейки памяти с адресом 1000000. Код этого адреса 1000000 будет располагаться в памяти, внутри программы в следующем адресе за кодом данной команды очистки.

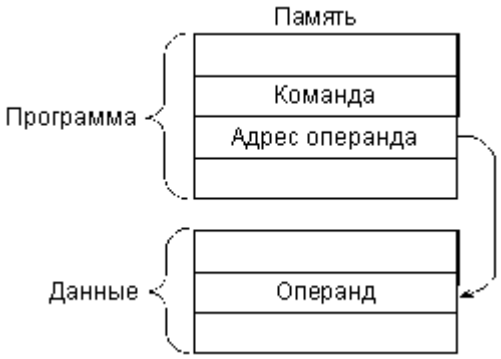

Рис. 3.2. Прямая адресация.

Регистровая адресация (рис. 3.3) предполагает, что операнд (входной или выходной) находится во внутреннем регистре процессора. Например, команда может состоять в том, чтобы переслать число из нулевого регистра в первый. Номера обоих регистров (0 и 1) будут определяться кодом команды пересылки.

Косвенно-регистровая (она же косвенная) адресация предполагает, что во внутреннем регистре процессора находится не сам операнд, а его адрес в памяти (рис. 3.4). Например, команда может состоять в том, чтобы очистить ячейку памяти с адресом, находящимся в нулевом регистре. Номер этого регистра (0) будет определяться кодом команды очистки.

Рис. 3.3. Регистровая адресация.

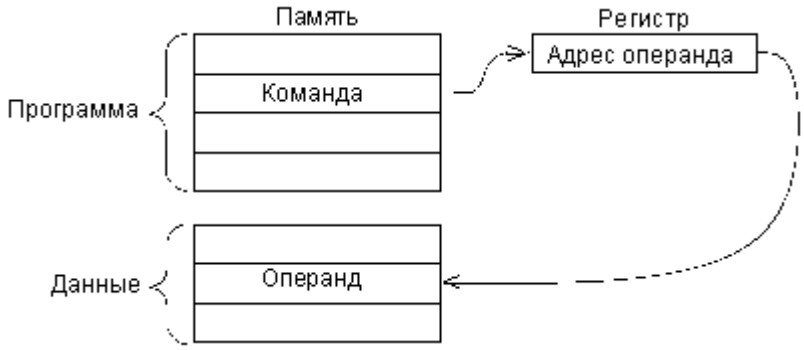

Рис. 3.4. Косвенная адресация.

Реже встречаются еще два метода адресации.

Автоинкрементная адресация очень близка к косвенной адресации, но отличается от нее тем, что после выполнения команды содержимое используемого регистра увеличивается на единицу или на два. Этот метод адресации очень удобен, например, при последовательной обработке кодов из массива данных, находящегося в памяти. После обработки какого-то кода адрес в регистре будет указывать уже на следующий код из массива. При использовании косвенной адресации в данном случае пришлось бы увеличивать содержимое этого регистра отдельной командой.

Автодекрементная адресация работает похоже на автоинкрементную, но только содержимое выбранного регистра уменьшается на единицу или на два перед выполнением команды. Эта адресация также удобна при обработке массивов данных. Совместное использование автоинкрементной и автодекрементной адресаций позволяет организовать память стекового типа (см. раздел 2.4.2).

Из других распространенных методов адресации можно упомянуть об индексных методах, которые предполагают для вычисления адреса операнда прибавление к содержимому регистра заданной константы (индекса). Код этой константы располагается в памяти непосредственно за кодом команды.

Отметим, что выбор того или иного метода адресации в значительной степени определяет время выполнения команды. Самая быстрая адресация — это регистровая, так как она не требует дополнительных циклов обмена по магистрали. Если же адресация требует обращения к памяти, то время выполнения команды будет увеличиваться за счет длительности необходимых циклов обращения к памяти. Понятно, что чем больше внутренних регистров у процессора, тем чаще и свободнее можно применять регистровую адресацию, и тем быстрее будет работать система в целом.

3.1.2. Сегментирование памяти

Говоря об адресации, нельзя обойти вопрос о сегментировании памяти, применяемой в некоторых процессорах, например в процессорах IBM PC-совместимых персональных компьютеров.

В процессоре Intel 8086 сегментирование памяти организовано следующим образом.

Вся память системы представляется не в виде непрерывного пространства, а в виде нескольких кусков — сегментов заданного размера (по 64 Кбайта), положение которых в пространстве памяти можно изменять программным путем.

Для хранения кодов адресов памяти используются не отдельные регистры, а пары регистров:

- сегментный регистр определяет адрес начала сегмента (то есть положение сегмента в памяти);
- регистр указателя (регистр смещения) определяет положение рабочего адреса внутри сегмента.

При этом физический 20-разрядный адрес памяти, выставляемый на внешнюю шину адреса, образуется так, как показано на рис. 3.5, то есть путем сложения смещения и адреса сегмента со сдвигом на 4 бита. Положение этого адреса в памяти показано на рис. 3.6.

Сегмент может начинаться только на 16-байтной границе памяти (так как адрес начала сегмента, по сути, имеет четыре младших нулевых разряда, как видно из рис. 3.5), то есть с адреса, кратного 16. Эти допустимые границы сегментов называются границами параграфов.

Отметим, что введение сегментирования, прежде всего, связано с тем, что внутренние регистры процессора 16-разрядные, а физический адрес памяти 20-разрядный (16-разрядный адрес позволяет использовать память только в 64 Кбайт, что явно недостаточно). В появившемся в то же время процессоре MC68000 фирмы Motorola внутренние регистры 32-разрядные, поэтому там проблемы сегментирования памяти не возникает.

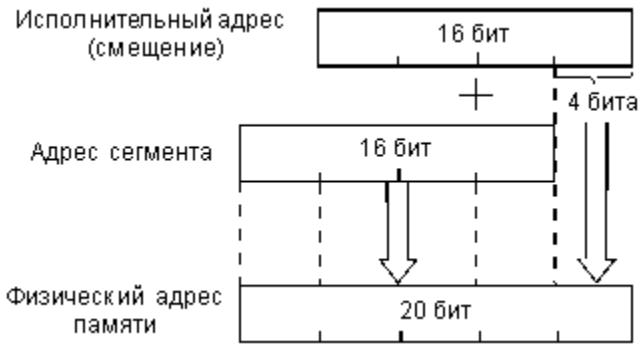

Рис. 3.5. Формирование физического адреса памяти из адреса сегмента и смещения.

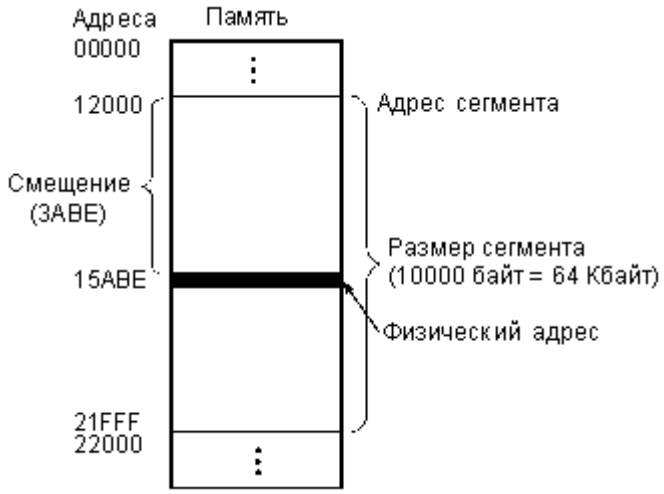

Рис. 3.6. Физический адрес в сегменте (все коды — шестнадцатеричные).

Применяются и более сложные методы сегментирования памяти. Например, в процессоре Intel 80286 в так называемом защищенном режиме адрес памяти вычисляется в соответствии с рис. 3.7.

В сегментном регистре в данном случае хранится не базовый (начальный) адрес сегментов, а коды селекторов, определяющие адреса в памяти, по которым хранятся дескрипторы (то есть описатели) сегментов. Область памяти с дескрипторами называется таблицей дескрипторов. Каждый дескриптор сегмента содержит базовый адрес сегмента, размер сегмента (от 1 до 64 Кбайт) и его атрибуты. Базовый адрес сегмента имеет разрядность 24 бит, что обеспечивает адресацию 16 Мбайт физической памяти.

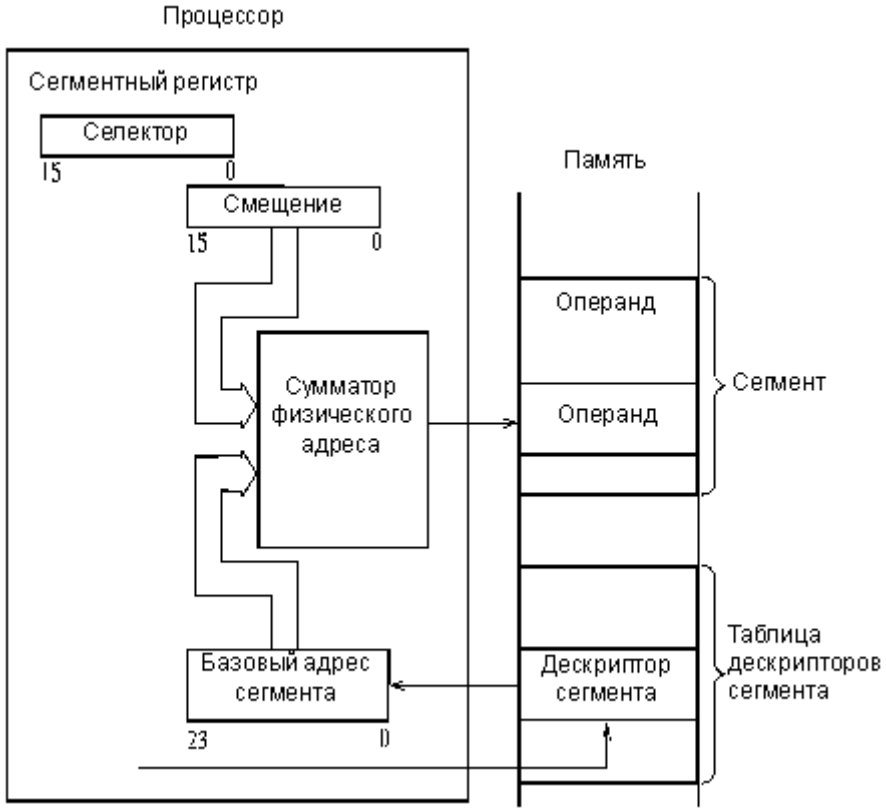

Рис. 3.7. Адресация памяти в защищенном режиме процессора Intel 80286.

Таким образом, на сумматор, вычисляющий физический адрес памяти, подается не содержимое сегментного регистра, как в предыдущем случае, а базовый адрес сегмента из таблицы дескрипторов.

Еще более сложный метод адресации памяти с сегментированием использован в процессоре Intel 80386 и в более поздних моделях процессоров фирмы Intel. Этот метод иллюстрируется рис. 3.8.

Адрес памяти (физический адрес) вычисляется в три этапа. Сначала вычисляется так называемый **эффективный адрес** (32-разрядный) путем суммирования трех компонентов: базы, индекса и смещения (Base, Index, Displacement), причем возможно умножение индекса на масштаб (Scale). Эти компоненты имеют следующий смысл:

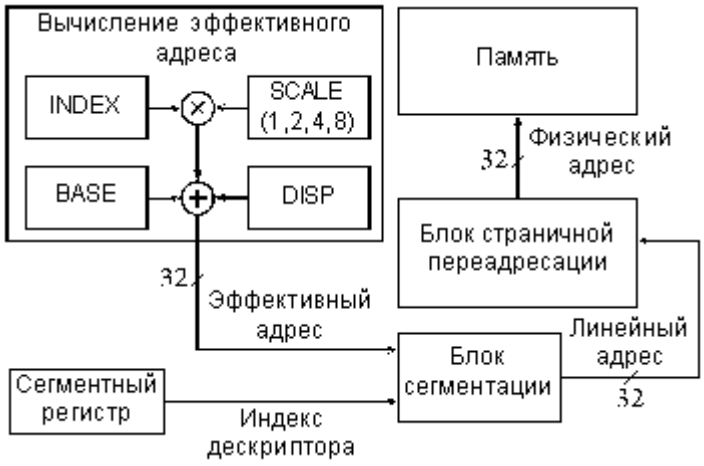

Рис. 3.8. Формирование физического адреса памяти процессора 80386 в защищенном режиме.

- смещение — это 8-, 16- или 32-разрядное число, включенное в команду.
- база — это содержимое базового регистра процессора. Обычно оно используется для указания на начало некоторого массива.
- индекс — это содержимое индексного регистра процессора. Обычно оно используется для выбора одного из элементов массива.
- масштаб — это множитель (он может быть равен 1, 2, 4 или 8), указанный в коде команды, на который перед суммированием с другими компонентами умножается индекс. Он используется для указания размера элемента массива.

Затем специальный блок сегментации вычисляет 32-разрядный линейный адрес, который представляет собой сумму базового адреса сегмента из сегментного регистра с эффективным адресом. Наконец, физический 32-битный адрес памяти образуется путем преобразования линейного адреса блоком страничной переадресации, который осуществляет перевод линейного адреса в физический страницами по 4 Кбайта.

В любом случае сегментирование позволяет выделить в памяти один или несколько сегментов для данных и один или несколько сегментов для программ. Переход от одного сегмента к другому сводится всего лишь к изменению содержимого сегментного регистра. Иногда это бывает очень удобно. Но для программиста работать с сегментированной памятью обычно сложнее, чем с непрерывной, несегментированной памятью, так как приходится следить за границами сегментов, за их описанием, переключением и т.д.

3.1.3. Адресация байтов и слов

Многие процессоры, имеющие разрядность 16 или 32, способны адресовать не только целое слово в памяти (16-разрядное или 32-разрядное), но и отдельные байты. Каждому байту в каждом слове при этом отводится свой адрес.

Так, в случае 16-разрядных процессоров все слова в памяти (16-разрядные) имеют четные адреса. А байты, входящие в эти слова, могут иметь как четные адреса, так и нечетные.

Например, пусть 16-разрядная ячейка памяти имеет адрес 23420, и в ней хранится код 2A5E (рис. 3.9).

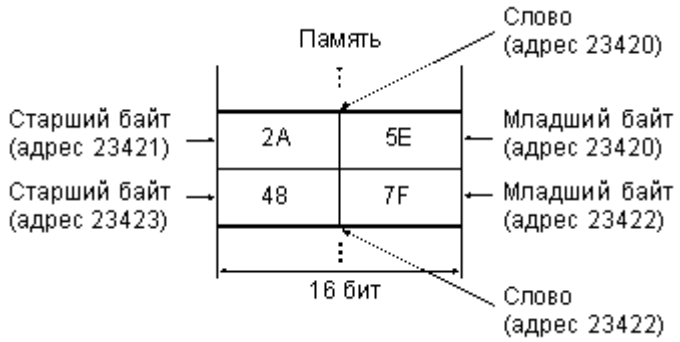

Рис. 3.9. Адресация слов и байтов.

При обращении к целому слову (с содержимым 2A5E) процессор выставляет адрес 23420. При обращении к младшему байту этой ячейки (с содержимым 5E) процессор выставляет тот же самый адрес 23420, но использует команду, адресующую байт, а не слово. При обращении к старшему байту этой же ячейки (с содержимым 2A) процессор выставляет адрес 23421 и использует команду, адресующую байт. Следующая по порядку 16-разрядная ячейка памяти с содержимым 487F будет иметь адрес 23422, то есть опять же четный. Ее байты будут иметь адреса 23422 и 23423.

Для различия байтовых и словных циклов обмена на магистрали в шине управления предусматривается специальный сигнал байтового обмена. Для работы с байтами в систему команд процессора вводятся специальные команды или предусматриваются методы байтовой адресации.

3.2. Регистры процессора

Как уже упоминалось, внутренние регистры процессора представляют собой сверхоперативную память небольшого размера, которая предназначена для временного хранения служебной информации или данных. Количество регистров в разных процессорах может быть от 6—8 до нескольких десятков. Регистры могут быть универсальными и специализированными. Специализированные регистры, которые присутствуют в большинстве процессоров, — это регистр-счетчик команд, регистр состояния (PSW), регистр указателя стека. Остальные регистры процессора могут быть как универсальными, так и специализированными.

Например, в 16-разрядном процессоре T-11 фирмы DEC было 8 регистров общего назначения (РОН) и один регистр состояния. Все регистры имели по 16 разрядов. Из регистров общего назначения один отводился под счетчик команд, другой — под указатель стека. Все остальные регистры общего назначения полностью взаимозаменяемы, то есть имеют универсальное назначение, могут хранить как данные, так и адреса (указатели), индексы и т.д. Максимально допустимый объем памяти для данного процессора составлял 64 Кбайт (адрес памяти 16-разрядный).

В 16-разрядном процессоре MC68000 фирмы Motorola было 19 регистров: 16-разрядный регистр состояния, 32-разрядный регистр счетчика команд, 9 регистров адреса (32-разрядных) и 8 регистров данных (32-разрядных). Два регистра адреса отведены под указатели стека. Максимально допустимый объем адресуемой памяти — 16 Мбайт (внешняя шина адреса 24-разрядная). Все 8 регистров данных взаимозаменяемы. 7 регистров адреса – тоже взаимозаменяемы.

В 16-разрядном процессоре Intel 8086, который стал базовым в линии процессоров, используемых в персональных компьютерах, реализован принципиально другой подход. Каждый регистр этого процессора имеет свое особое назначение, и заменять друг друга регистры могут только частично или же не могут вообще. Остановимся на особенностях этого процессора подробнее.

Процессор 8086 имеет 14 регистров разрядностью по 16 бит. Из них четыре регистра (AX, BX, CX, DX) — это регистры данных, каждый из которых помимо хранения операндов и результатов операций имеет еще и свое специфическое назначение:

- регистр AX — умножение, деление, обмен с устройствами ввода/вывода (команды ввода и вывода);
- регистр BX — базовый регистр в вычислениях адреса;
- регистр CX — счетчик циклов;
- регистр DX — определение адреса ввода/вывода.

Для регистров данных существует возможность раздельного использования обоих байтов (например, для регистра AX они имеют обозначения AL – младший байт и AH — старший байт).

Следующие четыре внутренних регистра процессора — это сегментные регистры, каждый из которых определяет положение одного из рабочих сегментов (рис. 3.10):

- регистр CS (Code Segment) соответствует сегменту команд, исполняемых в данный момент;
- регистр DS (Data Segment) соответствует сегменту данных, с которыми работает процессор;
- регистр ES (Extra Segment) соответствует дополнительному сегменту данных;
- регистр SS (Stack Segment) соответствует сегменту стека.

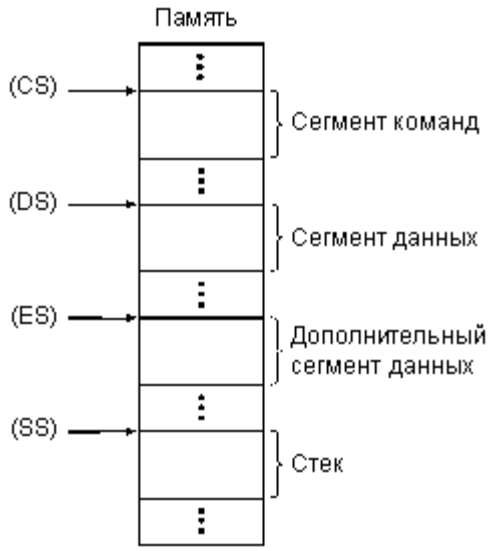

Рис. 3.10. Сегменты команд, данных и стека в памяти.

В принципе, все эти сегменты могут и перекрываться для оптимального использования пространства памяти. Например, если программа занимает только часть сегмента, то сегмент данных может начинаться сразу после завершения работы программы (с точностью 16 байт), а не после окончания всего сегмента программы.

Следующие пять регистров процессора (SP — Stack Pointer, BP — Base Pointer, SI — Source Index, DI — Destination Index, IP —Instruction Pointer) служат указателями (то есть определяют смещение в пределах сегмента). Например, счетчик команд процессора образуется парой регистров CS и IP, а указатель стека — парой регистров SP и SS. Регистры SI, DI используются в строковых операциях, то есть при последовательной обработке нескольких ячеек памяти одной командой.

Последний регистр FLAGS — это регистр состояния процессора (PSW). Из его 16 разрядов используются только девять (рис. 3.11): CF (Carry Flag) — флаг переноса при арифметических операциях, PF (Parity Flag) — флаг четности результата, AF (Auxiliary Flag) — флаг дополнительного переноса, ZF (Zero Flag) — флаг нулевого результата, SF (Sign Flag) — флаг знака (совпадает со старшим битом результата), TF (Trap Flag) — флаг пошагового режима (используется при отладке), IF (Interrupt-enable Flag) — флаг разрешения аппаратных прерываний, DF (Direction Flag) — флаг направления при строковых операциях, OF (Overflow Flag) — флаг переполнения.

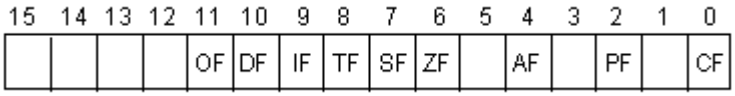

Рис. 3.11. Регистр состояния процессора 8086.

Биты регистра состояния устанавливаются или очищаются в зависимости от результата исполнения предыдущей команды и используются некоторыми командами процессора. Биты регистра состояния могут также устанавливаться и очищаться специальными командами процессора (о системе команд процессора будет рассказано в следующем разделе).

Во многих процессорах выделяется специальный регистр, называемый **аккумулятором** (то есть накопителем). При этом, как правило, только этот регистр-аккумулятор может участвовать во всех операциях, только через него может производиться взаимодействие с устройствами ввода/вывода. Иногда в него же помещается результат любой выполненной команды (в этом случае говорят даже об "аккумуляторной" архитектуре процессора). Например, в процессоре 8086 регистр данных AX можно считать своеобразным аккумулятором, так как именно он обязательно участвует в командах умножения и деления, а также только через него можно пересылать данные в устройство ввода/вывода и из устройства ввода/вывода. Выделение специального регистра-аккумулятора упрощает структуру процессора и ускоряет пересылки кодов внутри процессора, но в некоторых случаях замедляет работу системы в целом, так как весь поток информации должен пройти через один регистр-аккумулятор. В случае, когда несколько регистров процессора полностью взаимозаменяемы, таких проблем не возникает.

5. Лекция: Система команд процессора

В этой лекции рассказывается об основных группах команд процессора, об особенностях выполнения различных команд, о методах организации подпрограмм.

3.3. Система команд процессора

В общем случае система команд процессора включает в себя следующие четыре основные группы команд:

- команды пересылки данных;
- арифметические команды;
- логические команды;
- команды переходов.

Команды пересылки данных не требуют выполнения никаких операций над операндами. Операнды просто пересылаются (точнее, копируются) из источника (Source) в приемник (Destination). Источником и приемником могут быть внутренние регистры процессора, ячейки памяти или устройства ввода/вывода. АЛУ в данном случае не используется.

Арифметические команды выполняют операции сложения, вычитания, умножения, деления, увеличения на единицу (инкрементирования), уменьшения на единицу (декрементирования) и т.д. Этим командам требуется один или два входных операнда. Формируют команды один выходной операнд.

Логические команды производят над операндами логические операции, например, логическое И, логическое ИЛИ, исключающее ИЛИ, очистку, инверсию, разнообразные сдвиги (вправо, влево, арифметический сдвиг, циклический сдвиг). Этим командам, как и арифметическим, требуется один или два входных операнда, и формируют они один выходной операнд.

Наконец, команды переходов предназначены для изменения обычного порядка последовательного выполнения команд. С их помощью организуются переходы на подпрограммы и возвраты из них, всевозможные циклы, ветвления программ, пропуски фрагментов программ и т.д. Команды переходов всегда меняют содержимое счетчика команд. Переходы могут быть условными и безусловными. Именно эти команды позволяют строить сложные алгоритмы обработки информации.

В соответствии с результатом каждой выполненной команды устанавливаются или очищаются биты регистра состояния процессора (PSW). Но надо помнить, что не все команды изменяют все имеющиеся в PSW флаги. Это определяется особенностями каждого конкретного процессора.

У разных процессоров системы команд существенно различаются, но в основе своей они очень похожи. Количество команд у процессоров также различно. Например, у упоминавшегося уже процессора MC68000 всего 61 команда, а у процессора 8086 — 133 команды. У современных мощных процессоров количество команд достигает нескольких сотен. В то же время существуют процессоры с сокращенным набором команд (так называемые RISC-процессоры), в которых за счет максимального сокращения количества команд достигается увеличение эффективности и скорости их выполнения.

Рассмотрим теперь особенности четырех выделенных групп команд процессора более подробно.

3.3.1. Команды пересылки данных

Команды пересылки данных занимают очень важное место в системе команд любого процессора. Они выполняют следующие важнейшие функции:

- загрузка (запись) содержимого во внутренние регистры процессора;
- сохранение в памяти содержимого внутренних регистров процессора;
- копирование содержимого из одной области памяти в другую;
- запись в устройства ввода/вывода и чтение из устройств ввода/вывода.

В некоторых процессорах (например, T-11) все эти функции выполняются одной единственной командой MOV (для байтовых пересылок — MOVB) но с различными методами адресации операндов.

В других процессорах помимо команды MOV имеется еще несколько команд для выполнения перечисленных функций. Например, для загрузки регистров могут использоваться команды загрузки, причем для разных регистров — разные команды (их обозначения обычно строятся с использованием слова LOAD — загрузка). Часто выделяются специальные команды для сохранения в стеке и для извлечения из стека (PUSH — сохранить в стеке, POP — извлечь из стека). Эти команды выполняют пересылку с автоинкрементной и с автодекрементной адресацией (даже если эти режимы адресации не предусмотрены в процессоре в явном виде).

Иногда в систему команд вводится специальная команда MOVS для строчной (или цепочечной) пересылки данных (например, в процессоре 8086). Эта команда пересылает не одно слово или байт, а заданное количество слов или байтов (MOVSB), то есть инициирует не один цикл обмена по магистрали, а несколько. При этом адрес памяти, с которым происходит взаимодействие, увеличивается на 1 или на 2 после каждого обращения или же уменьшается на 1 или на 2 после каждого обращения. То есть в неявном виде применяется автоинкрементная или автодекрементная адресация.

В некоторых процессорах (например, в процессоре 8086) специально выделяются функции обмена с устройствами ввода/вывода. Команда IN используется для ввода (чтения) информации из устройства ввода/вывода, а команда OUT используется для вывода (записи) в устройство ввода/вывода. Обмен информацией в этом случае производится между регистром-аккумулятором и устройством ввода/вывода. В более продвинутых процессорах этого же семейства (начиная с процессора 80286) добавлены команды строчного (цепочечного) ввода (команда INS) и строчного вывода (команда OUTS). Эти команды позволяют пересылать целый массив (строку) данных из памяти в устройство ввода/вывода (OUTS) или из устройства ввода/вывода в память (INS). Адрес памяти после каждого обращения увеличивается или уменьшается (как и в случае с командой MOVS).

Также к командам пересылки данных относятся команды обмена информацией (их обозначение строится на основе слова Exchange). Может быть предусмотрен обмен информацией между внутренними регистрами, между двумя половинами одного регистра (SWAP) или между регистром и ячейкой памяти.

3.3.2. Арифметические команды

Арифметические команды рассматривают коды операндов как числовые двоичные или двоично-десятичные коды. Эти команды могут быть разделены на пять основных групп:

- команды операций с фиксированной запятой (сложение, вычитание, умножение, деление);
- команды операций с плавающей запятой (сложение, вычитание, умножение, деление);
- команды очистки;
- команды инкремента и декремента;
- команда сравнения.

Команды операций с фиксированной запятой работают с кодами в регистрах процессора или в памяти как с обычными двоичными кодами. Команда сложения (ADD) вычисляет сумму двух кодов. Команда вычитания (SUB) вычисляет разность двух кодов. Команда умножения (MUL) вычисляет произведение двух кодов (разрядность результата вдвое больше разрядности сомножителей). Команда деления (DIV) вычисляет частное от деления одного кода на другой. Причем все эти команды могут работать как с числами со знаком, так и с числами без знака.

Команды операций с плавающей запятой (точкой) используют формат представления чисел с порядком и мантиссой (обычно эти числа занимают две последовательные ячейки памяти). В современных мощных процессорах набор команд с плавающей запятой не ограничивается только четырьмя арифметическими действиями, а содержит и множество других более сложных команд, например, вычисление тригонометрических функций, логарифмических функций, а также сложных функций, необходимых при обработке звука и изображения.

Команды очистки (CLR) предназначены для записи нулевого кода в регистр или ячейку памяти. Эти команды могут быть заменены командами пересылки нулевого кода, но специальные команды очистки обычно выполняются быстрее, чем команды пересылки. Команды очистки иногда относят к группе логических команд, но суть их от этого не меняется.

Команды инкремента (увеличения на единицу, INC) и декремента (уменьшения на единицу, DEC) также бывают очень удобны. Их можно в принципе заменить командами суммирования с единицей или вычитания единицы, но инкремент и декремент выполняются быстрее, чем суммирование и вычитание. Эти команды требуют одного входного операнда, который одновременно является и выходным операндом.

Наконец, команда сравнения (обозначается CMP) предназначена для сравнения двух входных операндов. По сути, она вычисляет разность этих двух операндов, но выходного операнда не формирует, а всего лишь изменяет биты в регистре состояния процессора (PSW) по результату этого вычитания. Следующая за командой сравнения команда (обычно это команда перехода) будет анализировать биты в регистре состояния процессора и выполнять действия в зависимости от их значений (о командах перехода речь идет в разделе 3.3.4). В некоторых процессорах предусмотрены команды цепочечного сравнения двух последовательностей операндов, находящихся в памяти (например, в процессоре 8086 и совместимых с ним).

3.3.3. Логические команды

Логические команды выполняют над операндами логические (побитовые) операции, то есть они рассматривают коды операндов не как единое число, а как набор отдельных битов. Этим они отличаются от арифметических команд. Логические команды выполняют следующие основные операции:

- логическое И, логическое ИЛИ, сложение по модулю 2 (Исключающее ИЛИ);
- логические, арифметические и циклические сдвиги;
- проверка битов и операндов;
- установка и очистка битов (флагов) регистра состояния процессора (PSW).

Команды логических операций позволяют побитно вычислять основные логические функции от двух входных операндов. Кроме того, операция И (AND) используется для принудительной очистки заданных битов (в качестве одного из операндов при этом используется код маски, в котором разряды, требующие очистки, установлены в нуль). Операция ИЛИ (OR) применяется для принудительной установки заданных битов (в качестве одного из операндов при этом используется код маски, в котором разряды, требующие установки в единицу, равны единице). Операция "Исключающее ИЛИ" (XOR) используется для инверсии заданных битов (в качестве одного из операндов при этом применяется код маски, в котором биты, подлежащие инверсии, установлены в единицу). Команды требуют двух входных операндов и формируют один выходной операнд.

Команды сдвигов позволяют побитно сдвигать код операнда вправо (в сторону младших разрядов) или влево (в сторону старших разрядов). Тип сдвига (логический, арифметический или циклический) определяет, каково будет новое значение старшего бита (при сдвиге вправо) или младшего бита (при сдвиге влево), а также определяет, будет ли где-то сохранено прежнее значение старшего бита (при сдвиге влево) или младшего бита (при сдвиге вправо). Например, при логическом сдвиге вправо в старшем разряде кода операнда устанавливается нуль, а младший разряд записывается в качестве флага переноса в регистр состояния процессора. А при арифметическом сдвиге вправо значение старшего разряда сохраняется прежним (нулем или единицей), младший разряд также записывается в качестве флага переноса.

Циклические сдвиги позволяют сдвигать биты кода операнда по кругу (по часовой стрелке при сдвиге вправо или против часовой стрелки при сдвиге влево). При этом в кольцо сдвига может входить или не входить флаг переноса. В бит флага переноса (если он используется) записывается значение старшего бита при циклическом сдвиге влево и младшего бита при циклическом сдвиге вправо. Соответственно, значение бита флага переноса будет переписываться в младший разряд при циклическом сдвиге влево и в старший разряд при циклическом сдвиге вправо.

Для примера на рис. 3.12 показаны действия, выполняемые командами сдвигов вправо.

Команды проверки битов и операндов предназначены для установки или очистки битов регистра состояния процессора в зависимости от значения выбранных битов или всего операнда в целом. Выходного операнда команды не формируют. Команда проверки операнда (TST) проверяет весь код операнда в целом на равенство нулю и на знак (на значение старшего бита), она требует только одного входного операнда. Команда проверки бита (BIT) проверяет только отдельные биты, для выбора которых в качестве второго операнда используется код маски. В коде маски проверяемым битам основного операнда должны соответствовать единичные разряды.

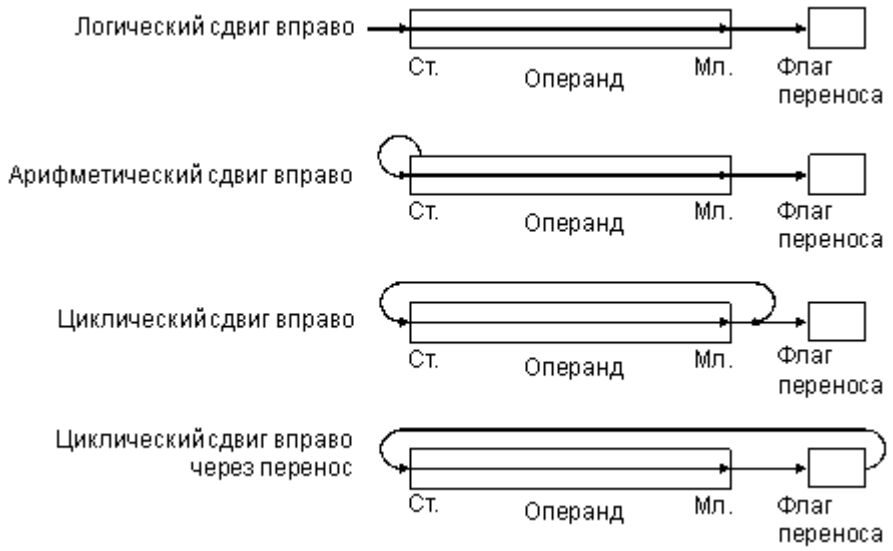

Рис. 3.12. Команды сдвигов вправо.

Наконец, команды установки и очистки битов регистра состояния процессора (то есть флагов) позволяют установить или очистить любой флаг, что бывает очень удобно. Каждому флагу обычно соответствуют две команды, одна из которых устанавливает его в единицу, а другая сбрасывает в нуль. Например, флагу переноса C (от Carry) будут соответствовать команды CLC (очистка) и SEC или STC (установка).

3.3.4. Команды переходов

Команды переходов предназначены для организации всевозможных циклов, ветвлений, вызовов подпрограмм и т.д., то есть они нарушают последовательный ход выполнения программы. Эти команды записывают в регистр-счетчик команд новое значение и тем самым вызывают переход процессора не к следующей по порядку команде, а к любой другой команде в памяти программ. Некоторые команды переходов предусматривают в дальнейшем возврат назад, в точку, из которой был сделан переход, другие не предусматривают этого. Если возврат предусмотрен, то текущие параметры процессора сохраняются в стеке. Если возврат не предусмотрен, то текущие параметры процессора не сохраняются.

Команды переходов без возврата делятся на две группы:

- команды безусловных переходов;
- команды условных переходов.

В обозначениях этих команд используются слова Branch (ветвление) и Jump (прыжок).

Команды безусловных переходов вызывают переход в новый адрес независимо ни от чего. Они могут вызывать переход на указанную величину смещения (вперед или назад) или же на указанный адрес памяти. Величина смещения или новое значение адреса указываются в качестве входного операнда.

Команды условных переходов вызывают переход не всегда, а только при выполнении заданных условий. В качестве таких условий обычно выступают значения флагов в регистре состояния процессора (PSW). То есть условием перехода является результат предыдущей операции, меняющей значения флагов. Всего таких условий перехода может быть от 4 до 16. Несколько примеров команд условных переходов:

- переход, если равно нулю;
- переход, если не равно нулю;

- переход, если есть переполнение;
- переход, если нет переполнения;
- переход, если больше нуля;
- переход, если меньше или равно нулю.

Если условие перехода выполняется, то производится загрузка в регистр-счетчик команд нового значения. Если же условие перехода не выполняется, счетчик команд просто наращивается, и процессор выбирает и выполняет следующую по порядку команду.

Специально для проверки условий перехода применяется команда сравнения (CMP), предшествующая команде условного перехода (или даже нескольким командам условных переходов). Но флаги могут устанавливаться и любой другой командой, например командой пересылки данных, любой арифметической или логической командой. Отметим, что сами команды переходов флаги не меняют, что как раз и позволяет ставить несколько команд переходов одну за другой.

Совместное использование нескольких команд условных и безусловных переходов позволяет процессору выполнять разветвленные алгоритмы любой сложности. Для примера на рис. 3.13 показано разветвление программы на две ветки с последующим соединением, а на рис. 3.14 — разветвление на три ветки с последующим соединением.

Команды переходов с дальнейшим возвратом в точку, из которой был произведен переход, применяются для выполнения подпрограмм, то есть вспомогательных программ. Эти команды называются также командами вызова подпрограмм (распространенное название — CALL). Использование подпрограмм позволяет упростить структуру основной программы, сделать ее более логичной, гибкой, легкой для написания и отладки. В то же время надо учитывать, что широкое использование подпрограмм, как правило, увеличивает время выполнения программы.

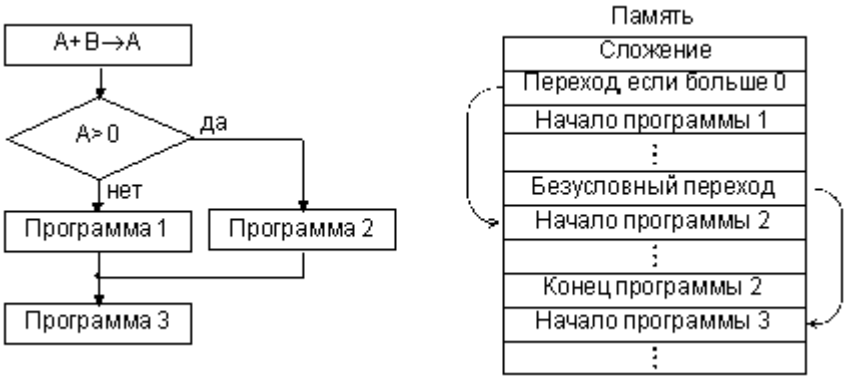

Рис. 3.13. Реализация разветвления на две ветки.

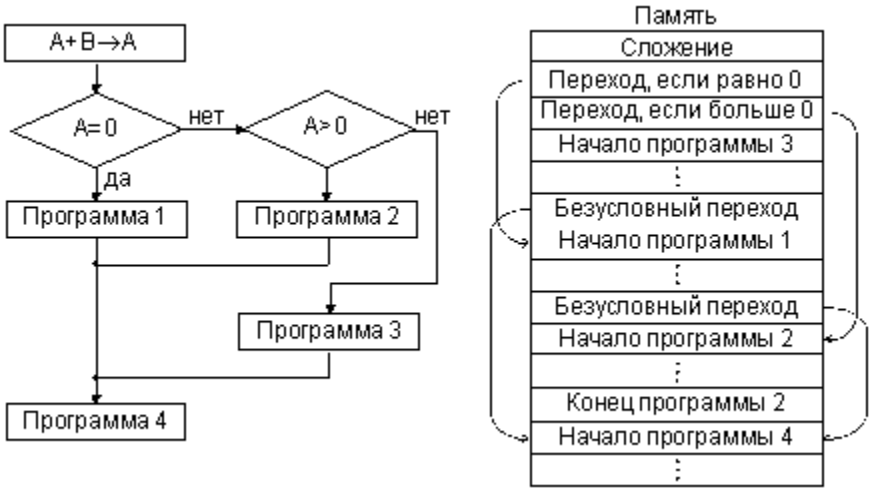

Рис. 3.14. Реализация разветвления на три ветки.

Все команды переходов с возвратом предполагают безусловный переход (они не проверяют никаких флагов). При этом они требуют одного входного операнда, который может указывать как абсолютное значение нового адреса, так и смещение, складываемое с текущим значением адреса. Текущее значение счетчика команд (текущий адрес) сохраняется перед выполнением перехода в стеке.

Для обратного возврата в точку вызова подпрограммы (точку перехода) используется специальная команда возврата (RET или RTS). Эта команда извлекает из стека значение адреса команды перехода и записывает его в регистр-счетчик команд.

Особое место среди команд перехода с возвратом занимают команды прерываний (распространенное название — INT). Эти команды в качестве входного операнда требуют номер прерывания (адрес вектора). Обслуживание таких переходов осуществляется точно так же, как и аппаратных прерываний. То есть для выполнения данного перехода процессор обращается к таблице векторов прерываний и получает из нее по номеру прерывания адрес памяти, в который ему необходимо перейти. Адрес вызова прерывания и содержимое регистра состояния процессора (PSW) сохраняются в стеке. Сохранение PSW — важное отличие команд прерывания от команд переходов с возвратом.

Команды прерываний во многих случаях оказываются удобнее, чем обычные команды переходов с возвратом. Сформировать таблицу векторов прерываний можно один раз, а потом уже обращаться к ней по мере необходимости. Номер прерывания соответствует номеру подпрограммы, то есть номеру функции, выполняемой подпрограммой. Поэтому команды прерывания гораздо чаще включаются в системы команд процессоров, чем обычные команды переходов с возвратом.

Для возврата из подпрограммы, вызванной командой прерывания, используется команда возврата из прерывания (IRET или RTI). Эта команда извлекает из стека сохраненное там значение счетчика команд и регистра состояния процессора (PSW).

Отметим, что у некоторых процессоров предусмотрены также команды условных прерываний, например, команда прерывания при переполнении.

Конечно, в данном разделе мы рассмотрели только основные команды, наиболее часто встречающиеся в процессорах. У конкретных процессоров могут быть и многие другие команды, не относящиеся к перечисленным группам команд. Но изучать их надо уже после того, как выбран тип процессора, подходящий для задачи, решаемой данной микропроцессорной системой.

3.4. Быстродействие процессора

Быстродействие процессора — это одна из важнейших его характеристик, определяющая эффективность работы всей микропроцессорной системы в целом. Быстродействие процессора зависит от множества факторов, что затрудняет сравнение быстродействия даже разных процессоров внутри одного семейства, не говоря уже о процессорах разных фирм и разного назначения.

Выделим важнейшие факторы, влияющие на быстродействие процессора.

Прежде всего, быстродействие зависит от тактовой частоты процессора. Все операции внутри процессора выполняются синхронно, тактируются единым тактовым сигналом. Понятно, что чем больше тактовая частота, тем быстрее работает процессор, причем, например, двукратное увеличение тактовой частоты какого-то процессора снижает вдвое время выполнения команд этим процессором.

Однако надо учитывать, что разные процессоры выполняют одинаковые команды за разное количество тактов, причем количество тактов, затрачиваемых на команду, может изменяться от одного такта до десятков или даже сотен. В некоторых процессорах за счет распараллеливания микроопераций на команду тратится даже меньше одного такта.

Количество тактов, затрачиваемых на выполнение команды, зависит от сложности этой команды и от методов адресации операндов. Например, быстрее всего (за меньшее число тактов) выполняются команды пересылки данных между внутренними регистрами процессора. Медленнее всего (за большое число тактов) выполняются сложные арифметические команды с плавающей запятой, операнды которых хранятся в памяти.

Первоначально для количественной оценки производительности процессоров применялась единица измерения MIPS (Mega Instruction Per Second), соответствовавшая количеству миллионов выполняемых инструкций (команд) за секунду. Естественно, изготовители микропроцессоров старались ориентироваться на самые быстрые команды. Понятно, что подобный показатель не слишком удачен. Для измерения производительности при выполнении вычислений с плавающей запятой (точкой) чуть позже была предложена единица FLOPS (Floating point Operations Per Second), но она по определению узкоспециальная, так как в некоторых системах операции с плавающей запятой просто не используются.

Другой аналогичный показатель быстродействия процессора — время выполнения коротких (быстрых) операций. Для примера в таблице 3.1 представлены показатели быстродействия нескольких 8-разрядных и 16-разрядных процессоров. В настоящее время этот показатель практически не используется, как и MIPS.

Время выполнения команд — важный, но далеко не единственный фактор, определяющий быстродействие. Большое значение имеет также структура системы команд процессора. Например, некоторым процессорам для выполнения какой-то операции понадобится одна команда, а другим процессорам — несколько команд. Какие-то процессоры имеют систему команд, позволяющую быстро решать задачи одного типа, а какие-то — задачи другого типа. Важны и методы адресации, разрешенные в данном процессоре, и наличие сегментирования памяти, и способы взаимодействия процессора с устройствами ввода/вывода и т.д.

Существенно влияет на быстродействие системы в целом и то, как процессор "общается" с памятью команд и памятью данных, применяется ли совмещение выборки команд из памяти с выполнением ранее выбранных команд.

Таблица 3.1. Параметры некоторых процессоров.

Процессор	**8085**	**6800**	**68000**	**8086**
Фирма	Intel	Motorola	Motorola	Intel
Разрядность	8	8	16	16
Количество команд	80	72	61	133
Тактовая частота, МГц	3	1	8	5
Время выполнения коротких операций, мкс	1,3	2	0,5	0,4

Быстродействие системы в целом определяется также и разрядностью процессора. Например, 8-разрядный процессор будет медленнее пересылать и обрабатывать большие массивы данных, чем 16-разрядный процессор. Точно так же 16-разрядный процессор будет значительно медленнее работать с большими числами (большими, чем 65536), чем 32-разрядный процессор.

При высокой сложности решаемых задач быстродействие системы зависит и от общего объема системной памяти. Ведь если системной памяти мало, системе приходится сохранять данные во внеш-

ней памяти (например, на магнитном диске), а это очень сильно (на несколько порядков) замедляет работу. Так что разрядность шины адреса процессора тоже важна.

Поэтому количественные показатели производительности процессоров очень условны, они лишь косвенно характеризуют быстродействие системы на базе этого процессора. Тем не менее, некоторые производители предлагают количественные показатели для своих процессоров, которые характеризуют время выполнения специально составленных тестовых программ, содержащих самые различные команды в тех или иных соотношениях.

Так, для сравнения производительности 32-разрядных процессоров фирма Intel, производящая процессоры для персональных компьютеров, в 1992 году предложила свою единицу измерения iCOMP Index (Intel COmparative Microprocessor Performance). Для вычисления этого показателя используется смесь 16- и 32-битных целочисленных команд, команд с плавающей точкой, команд обработки графики и видео. В качестве базового взят процессор i486SX-25, чей индекс принят равным 100. В Таблице 3.2 приведены индексы iCOMP для некоторых процессоров фирмы Intel. Как видно из таблицы, за счет более развитой архитектуры процессоры семейства 486 всегда быстрее процессоров семейства 386, а любой Pentium быстрее любого процессора из семейства 486. Тактовая частота (указана в таблице через черточку) определяет производительность только в пределах одного семейства. В 1996 году разработчиками Intel был предложен другой показатель — iCOMP Index 2.0, для вычисления которого не используются 16-разрядные команды, зато введен мультимедийный тест, а за базу взят Pentium-120, чей индекс принят равным 100. В таблице 3.3 представлены эти показатели для некоторых типов процессоров Intel.

При этом надо учитывать, что измерения проводятся в составе системы, настроенной на максимальное быстродействие именно данных процессоров, и только самой фирмой Intel.

Ценность этих показателей и всех им подобных не слишком велика. Для конкретного компьютера и разных процессоров величина показателя может предоставить вполне объективные данные, позволяющие оценить, например, целесообразность замены процессора на более мощный. Но усредненность показателей iCOMP не позволяет точно сказать, как будет себя вести процессор в различных задачах, которые ориентированы на преимущественное использование разных типов команд.

Таблица 3.2. Индексы производительности iCOMP.

i486SX-25	100	i486DX4-100	435
i386DX-33	56	Pentium-60	510
i486SX-33	136	Pentium-100	815
i486DX2-66	297	Pentium-133	1110

Таблица 3.3. Индексы производительности iCOMP Index 2.0.

Pentium-100	90	Pentium MMX-166	160
Pentium-120	100	Pentium MMX-233	203
Pentium-150	114	Pentium Pro-200	220
Pentium-200	142	Pentium II-266	303

Точная оценка быстродействия процессора возможна только в составе конкретной системы при решении определенной задачи. Но все перечисленные здесь факторы можно и нужно учитывать при выборе процессора. А количественные показатели помогают сделать выбор.

6. Лекция: Процессорное ядро и память микроконтроллеров

В этой лекции рассказывается о структуре процессорного ядра и особенностях системы команд микроконтроллеров, функциях и организации памяти программ и данных, об использовании стека и внешней памяти микроконтроллеров.

Основной особенностью современного этапа развития МПС является завершение перехода от систем, выполненных на основе нескольких больших ИС, к однокристальным МК, которые объединяют в одном кристалле все основные элементы МПС: центральный процессор (ЦП), постоянное запоминающее устройство (ПЗУ), оперативное запоминающее устройство (ОЗУ), порты ввода/вывода, таймеры.

4.1. Классификация и структура микроконтроллеров

В настоящее время выпускается целый ряд типов МК. Все эти приборы можно условно разделить на три основных класса:

- 8-разрядные МК для встраиваемых приложений;
- 16- и 32-разрядные МК;
- цифровые сигнальные процессоры (DSP).

Наиболее распространенным представителем семейства МК являются 8-разрядные приборы, широко используемые в промышленности, бытовой и компьютерной технике. Они прошли в своем развитии путь от простейших приборов с относительно слаборазвитой периферией до современных многофункциональных контроллеров, обеспечивающих реализацию сложных алгоритмов управления в реальном масштабе времени. Причиной жизнеспособности 8-разрядных МК является использование их для управления реальными объектами, где применяются, в основном, алгоритмы с преобладанием логических операций, скорость обработки которых практически не зависит от разрядности процессора.

Росту популярности 8-разрядных МК способствует постоянное расширение номенклатуры изделий, выпускаемых такими известными фирмами, как Motorola, Microchip, Intel, Zilog, Atmel и многими другими. Современные 8-разрядные МК обладают, как правило, рядом отличительных признаков. Перечислим основные из них:

- модульная организация, при которой на базе одного процессорного ядра (центрального процессора) проектируется ряд (линейка) МК, различающихся объемом и типом памяти программ, объемом памяти данных, набором периферийных модулей, частотой синхронизации;
- использование закрытой архитектуры МК, которая характеризуется отсутствием линий магистралей адреса и данных на выводах корпуса МК. Таким образом, МК представляет собой законченную систему обработки данных, наращивание возможностей которой с использованием параллельных магистралей адреса и данных не предполагается;
- использование типовых функциональных периферийных модулей (таймеры, процессоры событий, контроллеры последовательных интерфейсов, аналого-цифровые преобразователи и др.), имеющих незначительные отличия в алгоритмах работы в МК различных производителей;
- расширение числа режимов работы периферийных модулей, которые задаются в процессе инициализации регистров специальных функций МК.

При модульном принципе построения все МК одного семейства содержат процессорное ядро, одинаковое для всех МК данного семейства, и изменяемый функциональный блок, который отличает МК разных моделей. Структура модульного МК приведена на рис. 4.1.

Процессорное ядро включает в себя:

- центральный процессор;
- внутреннюю контроллерную магистраль (ВКМ) в составе шин адреса, данных и управления;
- схему синхронизации МК;
- схему управления режимами работы МК, включая поддержку режимов пониженного энергопотребления, начального запуска (сброса) и т.д.

Изменяемый функциональный блок включает в себя модули памяти различного типа и объема, порты ввода/вывода, модули тактовых генераторов (Г), таймеры. В относительно простых МК модуль обработки прерываний входит в состав процессорного ядра. В более сложных МК он представляет собой отдельный модуль с развитыми возможностями. В состав изменяемого функционального блока могут входить и такие дополнительные модули как компараторы напряжения, аналого-цифровые преобразователи (АЦП) и другие. Каждый модуль проектируется для работы в составе МК с учетом протокола ВКМ. Данный подход позволяет создавать разнообразные по структуре МК в пределах одного семейства.

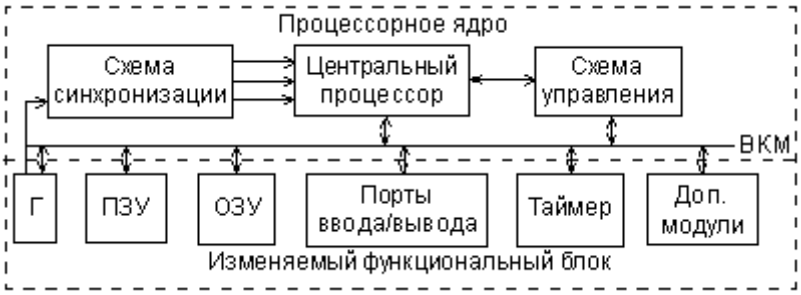

Рис. 4.1. Модульная организация МК.

4.2. Процессорное ядро микроконтроллера

4.2.1. Структура процессорного ядра МК

Основными характеристиками, определяющими производительность процессорного ядра МК, являются:

- набор регистров для хранения промежуточных данных;
- система команд процессора;
- способы адресации операндов в пространстве памяти;
- организация процессов выборки и исполнения команды.

С точки зрения системы команд и способов адресации операндов процессорное ядро современных 8-разрядных МК реализует один из двух принципов построения процессоров:

- процессоры с CISC-архитектурой, реализующие так называемую полную систему команд (Complicated Instruction Set Computer);
- процессоры с RISC-архитектурой, реализующие сокращенную систему команд (Reduced Instruction Set Computer).

CISC-процессоры выполняют большой набор команд с развитыми возможностями адресации, давая разработчику возможность выбрать наиболее подходящую команду для выполнения необходимой операции. В применении к 8-разрядным МК процессор с CISC-архитектурой может иметь однобайтовый, двухбайтовый и трехбайтовый (редко четырехбайтовый) формат команд. При этом система

команд, как правило, неортогональна, то есть не все команды могут использовать любой из способов адресации применительно к любому из регистров процессора. Выборка команды на исполнение осуществляется побайтно в течение нескольких циклов работы МК. Время выполнения команды может составлять от 1 до 12 циклов. К МК с CISC-архитектурой относятся МК фирмы Intel с ядром MCS-51, которые поддерживаются в настоящее время целым рядом производителей, МК семейств HC05, HC08 и HC11 фирмы Motorola и ряд других.

В процессорах с RISC-архитектурой набор исполняемых команд сокращен до минимума. Для реализации более сложных операций приходится комбинировать команды. При этом все команды имеют формат фиксированной длины (например, 12, 14 или 16 бит), выборка команды из памяти и ее исполнение осуществляется за один цикл (такт) синхронизации. Система команд RISC-процессора предполагает возможность равноправного использования всех регистров процессора. Это обеспечивает дополнительную гибкость при выполнении ряда операций. К МК с RISC-процессором относятся МК AVR фирмы Atmel, МК PIC16 и PIC17 фирмы Microchip и другие.

На первый взгляд, МК с RISC-процессором должны иметь более высокую производительность по сравнению с CISC МК при одной и той же тактовой частоте внутренней магистрали. Однако на практике вопрос о производительности более сложен и неоднозначен.

Во-первых, оценка производительности МК по времени выполнения команд различных систем (RISC и CISC) не совсем корректна. Обычно производительность МП и МК принято оценивать числом операций пересылки "регистр-регистр", которые могут быть выполнены в течение одной секунды. В МК с CISC-процессором время выполнения операции "регистр-регистр" составляет от 1 до 3 циклов, что, казалось бы, уступает производительности МК с RISC-процессором. Однако стремление к сокращению формата команд при сохранении ортогональности системы команд RISC-процессора приводит к вынужденному ограничению числа доступных в одной команде регистров. Так, например, системой команд МК PIC16 предусмотрена возможность пересылки результата операции только в один из двух регистров — регистр-источник операнда f или рабочий регистр W. Таким образом, операция пересылки содержимого одного из доступных регистров в другой (не источник операнда и не рабочий) потребует использования двух команд. Такая необходимость часто возникает при пересылке содержимого одного из регистров общего назначения (РОН) в один из портов МК. В то же время, в системе команд большинства CISC-процессоров присутствуют команды пересылки содержимого РОН в один из портов ввода/вывода. То есть более сложная система команд иногда позволяет реализовать более эффективный способ выполнения операции.

Во-вторых, оценка производительности МК по скорости пересылки "регистр-регистр" не учитывает особенностей конкретного реализуемого алгоритма управления. Так, при разработке быстродействующих устройств автоматизированного управления основное внимание следует уделять времени выполнения операций умножения и деления при реализации уравнений различных передаточных функций. А при реализации пульта дистанционного управления бытовой техникой следует оценивать время выполнения логических функций, которые используются при опросе клавиатуры и генерации последовательной кодовой посылки управления. Поэтому в критических ситуациях, требующих высокого быстродействия, следует оценивать производительность на множестве тех операций, которые преимущественно используются в алгоритме управления и имеют ограничения по времени выполнения.

В-третьих, необходимо еще учитывать, что указанные в справочных данных на МК частоты синхронизации обычно соответствуют частоте подключаемого кварцевого резонатора, в то время как длительность цикла центрального процессора определяется частотой обмена по ВКМ. Соотношение этих частот индивидуально для каждого МК и должно быть принято в расчет при сравнении производительности различных моделей контроллеров.

С точки зрения организации процессов выборки и исполнения команды в современных 8-разрядных МК применяется одна из двух уже упоминавшихся архитектур МПС: фон-неймановская (принстонская) или гарвардская.

Основной особенностью фон-неймановской архитектуры является использование общей памяти для хранения программ и данных, как показано на рис. 4.2.

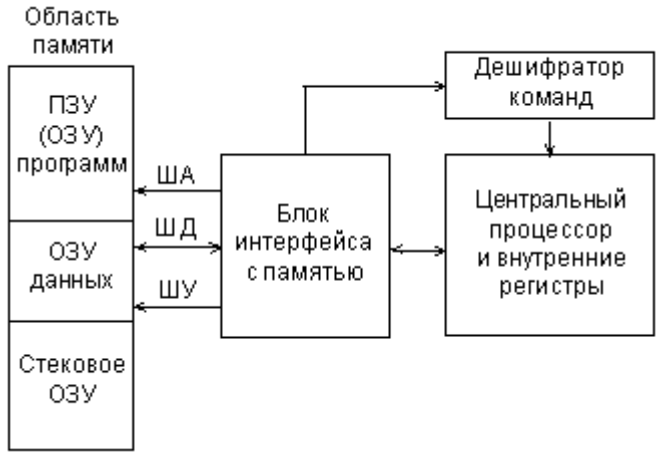

Рис. 4.2. Структура МПС с фон-неймановской архитектурой.

Основное преимущество архитектуры Фон-Неймана – упрощение устройства МПС, так как реализуется обращение только к одной общей памяти. Кроме того, использование единой области памяти позволяло оперативно перераспределять ресурсы между областями программ и данных, что существенно повышало гибкость МПС с точки зрения разработчика программного обеспечения. Размещение стека в общей памяти облегчало доступ к его содержимому. Неслучайно поэтому фон-неймановская архитектура стала основной архитектурой универсальных компьютеров, включая персональные компьютеры.

Основной особенностью гарвардской архитектуры является использование раздельных адресных пространств для хранения команд и данных, как показано на рис. 4.3.

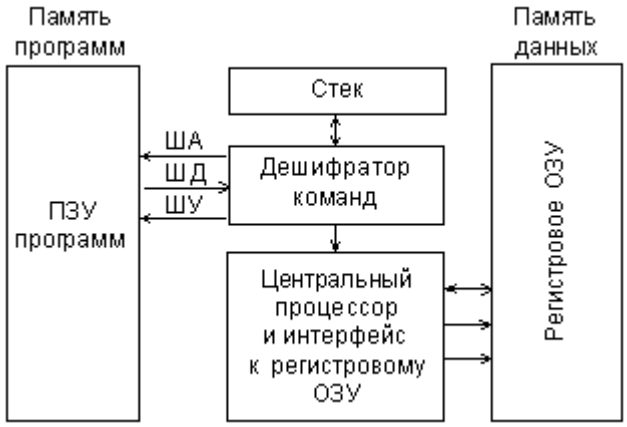

Рис. 4.3. Структура МПС с гарвардской архитектурой.

Гарвардская архитектура почти не использовалась до конца 70-х годов, пока производители МК не поняли, что она дает определенные преимущества разработчикам автономных систем управления.

Дело в том, что, судя по опыту использования МПС для управления различными объектами, для реализации большинства алгоритмов управления такие преимущества фон-неймановской архитектуры как гибкость и универсальность не имеют большого значения. Анализ реальных программ управления показал, что необходимый объем памяти данных МК, используемый для хранения промежуточных результатов, как правило, на порядок меньше требуемого объема памяти программ. В этих условиях использование единого адресного пространства приводило к увеличению формата команд

за счет увеличения числа разрядов для адресации операндов. Применение отдельной небольшой по объему памяти данных способствовало сокращению длины команд и ускорению поиска информации в памяти данных.

Кроме того, гарвардская архитектура обеспечивает потенциально более высокую скорость выполнения программы по сравнению с фон-неймановской за счет возможности реализации параллельных операций. Выборка следующей команды может происходить одновременно с выполнением предыдущей, и нет необходимости останавливать процессор на время выборки команды. Этот метод реализации операций позволяет обеспечивать выполнение различных команд за одинаковое число тактов, что дает возможность более просто определить время выполнения циклов и критичных участков программы.

Большинство производителей современных 8-разрядных МК используют гарвардскую архитектуру. Однако гарвардская архитектура является недостаточно гибкой для реализации некоторых программных процедур. Поэтому сравнение МК, выполненных по разным архитектурам, следует проводить применительно к конкретному приложению.

4.2.2. Система команд процессора МК

Так же, как и в любой микропроцессорной системе, набор команд процессора МК включает в себя четыре основные группы команд:

- команды пересылки данных;
- арифметические команды;
- логические команды;
- команды переходов.

Для реализации возможности независимого управления разрядами портов (регистров) в большинстве современных МК предусмотрена также группа команд битового управления (булевый или битовый процессор). Наличие команд битового процессора позволяет существенно сократить объем кода управляющих программ и время их выполнения.

В ряде МК выделяют также группу команд управления ресурсами контроллера, используемую для настройки режимов работы портов ввода/вывода, управления таймером и т.п. В большинстве современных МК внутренние ресурсы контроллера отображаются на память данных, поэтому для целей управления ресурсами используются команды пересылки данных.

Система команд МК по сравнению с системой команд универсального МП имеет, как правило, менее развитые группы арифметических и логических команд, зато более мощные группы команд пересылки данных и управления. Эта особенность связана со сферой применения МК, требующей, прежде всего, контроля окружающей обстановки и формирования управляющих воздействий.

4.2.3. Схема синхронизации МК

Схема синхронизации МК обеспечивает формирование сигналов синхронизации, необходимых для выполнения командных циклов центрального процессора, а также обмена информацией по внутренней магистрали. В зависимости от исполнения центрального процессора командный цикл может включать в себя от одного до нескольких (4 — 6) тактов синхронизации. Схема синхронизации формирует также метки времени, необходимые для работы таймеров МК. В состав схемы синхронизации входят делители частоты, которые формируют необходимые последовательности синхросигналов.

4.3. Память программ и данных МК

В МК используется три основных вида памяти. Память программ представляет собой постоянную память (ПЗУ), предназначенную для хранения программного кода (команд) и констант. Ее содержимое в ходе выполнения программы не изменяется. Память данных предназначена для хранения пе-

ременных в процессе выполнения программы и представляет собой ОЗУ. Регистры МК — этот вид памяти включает в себя внутренние регистры процессора и регистры, которые служат для управления периферийными устройствами (регистры специальных функций).

4.3.1. Память программ

Основным свойством памяти программ является ее энергонезависимость, то есть возможность хранения программы при отсутствии питания. С точки зрения пользователей МК следует различать следующие типы энергонезависимой памяти программ:

- ПЗУ масочного типа — mask-ROM. Содержимое ячеек ПЗУ этого типа заносится при ее изготовлении с помощью масок и не может быть впоследствии заменено или допрограммировано. Поэтому МК с таким типом памяти программ следует использовать только после достаточно длительной опытной эксплуатации. Основным недостатком данной памяти является необходимость значительных затрат на создание нового комплекта фотошаблонов и их внедрение в производство. Обычно такой процесс занимает 2-3 месяца и является экономически выгодным только при выпуске десятков тысяч приборов. ПЗУ масочного типа обеспечивают высокую надежность хранения информации по причине программирования в заводских условиях с последующим контролем результата.
- ПЗУ, программируемые пользователем, с ультрафиолетовым стиранием — EPROM (Erasable Programmable ROM). ПЗУ данного типа программируются электрическими сигналами и стираются с помощью ультрафиолетового облучения. Ячейка памяти EPROM представляет собой МОП-транзистор с "плавающим" затвором, заряд на который переносится с управляющего затвора при подаче соответствующих электрических сигналов. Для стирания содержимого ячейки она облучается ультрафиолетовым светом, который сообщает заряду на плавающем затворе энергию, достаточную для преодоления потенциального барьера и стекания на подложку. Этот процесс может занимать от нескольких секунд до нескольких минут. МК с EPROM допускают многократное программирование и выпускаются в керамическом корпусе с кварцевым окошком для доступа ультрафиолетового света. Такой корпус стоит довольно дорого, что значительно увеличивает стоимость МК. Для уменьшения стоимости МК с EPROM его заключают в корпус без окошка (версия EPROM с однократным программированием).
- ПЗУ, однократно программируемые пользователем, — OTPROM (One-Time Programmable ROM). Представляют собой версию EPROM, выполненную в корпусе без окошка для уменьшения стоимости МК на его основе. Сокращение стоимости при использовании таких корпусов настолько значительно, что в последнее время эти версии EPROM часто используют вместо масочных ПЗУ.
- ПЗУ, программируемые пользователем, с электрическим стиранием — EEPROM (Electrically Erasable Programmable ROM). ПЗУ данного типа можно считать новым поколением EPROM, в которых стирание ячеек памяти производится также электрическими сигналами за счет использования туннельных механизмов. Применение EEPROM позволяет стирать и программировать МК, не снимая его с платы. Таким способом можно производить отладку и модернизацию программного обеспечения. Это дает огромный выигрыш на начальных стадиях разработки микроконтроллерных систем или в процессе их изучения, когда много времени уходит на поиск причин неработоспособности системы и выполнение циклов стирания-программирования памяти программ. По цене EEPROM занимают среднее положение между OTPROM и EPROM. Технология программирования памяти EEPROM допускает побайтовое стирание и программирование ячеек. Несмотря на очевидные преимущества EEPROM, только в редких моделях МК такая память используется для хранения программ. Связано это с тем, что, во-первых, EEPROM имеют ограниченный объем памяти. Во-вторых, почти одновременно с EEPROM появились Flash-ПЗУ, которые при сходных потребительских характеристиках имеют более низкую стоимость;
- ПЗУ с электрическим стиранием типа Flash — Flash-ROM. Функционально Flash-память мало отличается от EEPROM. Основное различие состоит в способе стирания записанной информации. В памяти EEPROM стирание производится отдельно для каждой ячейки, а во Flash-памяти стирать можно только целыми блоками. Если необходимо изменить содержимое одной ячейки Flash-памяти, потребуется перепрограммировать весь блок. Упрощение декодирующих схем по сравнению с EEPROM привело к тому, что МК с Flash-памятью становятся конкурентоспособными по отношению не только к МК с однократно программируемыми ПЗУ, но и с масочными ПЗУ также.

4.3.2. Память данных

Память данных МК выполняется, как правило, на основе статического ОЗУ. Термин "статическое" означает, что содержимое ячеек ОЗУ сохраняется при снижении тактовой частоты МК до сколь угодно малых значений (с целью снижения энергопотребления). Большинство МК имеют такой параметр, как "напряжение хранения информации" — $U_{STANDBY}$. При снижении напряжения питания ниже минимально допустимого уровня U_{DDMIN}, но выше уровня $U_{STANDBY}$ работа программы МК выполняться не будет, но информация в ОЗУ сохраняется. При восстановлении напряжения питания можно будет сбросить МК и продолжить выполнение программы без потери данных. Уровень напряжения хранения составляет обычно около 1 В, что позволяет в случае необходимости перевести МК на питание от автономного источника (батареи) и сохранить в этом режиме данные ОЗУ.

Объем памяти данных МК, как правило, невелик и составляет обычно десятки и сотни байт. Это обстоятельство необходимо учитывать при разработке программ для МК. Так, при программировании МК константы, если возможно, не хранятся как переменные, а заносятся в ПЗУ программ. Максимально используются аппаратные возможности МК, в частности, таймеры. Прикладные программы должны ориентироваться на работу без использования больших массивов данных.

4.3.3. Регистры МК

Как и все МПС, МК имеют набор регистров, которые используются для управления его ресурсами. В число этих регистров входят обычно регистры процессора (аккумулятор, регистры состояния, индексные регистры), регистры управления (регистры управления прерываниями, таймером), регистры, обеспечивающие ввод/вывод данных (регистры данных портов, регистры управления параллельным, последовательным или аналоговым вводом/выводом). Обращение к этим регистрам может производиться по-разному.

В МК с RISC-процессором все регистры (часто и аккумулятор) располагаются по явно задаваемым адресам. Это обеспечивает более высокую гибкость при работе процессора.

Одним из важных вопросов является размещение регистров в адресном пространстве МК. В некоторых МК все регистры и память данных располагаются в одном адресном пространстве. Это означает, что память данных совмещена с регистрами. Такой подход называется "отображением ресурсов МК на память".

В других МК адресное пространство устройств ввода/вывода отделено от общего пространства памяти. Отдельное пространство ввода/вывода дает некоторое преимущество процессорам с гарвардской архитектурой, обеспечивая возможность считывать команду во время обращения к регистру ввода/вывода.

4.3.4. Стек МК

В микроконтроллерах ОЗУ данных используется также для организации вызова подпрограмм и обработки прерываний. При этих операциях содержимое программного счетчика и основных регистров (аккумулятор, регистр состояния и другие) сохраняется и затем восстанавливается при возврате к основной программе.

В фон-неймановской архитектуре единая область памяти используется, в том числе, и для реализации стека. При этом снижается производительность устройства, так как одновременный доступ к различным видам памяти невозможен. В частности, при выполнении команды вызова подпрограммы следующая команда выбирается после того, как в стек будет помещено содержимое программного счетчика.

В гарвардской архитектуре стековые операции производятся в специально выделенной для этой цели памяти. Это означает, что при выполнении программы вызова подпрограмм процессор с гарвардской архитектурой производит несколько действий одновременно.

Необходимо помнить, что МК обеих архитектур имеют ограниченную емкость памяти для хранения данных. Если в процессоре имеется отдельный стек и объем записанных в него данных превышает его емкость, то происходит циклическое изменение содержимого указателя стека, и он начинает ссылаться на ранее заполненную ячейку стека. Это означает, что после слишком большого количества вызовов подпрограмм в стеке окажется неправильный адрес возврата. Если МК использует общую область памяти для размещения данных и стека, то существует опасность, что при переполнении стека произойдет запись в область данных либо будет сделана попытка записи загружаемых в стек данных в область ПЗУ.

4.3.5. Внешняя память

Несмотря на существующую тенденцию по переходу к закрытой архитектуре МК, в некоторых случаях возникает необходимость подключения дополнительной внешней памяти (как памяти программ, так и данных).

Если МК содержит специальные аппаратные средства для подключения внешней памяти, то эта операция производится штатным способом (как для МП).

Второй, более универсальный, способ заключается в том, чтобы использовать порты ввода/вывода для подключения внешней памяти и реализовать обращение к памяти программными средствами. Такой способ позволяет задействовать простые устройства ввода/вывода без реализации сложных шинных интерфейсов, однако приводит к снижению быстродействия системы при обращении к внешней памяти.

7. Лекция: Организация связи микроконтроллера с внешней средой и временем

В этой лекции рассказывается о структуре и организации портов ввода/вывода информации микроконтроллеров, режимах работы таймеров и процессоров событий, а также об организации обработки прерываний.

4.4. Порты ввода/вывода

Каждый МК имеет некоторое количество линий ввода/вывода, которые объединены в многоразрядные (чаще 8-разрядные) параллельные порты ввода/вывода. В памяти МК каждому порту ввода/вывода соответствует свой адрес регистра данных. Обращение к регистру данных порта ввода/вывода производится теми же командами, что и обращение к памяти данных. Кроме того, во многих МК отдельные разряды портов могут быть опрошены или установлены командами битового процессора.

В зависимости от реализуемых функций различают следующие типы параллельных портов:

- однонаправленные порты, предназначенные только для ввода или только для вывода информации;
- двунаправленные порты, направление передачи которых (ввод или вывод) определяется в процессе инициализации МК;
- порты с альтернативной функцией (мультиплексированные порты). Отдельные линии этих портов используются совместно со встроенными периферийными устройствами МК, такими как таймеры, АЦП, контроллеры последовательных интерфейсов;
- порты с программно управляемой схемотехникой входного/выходного буфера.

Порты выполняют роль устройств временного согласования функционирования МК и объекта управления, которые в общем случае работают асинхронно. Различают три типа алгоритмов обмена информацией между МК и внешним устройством через параллельные порты ввода/вывода:

- режим простого программного ввода/вывода;

- режим ввода/вывода со стробированием;
- режим ввода/вывода с полным набором сигналов подтверждения обмена.

Типичная схема двунаправленного порта ввода/вывода МК приведена на рис. 4.4.

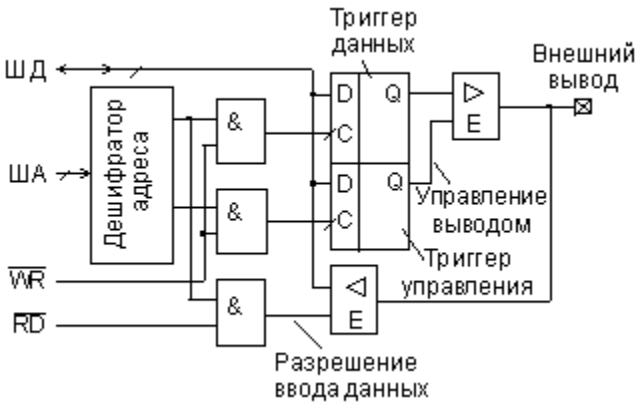

Рис. 4.4. Типовая схема двунаправленного порта ввода/вывода МК.

Триггер управления разрешает вывод данных на внешний вывод. В современных МК, как правило, обеспечивается индивидуальный доступ к триггерам данных и управления, что позволяет использовать каждую линию независимо в режиме ввода или вывода.

Необходимо обратить особое внимание на то, что при вводе данных считывается значение сигнала, поступающее на внешний вывод, а не содержимое триггера данных. Если к внешнему выводу МК подключены выходы других устройств, то они могут установить свой уровень выходного сигнала, который и будет считан вместо ожидаемого значения триггера данных.

Другим распространенным вариантом схемотехнической организации порта ввода/вывода является вывод с "открытым истоком", называемый еще "квазидвунаправленным". Такая организация вывода позволяет создавать шины с объединением устройств по схеме "монтажное И".

4.5. Таймеры и процессоры событий

Большинство задач управления, которые реализуются с помощью МК, требуют исполнения их в реальном времени. Под этим понимается способность системы получить информацию о состоянии управляемого объекта, выполнить необходимые расчетные процедуры и выдать управляющие воздействия в течение интервала времени, достаточного для желаемого изменения состояния объекта.

Возлагать функции формирования управления в реальном масштабе времени только на центральный процессор неэффективно, так как это занимает ресурсы, необходимые для расчетных процедур. Поэтому в большинстве современных МК используется аппаратная поддержка работы в реальном времени с использованием таймера (таймеров).

Модули таймеров служат для приема информации о времени наступления тех или иных событий от внешних датчиков событий, а также для формирования управляющих воздействий во времени.

Модуль таймера 8-разрядного МК представляет собой 8-ми или 16-разрядный счетчик со схемой управления. Схемотехникой МК обычно предусматривается возможность использования таймера в режиме счетчика внешних событий, поэтому его часто называют таймером/счетчиком. Структура типичного 16-разрядного таймера/счетчика в составе МК приведена на рис. 4.5.

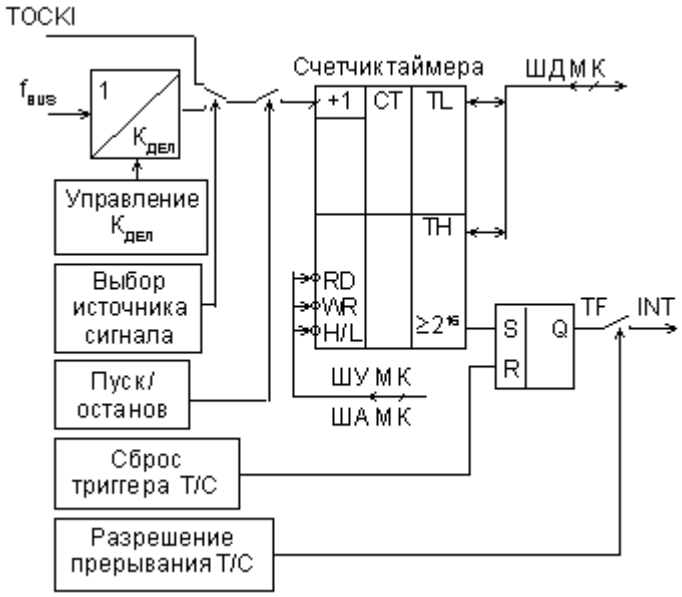

Рис. 4.5. Структура модуля таймера/счетчика.

В памяти МК 16-разрядный счетчик отображается двумя регистрами: TH — старший байт счетчика, TL — младший байт. Регистры доступны для чтения и для записи. Направление счета — только прямое, то есть при поступлении входных импульсов содержимое счетчика инкрементируется. В зависимости от настройки счетчик может использовать один из источников входных сигналов:

- импульсную последовательность с выхода управляемого делителя частоты f_{BUS};
- сигналы внешних событий, поступающие на вход TOCKI контроллера.

В первом случае говорят, что счетчик работает в режиме таймера, во втором — в режиме счетчика событий. При переполнении счетчика устанавливается в "единицу" триггер переполнения TF, который генерирует запрос на прерывание, если прерывания от таймера разрешены. Пуск и останов таймера могут осуществляться только под управлением программы. Программным способом можно также установить старший и младший биты счетчика в произвольное состояние или прочитать текущий код счетчика.

Рассмотренный "классический" модуль таймера/счетчика широко применяется в различных моделях относительно простых МК. Он может использоваться для измерения временных интервалов и формирования последовательности импульсов. Основными недостатками "классического" таймера/счетчика являются:

- потери времени на выполнение команд пуска и останова таймера, приводящие к появлению ошибки при измерении временных интервалов и ограничивающие минимальную длительность измеряемых интервалов времени единицами мс;
- сложности при формировании временных интервалов (меток времени), отличных от периода полного коэффициента счета, равного $(K_{дел}/f_{BUS}) \cdot 2^{16}$;
- невозможность одновременного обслуживания (измерения или формирования импульсного сигнала) сразу нескольких каналов.

Первые из двух перечисленных недостатков были устранены в усовершенствованном модуле таймера/счетчика, используемом в МК семейства MCS-51 (Intel). Дополнительная логика счетного входа позволяет тактовым импульсам поступать на вход счетчика, если уровень сигнала на одной из линий ввода равен "1". Такое решение повышает точность измерения временных интервалов, так как пуск и останов таймера производится аппаратно. Также в усовершенствованном таймере реализован режим перезагрузки счетчика произвольным кодом в момент переполнения. Это позволяет формировать временные последовательности с периодом, отличным от периода полного коэффициента счета.

Однако эти усовершенствования не устраняют главного недостатка модуля "классического" таймера — одноканального режима работы. Совершенствование подсистемы реального времени МК ведется по следующим направлениям:

- увеличение числа модулей таймеров/счетчиков. Этот путь характерен для фирм, выпускающих МК со структурой MCS-51, а также для МК компаний Mitsubishi и Hitachi;
- модификация структуры модуля таймера/счетчика, при которой увеличение числа каналов достигается не за счет увеличения числа счетчиков, а за счет введения дополнительных аппаратных средств входного захвата (input capture — IC) и выходного сравнения (output compare — OC). Такой подход используется, в частности, в МК компании Motorola.

Принцип действия канала входного захвата таймера/счетчика иллюстрирует рис. 4.6.

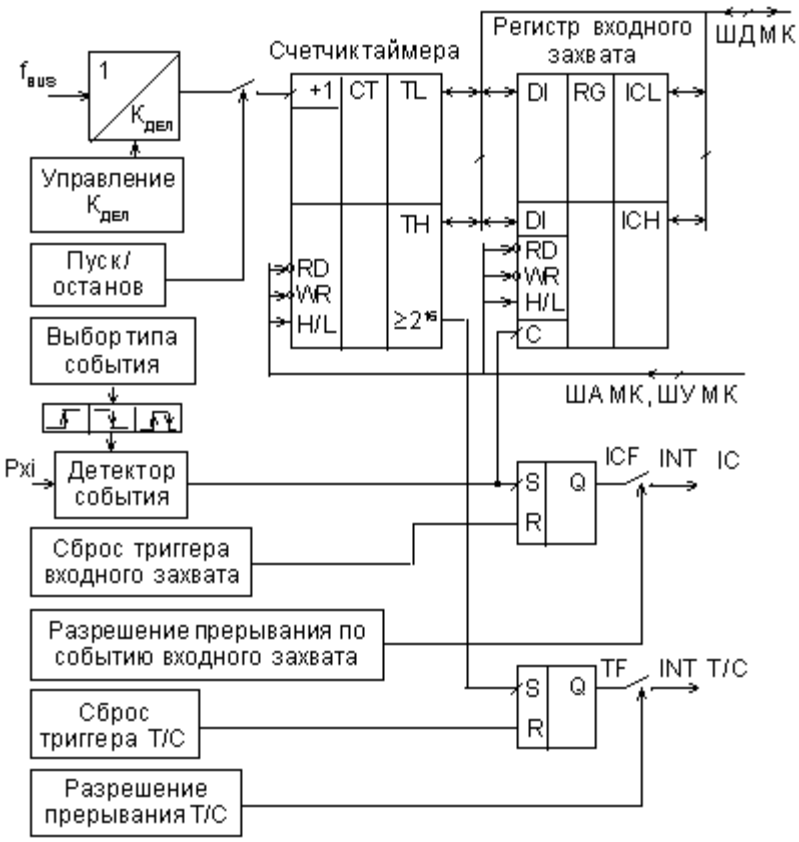

Рис. 4.6. Структурная схема канала входного захвата таймера.

Схема детектора события "наблюдает" за уровнем напряжения на одном из входов МК. Чаще всего это одна из линий порта ввода/вывода. При изменении уровня логического сигнала с "0" на "1" и наоборот вырабатывается строб записи, и текущее состояние счетчика таймера записывается в 16-разрядный регистр входного захвата. Описанное действие в микропроцессорной технике называют событием захвата. Предусмотрена возможность выбора типа сигнала на входе, и это воспринимается как событие:

- положительный (передний) фронт сигнала;
- отрицательный (задний) фронт сигнала;
- любое изменение логического уровня сигнала.

Выбор типа события захвата устанавливается в процессе инициализации таймера и может неоднократно изменяться в ходе выполнения программы. Каждое событие захвата приводит к установке в "1" триггера входного захвата и появлению на его выходе флага (признака) входного захвата ICF. Состояние триггера входного захвата может быть считано программно, а если прерывания по событию захвата разрешены — формируется запрос на прерывание INT IC.

Использование режима входного захвата позволяет исключить ошибки измерения входного интервала времени, связанные со временем перехода к подпрограмме обработки прерывания, так как копирование текущего состояния счетчика осуществляется аппаратными, а не программными средствами. Однако время перехода на подпрограмму обработки прерывания накладывает ограничение на длительность измеряемого интервала времени, так как предполагается, что второе событие захвата произойдет позже, чем код первого события будет считан МК.

Структура аппаратных средств канала выходного сравнения представлена на рис. 4.7.

Цифровой компаратор непрерывно сравнивает текущий код счетчика таймера с кодом, который записан в 16-разрядном регистре выходного сравнения. В момент равенства кодов на одном из выходов МК (Pxj на рис. 4.7) устанавливается заданный уровень логического сигнала. Обычно предусмотрено три типа изменения сигнала на выходе Pxj в момент события выходного сравнения:

- установка высокого логического уровня;
- установка низкого логического уровня;
- инвертирование сигнала на выходе.

При наступлении события сравнения устанавливаются в "1" триггер выходного сравнения и соответствующий ему признак выходного сравнения OCF. Аналогично режиму входного захвата состояние триггера выходного сравнения может быть считано программно, а если прерывания по событию сравнения разрешены — формируется запрос на прерывание INT OC.

Режим выходного сравнения предназначен, прежде всего, для формирования временных интервалов заданной длительности. Длительность сформированного временного интервала определяется только разностью кодов, последовательно загружаемых в регистр выходного сравнения, и не зависит от программного обеспечения МК. Время, необходимое для записи нового значения кода в регистр канала сравнения, ограничивает минимальную длительность формируемого временного интервала.

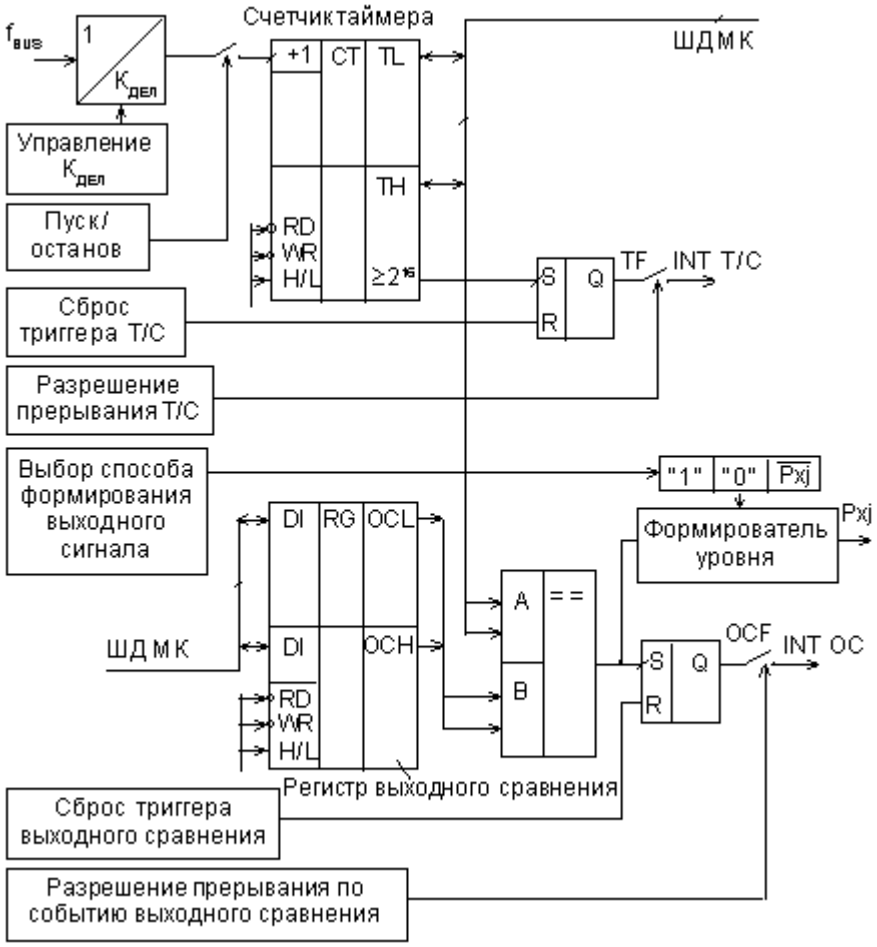

Рис. 4.7. Структурная схема канала выходного сравнения таймера.

Модули усовершенствованного таймера используются в составе МК в различных модификациях. При этом число каналов входного захвата и выходного сравнения в модуле может быть различным. Так, в МК семейства HC05 фирмы Motorola типовыми решениями являются модули 1IC+1OC или 2IC+2OC, а модуль таймера в составе МК только один. В ряде модулей каналы могут быть произвольно настроены на функцию входного захвата или выходного сравнения посредством инициализации. Счетчик модуля усовершенствованного таймера может не иметь функции программного останова. В этом случае состояние счетчика нельзя синхронизировать с каким-либо моментом работы МК, и такой счетчик характеризуется как свободно считающий (free counter).

Аппаратные средства усовершенствованного таймера позволяют решить многие задачи управления в реальном времени. Однако по мере роста сложности алгоритмов управления отчетливо проявляются ограничения модулей усовершенствованного таймера, а именно:

- недостаточное число каналов захвата и сравнения, принадлежащих одному счетчику временной базы. Это не позволяет сформировать синхронизированные между собой многоканальные импульсные последовательности;
- однозначно определенная конфигурация канала (или захват или сравнение) часто не удовлетворяет потребностям решаемой задачи;
- формирование сигналов по методу широтно-импульсной модуляции (ШИМ) требует программной поддержки, что снижает максимально достижимую частоту выходного сигнала.

Поэтому следующим этапом развития модулей подсистемы реального времени МК стали модули процессоров событий. Впервые модули процессоров событий были использованы компанией Intel в МК семейства 8xC51Fx. Этот модуль получил название программируемого счетного массива (Programmable Counter Array — PCA).

PCA обеспечивает более широкие возможности работы в реальном масштабе времени и в меньшей степени расходует ресурсы центрального процессора, чем стандартный и усовершенствованный таймеры/счетчики. К преимуществам PCA также можно отнести более простое программирование и более высокую точность. К примеру, PCA может обеспечить лучшее временное разрешение, чем таймеры 0, 1 и 2 МК семейства MCS-51, так как счетчик PCA способен работать с тактовой частотой, втрое большей, чем у этих таймеров. PCA также может решать многие задачи, выполнение которых с использованием таймеров требует дополнительных аппаратных затрат (например, определение фазового сдвига между импульсами или генерация ШИМ-сигнала). PCA состоит из 16-битного таймера-счетчика и пяти 16-битных модулей сравнения-защелки, как показано на рис. 4.8.

Таймер-счетчик PCA используется в качестве базового таймера для функционирования всех пяти модулей сравнения-защелки. Вход таймера-счетчика PCA может быть запрограммирован на счет сигналов от следующих источников:

- выход делителя на 12 тактового генератора МК;
- выход делителя на 4 тактового генератора МК;
- сигнал переполнения таймера 0;
- внешний входной сигнал на выводе ECI (P1.2).

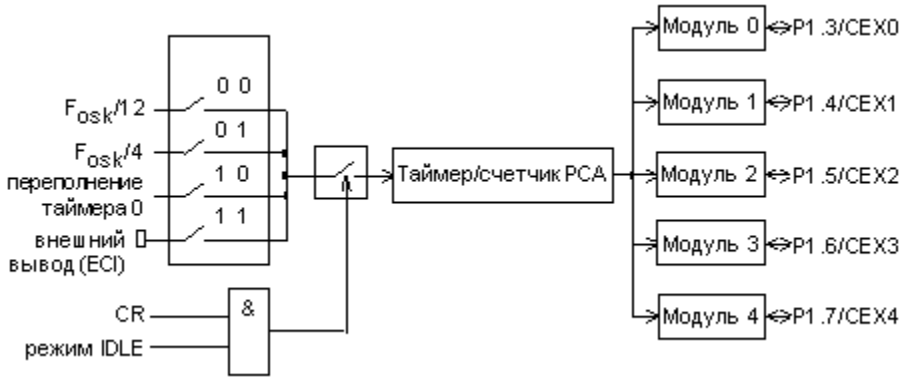

Рис. 4.8. Структура процессора событий МК семейства Intel 8xC51Fx.

Любой из модулей сравнения-защелки может быть запрограммирован для работы в следующих режимах:

- защелкивания по фронту и/или спаду импульса на входе CEXi;
- программируемого таймера;
- высокоскоростного выхода;
- широтно-импульсного модулятора.

Модуль 4 может быть также запрограммирован как сторожевой таймер (Watchdog Timer – WDT).

Режим защелкивания по импульсу на входе МК эквивалентен режиму входного захвата (IC) усовершенствованного таймера. Режимы программируемого таймера и высокоскоростного выхода близки по своим функциональным возможностям к режиму выходного сравнения (ОС).

В режиме ШИМ на соответствующем выводе МК формируется последовательность импульсов с периодом, равным периоду базового таймера/счетчика PCA. Значение 8-разрядного кода, записанное в младший байт регистра-защелки соответствующего модуля задает скважность формируемого сигнала. При изменении кода от 0 до 255 скважность меняется от 100 % до 0,4 %.

Режим ШИМ очень прост с точки зрения программного обслуживания. Если изменения скважности не предполагается, то достаточно один раз занести соответствующий код в регистр данных модуля, проинициализировать режим ШИМ, и импульсная последовательность будет воспроизводиться с заданными параметрами без вмешательства программы.

Назначение и особенности работы сторожевого таймера будут рассмотрены далее отдельно.

При работе модуля сравнения-защелки в режиме защелки, программируемого таймера или высокоскоростного выхода модуль может сформировать сигнал прерывания. Сигналы от всех пяти модулей сравнения-защелки и сигнал переполнения таймера PCA разделяют один вектор прерывания. Иными словами, если прерывания разрешены, то и сигнал переполнения таймера PCA и сигнал от любого из модулей вызывают одну и ту же подпрограмму прерываний, которая должна сама идентифицировать источник, вызвавший ее.

Для работы с внешними устройствами таймер-счетчик PCA и модули сравнения-защелки используют выводы P1 порта МК. Если какой-либо вывод порта не используется при работе PCA, или PCA не задействован, порт может применяться стандартным образом.

Реализованный в 8xC51FX PCA оказался настолько удачным, что архитектура данных МК стала промышленным стандартом де-факто, а сам PCA многократно воспроизводился в различных модификациях микроконтроллеров разных фирм.

Тенденция развития подсистемы реального времени современных МК находит свое отражение в увеличении числа каналов процессоров событий и расширении их функциональных возможностей.

4.6. Модуль прерываний МК

Обработка прерываний в МК происходит в соответствии с общими принципами обработки прерываний в МПС. Модуль прерываний принимает запросы прерывания и организует переход к выполнению определенной прерывающей программы. Запросы прерывания могут поступать как от внешних источников, так и от источников, расположенных в различных внутренних модулях МК. В качестве входов для приема запросов от внешних источников чаще всего используются выводы параллельных портов ввода/вывода, для которых эта функция является альтернативной. Источниками запросов внешних прерываний также могут быть любые изменения внешних сигналов на некоторых специально выделенных линиях портов ввода/вывода.

Источниками внутренних запросов прерываний могут служить следующие события:

- переполнение таймеров/счетчиков;
- сигналы от каналов входного захвата и выходного сравнения таймеров/счетчиков или от процессора событий;
- готовность памяти EEPROM;
- сигналы прерывания от дополнительных модулей МК, включая завершение передачи или приема информации по одному из последовательных портов и другие.

Любой запрос прерывания поступает на обработку, если прерывания в МК разрешены и разрешено прерывание по данному запросу. Адрес, который загружается в программный счетчик при переходе к обработке прерывания, называется "вектор прерывания". В зависимости от организации модуля прерываний конкретного МК различные источники прерываний могут иметь разные векторы или использовать некоторые из них совместно. Использование различными прерываниями одного вектора обычно не вызывает проблем при разработке программного обеспечения, так как аппаратная часть МК фиксирована, а контроллер чаще всего выполняет одну-единственную программу.

Вопрос о приоритетах при одновременном поступлении нескольких запросов на прерывание решается в различных МК по-разному. Есть МК с одноуровневой системой приоритетов (все запросы равноценны), многоуровневой системой с фиксированными приоритетами и многоуровневой программируемой системой приоритетов.

Отдельно необходимо описать аппаратные прерывания, связанные с включением питания, подачей сигнала "сброс" и переполнением сторожевого таймера. Они имеют немаскируемый характер и чаще всего разделяют один общий вектор прерывания. Это вполне логично, поскольку результатом каждого из событий является начальный сброс МК.

8. Лекция: Вспомогательные аппаратные средства микроконтроллера

В этой лекции рассказывается о режимах пониженного энергопотребления микроконтроллера, а также о структуре и организации таких вспомогательных аппаратных средств как тактовые генераторы, схемы контроля напряжения питания, сторожевые таймеры и дополнительные модули микроконтроллера.

4.7. Минимизация энергопотребления в системах на основе МК

Малый уровень энергопотребления является зачастую определяющим фактором при выборе способа реализации цифровой управляющей системы. Современные МК предоставляют пользователю большие возможности в плане экономии энергопотребления и имеют, как правило, следующие основные режимы работы:

- **активный режим** (Run mode) — основной режим работы МК. В этом режиме МК исполняет рабочую программу, и все его ресурсы доступны. Потребляемая мощность имеет максимальное значение P_{RUN}. Большинство современных МК выполнено по КМОП-технологии, поэтому мощность потребления в активном режиме сильно зависит от тактовой частоты;
- **режим ожидания** (Wait mode, Idle mode или Halt mode). В этом режиме прекращает работу центральный процессор, но продолжают функционировать периферийные модули, которые контролируют состояние объекта управления. При необходимости сигналы от периферийных модулей переводят МК в активный режим, и рабочая программа формирует необходимые управляющие воздействия. Перевод МК из режима ожидания в рабочий режим осуществляется по прерываниям от внешних источников или периферийных модулей, либо при сбросе МК. В режиме ожидания мощность потребления МК PWAIT снижается по сравнению с активным режимом в 5…10 раз;
- режим останова (Stop mode, Sleep mode или Power Down mode). В этом режиме прекращает работу как центральный процессор, так и большинство периферийных модулей. Переход МК из состояния останова в рабочий режим возможен, как правило, только по прерываниям от внешних источников или после подачи сигнала сброса. В режиме останова мощность потребления МК P_{STOP} снижается по сравнению с активным режимом примерно на три порядка и составляет единицы микроватт.

Два последних режима называют режимами пониженного энергопотребления. Минимизация энергопотребления системы на МК достигается за счет оптимизации мощности потребления МК в активном режиме, а также использования режимов пониженного энергопотребления. При этом необходимо иметь в виду, что режимы ожидания и останова существенно отличаются временем перехода из режима пониженного энергопотребления в активный режим. Выход из режима ожидания обычно происходит в течение 3…5 периодов синхронизации МК, в то время как задержка выхода из режима останова составляет несколько тысяч периодов синхронизации. Кроме снижения динамики работы системы значительное время перехода в активный режим является причиной дополнительного расхода энергии.

Мощность потребления МК в активном режиме является одной из важнейших характеристик контроллера. Она в значительной степени зависит от напряжения питания МК и частоты тактирования.

В зависимости от диапазона питающих напряжений все МК можно разделить на три основные группы:

- МК с напряжением питания 5,0 В±10%. Эти МК предназначены, как правило, для работы в составе устройств с питанием от промышленной или бытовой сети, имеют развитые функциональные возможности и высокий уровень энергопотребления.
- МК с расширенным диапазоном напряжений питания: от 2,0…3,0 В до 5,0-7,0 В. МК данной группы могут работать в составе устройств как с сетевым, так и с автономным питанием.
- МК с пониженным напряжением питания: от 1,8 до 3 В. Эти МК предназначены для работы в устройствах с автономным питанием и обеспечивают экономный расход энергии элементов питания.

Зависимость тока потребления от напряжения питания МК почти прямо пропорциональная. Поэтому снижение напряжения питания весьма существенно понижает мощность потребления МК. Необходимо, однако, иметь в виду, что для многих типов МК с понижением напряжения питания уменьшается максимально допустимая частота тактирования, то есть выигрыш в потребляемой мощности сопровождается снижением производительности системы.

Большинство современных МК выполнено по технологии КМОП, поэтому мощность потребления в активном режиме P_{RUN} практически прямо пропорциональна тактовой частоте. Поэтому, выбирая частоту тактового генератора, не следует стремиться к предельно высокому быстродействию МК в задачах, которые этого не требуют. Часто определяющим фактором оказывается разрешающая способность измерителей или формирователей временных интервалов на основе таймера или скорость передачи данных по последовательному каналу.

В большинстве современных МК используется статическая КМОП-технология, поэтому они способны работать при сколь угодно низких тактовых частотах вплоть до нулевых. В справочных данных при этом указывается, что минимальная частота тактирования равна dc (direct current). Это означает,

что возможно использование МК в пошаговом режиме, например, для отладки. Мощность потребления МК при низких частотах тактирования обычно отражает значение тока потребления при f_{OSC} = 32768 Гц (часовой кварцевый резонатор).

4.8. Тактовые генераторы МК

Современные МК содержат встроенные тактовые генераторы, которые требуют минимального числа внешних времязадающих элементов. На практике используются три основных способа определения тактовой частоты генератора: с помощью кварцевого резонатора, керамического резонатора и внешней RC-цепи.

Типовая схема подключения кварцевого или керамического резонатора приведена на рис. 4.9а.

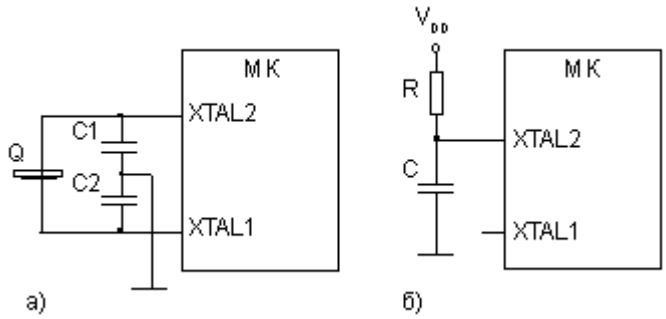

Рис. 4.9. Тактирование с использованием кварцевого или керамического резонаторов (а) и с использованием RC-цепи (б).

Кварцевый или керамический резонатор Q подключается к выводам XTAL1 и XTAL2, которые обычно представляют собой вход и выход инвертирующего усилителя. Номиналы конденсаторов C1 и C2 определяются производителем МК для конкретной частоты резонатора. Иногда требуется включить резистор порядка нескольких мегаом между выводами XTAL1 и XTAL2 для стабильной работы генератора.

Использование кварцевого резонатора позволяет обеспечить высокую точность и стабильность тактовой частоты (разброс частот кварцевого резонатора обычно составляет менее 0,01%). Такой уровень точности требуется для обеспечения точного хода часов реального времени или организации интерфейса с другими устройствами. Основными недостатками кварцевого резонатора являются его низкая механическая прочность (высокая хрупкость) и относительно высокая стоимость.

При менее жестких требованиях к стабильности тактовой частоты возможно использование более стойких к ударной нагрузке керамических резонаторов. Многие керамические резонаторы имеют встроенные конденсаторы, что позволяет уменьшить количество внешних подключаемых элементов с трех до одного. Керамические резонаторы имеют разброс частот порядка нескольких десятых долей процента (обычно около 0,5 %).

Самым дешевым способом задания тактовой частоты МК является использование внешней RC-цепи, как показано на рис. 4.9б. Внешняя RC-цепь не обеспечивает высокой точности задания тактовой частоты (разброс частот может доходить до десятков процентов). Это неприемлемо для многих приложений, где требуется точный подсчет времени. Однако имеется масса практических задач, где точность задания тактовой частоты не имеет большого значения.

Зависимость тактовой частоты МК от номиналов RC-цепи зависит от конкретной реализации внутреннего генератора и приводится в руководстве по применению контроллера.

Практически все МК допускают работу от внешнего источника тактового сигнала, который подключается ко входу XTAL1 внутреннего усилителя. При помощи внешнего тактового генератора можно

задать любую тактовую частоту МК (в пределах рабочего диапазона) и обеспечить синхронную работу нескольких устройств.

Некоторые современные МК содержат встроенные RC или кольцевые генераторы, которые позволяют контроллеру работать без внешних цепей синхронизации. Работа внутреннего генератора обычно разрешается путем программирования соответствующего бита регистра конфигурации МК.

В большинстве моделей МК частота времязадающего элемента (резонатора или RC-цепи) и частота тактирования f_{BUS} жестко связаны коэффициентом деления встроенного делителя частоты. Поэтому изменение частоты программным путем не представляется возможным. Однако ряд последних семейств МК (например, HC08 фирмы Motorola) имеют в своем составе схему тактирования, основанную на принципе синтезатора частоты с контуром фазовой автоподстройки (PLL — phase loop lock). Такая схема работает как умножитель частоты и позволяет задавать тактовую частоту с помощью низкочастотного кварцевого резонатора, что снижает уровень электромагнитного излучения МК. Коэффициенты деления контура PLL могут быть изменены программным путем, что позволяет снизить тактовую частоту (и, соответственно, потребляемую мощность) в промежутки времени, когда высокое быстродействие не требуется.

В некоторых МК семейства AVR фирмы Atmel тактовая частота контроллера, задаваемая внутренней RC-цепью, также может изменяться программными средствами.

4.9. Аппаратные средства обеспечения надежной работы МК

Прикладная программа, записанная в память программ МК, должна обеспечивать его надежную работу при любых комбинациях входных сигналов. Однако в результате электромагнитных помех, колебаний напряжения питания и других внешних факторов предусмотренный разработчиком ход выполнения программы может быть нарушен. С целью обеспечения надежного запуска, контроля работы МК и восстановления работоспособности системы в отсутствие оператора все современные МК снабжаются аппаратными средствами обеспечения надежной работы. К ним относятся:

- схема формирования сигнала сброса МК;
- модуль мониторинга напряжения питания;
- сторожевой таймер.

4.9.1. Схема формирования сигнала сброса МК

При включении напряжения питания МК должен начать выполнять записанную в памяти программу работы. На этапе нарастания напряжения питания МК принудительно переводится в начальное состояние, которое называют состоянием сброса. При этом устанавливаются в исходное состояние внутренние магистрали МК, сигналы управления и регистры специальных функций. Последние определяют начальное состояние периферийных модулей МК, которое чаще всего по умолчанию неактивно.

С целью обеспечения надежного запуска от любых источников питания с различной динамикой нарастания напряжения большинство современных МК содержат встроенный детектор напряжения питания (схема Power-On-Reset — POR), который формирует сигнал сброса при нарастании напряжения питания. В частности, входящий в состав МК семейства PIC16 таймер установления питания (PWRT) начинает отсчет времени после того, как напряжение питания пересекло уровень около 1,2…1,8 В. По истечении выдержки около 72 мс считается, что напряжение достигло номинала.

Сразу после выхода из состояния сброса МК выполняет следующие действия:

- запускает генератор синхронизации МК. Для стабилизации частоты тактирования внутренними средствами формируется задержка времени;
- считывает энергонезависимые регистры конфигурации в соответствующие регистры ОЗУ (если необходимо);

- загружает в счетчик команд адрес начала рабочей программы;
- производит выборку первой программы из памяти программ и приступает к выполнению программы.

Адрес ячейки памяти, в которой хранится код первой исполняемой команды, называют вектором начального запуска или вектором сброса. В некоторых МК этот адрес однозначно определен и приведен в техническом описании. Про такие МК говорят, что они имеют фиксированный вектор сброса. В других МК вектор сброса может быть произвольно определен пользователем. На этапе программирования МК необходимый вектор начального запуска записывается в ячейки с фиксированными адресами, и при выходе МК из сброса автоматически загружается в счетчик команд. О таких МК говорят, что они имеют загружаемый вектор сброса. Загружаемый вектор сброса имеют все 8-разрядные МК фирмы Motorola, выполненные по структуре с единым адресным пространством команд и данных.

Для перевода МК в состояние сброса при установившемся напряжении питания достаточно подать сигнал высокого или низкого уровня (в соответствии со спецификацией МК) на вход сброса (RESET). Обычно для формирования сигнала сброса при включении напряжения питания и нажатии кнопки сброса используют RC-цепь. Типовые схемы формирования сигнала сброса представлены на рис. 4.10.

Кнопка Кн предназначена для "ручного" сброса МК при отладке. Диод VD препятствует попаданию на вход RESET отрицательного напряжения при выключении питания. Номиналы R и C определяют задержку времени, необходимую для завершения всех переходных процессов при сбросе (указываются в техническом описании на МК). Триггер Шмитта на входе допускает подачу сигнала сброса с ненормированной длительностью фронта. При отсутствии триггера Шмитта на входе приходится использовать специальную внешнюю схему формирователя.

В современных МК линия RESET обычно является двунаправленной и имеет низкий активный уровень. При нажатии кнопки "сброс" или включении питания буфер линии устанавливается в режим ввода и реализует так называемый внешний сброс. МК может перейти в состояние сброса также по сигналам устройств контроля состояния, которые имеются в составе контроллера. В этом случае говорят, что МК находится в состоянии внутреннего сброса. При этом буфер линии RESET устанавливается в состояние вывода с низким логическим уровнем на выходе. Данный сигнал может быть использован для установки в начальное состояние периферийных ИС.

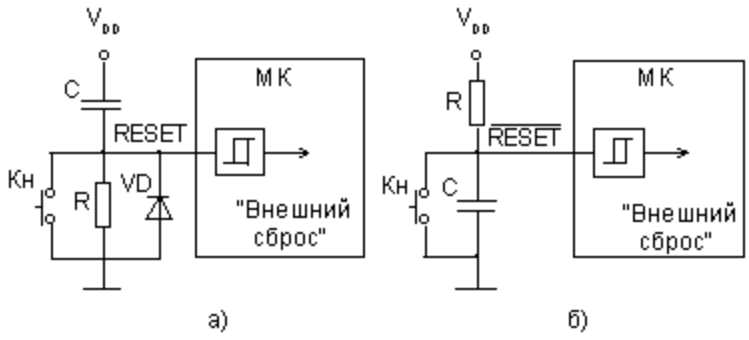

Рис. 4.10. Типовые схемы формирования сигнала внешнего сброса для МК с высоким активным уровнем сигнала сброса (а) и низким активным уровнем сигнала сброса (б).

Порядок выхода МК из состояний внешнего и внутреннего сброса в целом одинаков.

4.9.2. Блок детектирования пониженного напряжения питания

В реальных условиях эксплуатации может сложиться такая ситуация, при которой напряжение питания МК опустится ниже минимально допустимого, но не достигнет порога отпускания схемы POR. В этих условиях МК может "зависнуть". При восстановлении напряжения питания до номинального значения МК останется неработоспособным.

Для восстановления работоспособности системы после "просадки" напряжения питания МК необходимо снова сбросить. Для этой цели в современных МК реализован дополнительный блок детектирования пониженного напряжения питания. Такой модуль используется в МК семейства HC08 фирмы Motorola, аналогичный модуль имеется в составе семейства PIC17 фирмы Microchip. Рассматриваемый модуль генерирует сигнал внутреннего сброса при снижении напряжения питания до уровня чуть ниже минимально допустимого. Уровень срабатывания блока детектирования пониженного напряжения питания значительно превышает напряжение сохранения данных в ОЗУ МК. Событие сброса по сигналу блока пониженного напряжения питания отмечается специальным битом в одном из регистров МК. Следовательно, программно анализируя этот бит после сброса МК, можно установить, что данные целы, и продолжить выполнение программы.

4.9.3. Сторожевой таймер

Если, несмотря на все принятые меры, МК все же "завис" , то на случай выхода из этого состояния все современные контроллеры имеют встроенный модуль сторожевого таймера. Принцип действия сторожевого таймера показан на рис. 4.11.

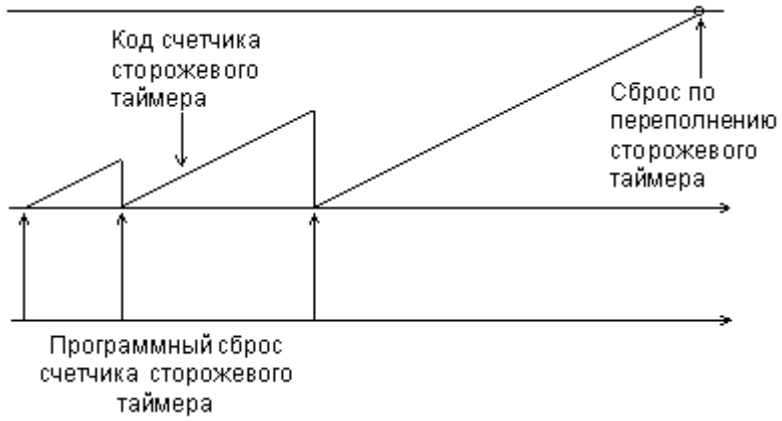

Рис. 4.11. Принцип действия сторожевого таймера.

Основу сторожевого таймера составляет многоразрядный счетчик. При сбросе МК счетчик обнуляется. После перехода МК в активный режим работы значение счетчика начинает увеличиваться независимо от выполняемой программы. При достижении счетчиком максимального кода генерируется сигнал внутреннего сброса, и МК начинает выполнять рабочую программу сначала.

Для исключения сброса по переполнению сторожевого таймера рабочая программа МК должна периодически сбрасывать счетчик. Сброс счетчика сторожевого таймера осуществляется путем исполнения специальной команды (например, CLRWDT) или посредством записи некоторого указанного кода в один из регистров специальных функций. Тогда при нормальном, предусмотренном разработчиком, порядке исполнения рабочей программы переполнения счетчика сторожевого таймера не происходит, и он не оказывает влияния на работу МК. Однако, если исполнение рабочей программы было нарушено, например, вследствие "зависания", то велика вероятность того, что счетчик не будет сброшен вовремя. Тогда произойдет сброс по переполнению сторожевого таймера, и нормальный ход выполнения рабочей программы будет восстановлен.

Модули сторожевых таймеров конкретных МК могут иметь различные особенности:

- в ряде МК векторы внешнего сброса и сброса по переполнению сторожевого таймера совпадают. Это не позволяет выявить причину сброса программным путем и затрудняет написание рабочей программы. Более высокоуровневые МК имеют либо различные векторы сброса, либо отмечают событие сброса по переполнению сторожевого таймера установкой специального бита в одном из регистров специальных функций;
- в некоторых МК при переходе в один из режимов пониженного энергопотребления, когда рабочая программа не выполняется, автоматически приостанавливается работа сторожевого таймера. В других

МК сторожевой таймер имеет независимый тактовый генератор, который продолжает функционировать и в режиме ожидания. В этом случае необходимо периодически выводить МК из состояния ожидания для сброса сторожевого таймера. В PIC-контроллерах фирмы Microchip выработка таких сбросов может быть запрещена путем записи нуля в специальный бит конфигурации WDTE.

Использование сторожевого таймера существенно повышает способность к самовосстановлению системы на основе МК.

4.10. Дополнительные модули МК

Описанные выше модули составляют так называемый базовый комплект МК и входят в состав любого современного контроллера. Очевидна необходимость включения в состав МК дополнительных модулей, состав и возможности которых определяются конкретной решаемой задачей. Среди таких дополнительных модулей следует, прежде всего, отметить:

- модули последовательного ввода/вывода данных;
- модули аналогового ввода/вывода.

4.10.1. Модули последовательного ввода/вывода

Наличие в составе 8-разрядного МК модуля контроллера последовательного ввода/вывода стало в последнее время обычным явлением. Задачи, которые решаются средствами модуля контроллера последовательного ввода/вывода, можно разделить на три основные группы:

- связь встроенной микроконтроллерной системы с системой управления верхнего уровня, например, с персональным компьютером. Чаще всего для этой цели используются интерфейсы RS-232C и RS-485;
- связь с внешними по отношению к МК периферийными ИС, а также с датчиками физических величин с последовательным выходом. Для этих целей используются интерфейсы I^2C, SPI, а также нестандартные протоколы обмена;
- интерфейс связи с локальной сетью в мультимикроконтроллерных системах. В системах с числом МК до пяти обычно используются сети на основе интерфейсов I^2C, RS-232C и RS-485 с собственными сетевыми протоколами высокого уровня. В более сложных системах все более популярным становится протокол CAN.

С точки зрения организации обмена информацией упомянутые типы интерфейсов последовательной связи отличаются режимом передачи данных (синхронный или асинхронный), форматом кадра (число бит в посылке при передаче байта полезной информации) и временными диаграммами сигналов на линиях (уровни сигналов и положение фронтов при переключениях).

Число линий, по которым происходит передача в последовательном коде, обычно равно двум (I^2C, RS-232C, RS-485) или трем (SPI, некоторые нестандартные протоколы). Данное обстоятельство позволяет спроектировать модули контроллеров последовательного обмена таким образом, чтобы с их помощью на аппаратном уровне можно было реализовать несколько типов последовательных интерфейсов. При этом режим передачи (синхронный или асинхронный) и формат кадра поддерживаются на уровне логических сигналов, а реальные физические уровни сигналов для каждого интерфейса получают с помощью специальных ИС, которые называют приемопередатчиками, конверторами, трансиверами.

Среди различных типов встроенных контроллеров последовательного обмена, которые входят в состав тех или иных 8-разрядных МК, сложился стандарт "де-факто" — модуль UART (Universal Asynchronous Receiver and Transmitter). UART — это универсальный асинхронный приемопередатчик. Однако большинство модулей UART, кроме асинхронного режима обмена, способны также реализовать режим синхронной передачи данных.

Не все производители МК используют термин UART для обозначения типа модуля контроллера последовательного обмена. Так, в МК фирмы Motorola модуль асинхронной приемопередачи, который

поддерживает те же режимы асинхронного обмена, что и UART, принято называть SCI (Serial Communication Interface). Следует отметить, что модуль типа SCI обычно реализует только режим асинхронного обмена, то есть его функциональные возможности уже по сравнению с модулями типа UART. Однако бывают и исключения: под тем же именем SCI в МК MC68HC705B16 скрывается модуль синхронно-асинхронной передачи данных.

Модули типа UART в асинхронном режиме работы позволяют реализовать протокол обмена для интерфейсов RS-232C, RS-422A, RS-485, в синхронном режиме — нестандартные синхронные протоколы обмена, и в некоторых моделях — SPI. В МК фирмы Motorola традиционно предусмотрены два модуля последовательного обмена: модуль SCI с возможностью реализации только протоколов асинхронной приемопередачи для интерфейсов RS-232C, RS-422A, RS-485 и модуль контроллера синхронного интерфейса в стандарте SPI.

Протоколы интерфейсов локальных сетей на основе МК (I^2C и CAN) отличает более сложная логика работы. Поэтому контроллеры CAN интерфейса всегда выполняются в виде самостоятельного модуля. Интерфейс I^2C с возможностью работы как в ведущем, так и ведомом режиме, также обычно поддерживается специальным модулем (модуль последовательного порта в МК 89C52 фирмы Philips). Но если реализуется только ведомый режим I^2C, то в МК PIC16 фирмы Microchip он успешно сочетается с SPI: настройка одного и того же модуля на один из протоколов осуществляется путем инициализации.

В последнее время появилось большое количество МК со встроенными модулями контроллеров CAN и модулями универсального последовательного интерфейса периферийных устройств USB (Universal Serial Bus). Каждый из этих интерфейсов имеет достаточно сложные протоколы обмена, для ознакомления с которыми следует обращаться к специальной литературе.

4.10.2. Модули аналогового ввода/вывода

Необходимость приема и формирования аналоговых сигналов требует наличия в МК модулей аналогового ввода/вывода.

Простейшим устройством аналогового ввода в МК является встроенный компаратор напряжения. Компаратор сравнивает входное аналоговое напряжение с опорным потенциалом V_{REF} и устанавливает на выходе логическую "1", если входное напряжение больше опорного. Компараторы удобнее всего использовать для контроля определенного значения входного напряжения, например, в термостатах. В комбинации с внешним генератором линейно изменяющегося напряжения встроенный компаратор позволяет реализовать на МК интегрирующий аналого-цифровой преобразователь (АЦП).

Однако более широкие возможности для работы с аналоговыми сигналами дает АЦП, встроенный в МК. Чаще всего он реализуется в виде модуля многоканального АЦП, предназначенного для ввода в МК аналоговых сигналов с датчиков физических величин и преобразования этих сигналов в двоичный код. Структурная схема типового модуля АЦП представлена на рис. 4.12.

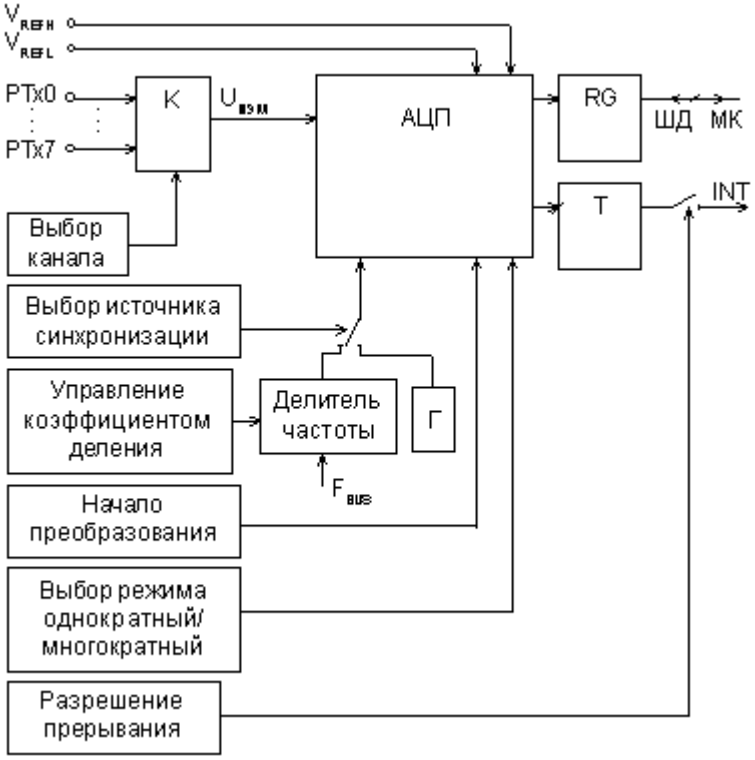

Рис. 4.12. Структура модуля АЦП.

Многоканальный аналоговый коммутатор К служит для подключения одного из источников аналоговых сигналов (PTx0...PTx7) ко входу АЦП. Выбор источника сигнала для преобразования осуществляется посредством записи номера канала коммутатора в соответствующие разряды регистра управления АЦП.

Два вывода модуля АЦП используются для задания опорного напряжения $U_{оп}$: V_{REFH} — верхний предел $U_{оп}$, V_{REFL} — нижний предел. Разность потенциалов на входах V_{REFH} и V_{REFL} и составляет $U_{оп}$. Разрешающая способность АЦП составляет $U_{оп}/2^n$, где n — число двоичных разрядов в слове результата. Максимальное значение опорного напряжения, как правило, равно напряжению питания МК. Если измеряемое напряжение $U_{изм} > V_{REFH}$, то результат преобразования будет равен FF, код 00 соответствует напряжениям $U_{изм}$ J V_{REFL}. Для достижения максимальной точности измерения следует выбрать максимально допустимое значение $U_{оп}$. В этом случае напряжение смещения нуля входного буфера и нелинейность передаточной характеристики АЦП будут вносить относительно малые погрешности.

Собственно аналого-цифровой преобразователь выполнен по методу последовательного приближения. Практически во всех моделях 8-разрядных МК разрядность АЦП также составляет 8 разрядов. Соответственно, формат представления результатов измерения АЦП — однобайтовый. Исключение составляют лишь модули АЦП микроконтроллеров для управления преобразователями частоты для электроприводов, разрешающая способность которых равна 10 разрядам. Два младших разряда результата получают с помощью дополнительного емкостного делителя, не связанного с регистром последовательного приближения.

Длительность такта преобразования задает генератор синхронизации: один цикл равен двум периодам частоты генератора t_{ADC}. Время преобразования для типовых модулей АЦП микроконтроллеров составляет от единиц до десятков микросекунд.

Источником синхронизации модуля АЦП может служить встроенный RC-генератор (Г) или импульсная последовательность тактирования межмодульных магистралей МК. В первом случае частота синхронизации АЦП обязательно окажется оптимальной, то есть той, которая рекомендуется в техническом описании. Во втором случае выбранная по другим соображениям f_{BUS} может оказаться

неподходящей для модуля АЦП. На этот случай в составе некоторых модулей предусмотрен программируемый делитель частоты f_{BUS}.

Момент завершения каждого цикла преобразования отмечается установкой триггера готовности данных. Если прерывания от модуля АЦП разрешены, то генерируется запрос на прерывания. Как правило, чтение регистра результата сбрасывает триггер готовности.

Большинство модулей АЦП имеют только режим программного запуска: установка одного из битов регистра режима запускает очередное измерение. Наиболее универсальные модули АЦП имеют также режим автоматического запуска, при котором после завершения одного цикла преобразования немедленно начинается следующий. Однако данные измерения каждого цикла должны быть считаны программным способом.

Цифро-аналоговые преобразователи в составе МК являются большой редкостью. Функция цифро-аналогового преобразователя реализуется средствами модуля программируемого таймера в режиме ШИМ. На одном из выводов МК формируется высокочастотная импульсная последовательность с регулируемой длительностью импульса. Полученный сигнал сглаживается фильтром нижних частот на операционном усилителе. Разрешающая способность такого ЦАП определяется дискретностью регулирования коэффициента заполнения в режиме ШИМ.

Глава 5. Однокристальные микроконтроллеры серии PIC

9. Лекция: Аппаратные средства микроконтроллеров серии PIC

В этой лекции рассказывается об особенностях организации и параметрах популярных PIC-микроконтроллеров фирмы "Microchip", а также более подробно о составе, структуре и возможностях аппаратных средств микроконтроллеров подгруппы PIC16F8X.

5.1. Основные особенности микроконтроллеров серии PIC

5.1.1. Состав и назначение семейств PIC-контроллеров

Микроконтроллеры семейств PIC (Peripheral Interface Controller) компании Microchip объединяют все передовые технологии микроконтроллеров: электрически программируемые пользователем ППЗУ, минимальное энергопотребление, высокую производительность, хорошо развитую RISC-архитектуру, функциональную законченность и минимальные размеры. Широкая номенклатура изделий обеспечивает использование микроконтроллеров в устройствах, предназначенных для разнообразных сфер применения.

Первые микроконтроллеры компании Microchip PIC16C5x появились в конце 1980-х годов и благодаря своей высокой производительности и низкой стоимости составили серьезную конкуренцию производившимся в то время 8-разрядным МК с CISC-архитектурой.

Высокая скорость выполнения команд в PIC-контроллерах достигается за счет использования двухшинной гарвардской архитектуры вместо традиционной одношинной фон-неймановской. Гарвардская архитектура основывается на наборе регистров с разделенными шинами и адресными пространствами для команд и данных. Все ресурсы микроконтроллера, такие как порты ввода/вывода, ячейки памяти и таймер, представляют собой физически реализованные аппаратные регистры.

Микроконтроллеры PIC содержат RISC-процессор с симметричной системой команд, позволяющей выполнять операции с любым регистром, используя произвольный метод адресации. Пользователь может сохранять результат операции в самом регистре-аккумуляторе или во втором регистре, используемом для операции.

В настоящее время компания Microchip выпускает пять основных семейств 8-разрядных RISC-микроконтроллеров, совместимых снизу вверх по программному коду:

- PIC12CXXX – семейство микроконтроллеров, выпускаемых в миниатюрном 8-выводном исполнении. Эти микроконтроллеры выпускаются как с 12-разрядной (33 команды), так и с 14-разрядной (35 команд) системой команд. Содержат встроенный тактовый генератор, таймер/счетчик, сторожевой таймер, схему управления прерываниями. В составе семейства есть микроконтроллеры со встроенным 8-разрядным четырехканальным АЦП. Способны работать при напряжении питания до 2,5 В;
- PIC16C5X – базовое семейство микроконтроллеров с 12-разрядными командами (33 команды), выпускаемое в 18-, 20- и 28-выводных корпусах. Представляют собой простые недорогие микроконтроллеры с минимальной периферией. Способность работать при малом напряжении питания (до 2 В) делает их удобными для применения в переносных конструкциях. В состав семейства входят микроконтроллеры подгруппы PIC16HV5XX, способные работать непосредственно от батареи в диапазоне питающих напряжений до 15 В;
- PIC16CXXX – семейство микроконтроллеров среднего уровня с 14-разрядными командами (35 команд). Наиболее многочисленное семейство, объединяющее микроконтроллеры с разнообразными периферийными устройствами, в число которых входят аналоговые компараторы, аналого-цифровые преобразователи, контроллеры последовательных интерфейсов SPI, USART и I2C, таймеры-счетчики, модули захвата/сравнения, широтно-импульсные модуляторы, сторожевые таймеры, супервизорные схемы и так далее;
- PIC17CXXX – семейство высокопроизводительных микроконтроллеров с расширенной системой команд 16-разрядного формата (58 команд), работающие на частоте до 33 МГц, с объемом памяти программ до 16 Кслов. Кроме обширной периферии, 16-уровневого аппаратного стека и векторной системы прерываний, почти все микроконтроллеры этого семейства имеют встроенный аппаратный умножитель 8x8, выполняющий операцию умножения за один машинный цикл. Являются одними из самых быстродействующих в классе 8-разрядных микроконтроллеров;
- PIC18CXXX – семейство высокопроизводительных микроконтроллеров с расширенной системой команд 16-разрядного формата (75 команд) и встроенным 10-разрядным АЦП, работающие на частоте до 40 МГц. Содержат 31-уровневый аппаратный стек, встроенную память команд до 32 Кслов и способны адресовать до 4 Кбайт памяти данных и до 2 Мбайт внешней памяти программ. Расширенное RISC-ядро микроконтроллеров данного семейства оптимизировано под использование нового Си-компилятора.

Большинство PIC-контроллеров выпускаются с однократно программируемой памятью программ (OTP), с возможностью внутрисхемного программирования или масочным ПЗУ. Для целей отладки предлагаются более дорогие версии с ультрафиолетовым стиранием и Flash-памятью. Полный список выпускаемых модификаций PIC-контроллеров включает порядка пятисот наименований. Поэтому продукция компании перекрывает почти весь диапазон применений 8-разрядных микроконтроллеров.

Из программных средств отладки наиболее известны и доступны различные версии ассемблеров, а также интегрированная программная среда MPLAB. Российские производители программаторов и аппаратных отладочных средств также уделяют внимание PIC-контроллерам. Выпускаются как специализированные программаторы, такие как PICPROG, программирующие почти весь спектр PIC-микроконтроллеров, так и универсальные: UNIPRO и СТЕРХ, поддерживающие наиболее известные версии PIC-контроллеров.

Наиболее распространенными семействами PIC-контроллеров являются PIC16CXXX и PIC17CXXX.

5.1.2. Микроконтроллеры семейств PIC16CXXX и PIC17CXXX

Основным назначением микроконтроллеров семейств PIC16 и PIC17, как следует из аббревиатуры PIC (Peripheral Interface Controller), является выполнение интерфейсных функций. Этим объясняются особенности их архитектуры:

- RISC-система команд, характеризующаяся малым набором одноадресных инструкций (33, 35 или 58), каждая из которых имеет длину в одно слово (12, 14 или 16 бит) и большинство выполняется за один

машинный цикл. В системе команд отсутствуют сложные арифметические команды (умножение, деление), предельно сокращен набор условных переходов;
- высокая скорость выполнения команд: при тактовой частоте 20 МГц время машинного цикла составляет 200 нс (быстродействие равно 5 млн. операций/сек);
- наличие мощных драйверов (до 25 мА) на линиях портов ввода/вывода, что позволяет подключать непосредственно к ним довольно мощную нагрузку, например, светодиоды.
- низкая потребляемая мощность;
- ориентация на ценовую нишу предельно низкой стоимости, определяющая использование дешевых корпусов с малым количеством выводов (8, 14, 18, 28), отказ от внешних шин адреса и данных (кроме PIC17C4X), использование упрощенного механизма прерываний и аппаратного (программно недоступного) стека.

5.1.3. Особенности архитектуры микроконтроллеров семейства PIC16CXXX

Микроконтроллеры семейства PIC16CXXX, выполненные по технологии HCMOS представляют собой 8-разрядные микроконтроллеры на основе RISC-процессора, выполненные по гарвардской архитектуре. Имеют встроенное ПЗУ команд объемом от 0,5 до 4 Кслов (разрядность слова команд равна 12 – 14 бит). Память данных PIC-контроллеров организована в виде регистрового файла объемом 32 – 128 байт, в котором от 7 до 16 регистров отведено для управления системой и обмена данными с внешними устройствами.

Одним из основных достоинств этих устройств является очень широкий диапазон напряжений питания (2 – 6 В). Ток потребления на частоте 32768 Гц составляет менее 15 мкА, на частоте 4 МГц – 1 – 2 мА, на частоте 20 МГц 5 – 7 мА и в режиме микропотребления (режим SLEEP) – 1 – 2 мкА. Выпускаются модификации для работы в трех температурных диапазонах: от 0 до +70°С, от -40 до +85°С и от -40 до +125°С.

Каждый из контроллеров содержит универсальные (от 1 до 3) и сторожевой таймеры, а также надежную встроенную систему сброса при включении питания. Частота внутреннего тактового генератора задается либо кварцевым резонатором, либо RC-цепочкой в диапазоне 0 – 25 МГц. PIC-контроллеры имеют от 12 до 33 линий цифрового ввода-вывода, причем каждая из них может быть независимо настроена на ввод или вывод.

В устройство PIC16C64 входит широтно-импульсный модулятор, с помощью которого можно реализовать ЦАП с разрешением до 16 разрядов. Здесь есть и последовательный двунаправленный синхронно-асинхронный порт, обеспечивающий возможность организации шины I^2C. Приборы PIC16C71 и PIC16C74 содержат встроенный многоканальный 8-разрядный АЦП с устройством выборки-хранения.

Помимо памяти программ в PIC предусмотрено несколько индивидуально прожигаемых перемычек, с помощью которых можно на этапе программирования кристалла выбрать тип тактового генератора, отключить сторожевой таймер или систему сброса, включить защиту памяти программ от копирования, а также записать серийный номер кристалла (16 бит).

С программной точки зрения PIC-контроллер представляет собой 8-разрядный RISC-процессор с гарвардской архитектурой. Число команд небольшое — от 33 до 35. Все команды имеют одинаковую длину и, кроме команд ветвления, выполняются за четыре периода тактовой частоты (в отличие, например, от 12 периодов для i87C51). Поддерживаются непосредственный, косвенный и относительный методы адресации, можно эффективно управлять отдельными битами в пределах всего регистрового файла. Стек реализован аппаратно. Его максимальная глубина составляет два или восемь уровней в зависимости от типа контроллера. Почти во всех микросхемах PIC есть система прерываний, источниками которых могут быть таймер и внешние сигналы. Система команд практически симметрична и, как следствие, легка в освоении.

Применение PIC-контроллеров целесообразно в несложных приборах с ограниченным током потребления (автономные устройства, приборы с питанием от телефонной линии и т.п.). Благодаря малому

количеству компонентов, используемых при построении таких приборов, их размеры уменьшаются, а надежность увеличивается.

Типичным представителем микроконтроллеров семейства PIC16CXXX являются микроконтроллеры подгруппы PIC16F8X.

5.2. Микроконтроллеры подгруппы PIC16F8X

5.2.1. Основные характеристики

Микроконтроллеры подгруппы PIC16F8X относятся к семейству 8-разрядных КМОП микроконтроллеров группы PIC16CXXX, для которых характерны низкая стоимость, полностью статическая КМОП-технология и высокая производительность.

В состав подгруппы входят МК PIC16F83, PIC16CR83, PIC16F84 и PIC16CR84. Основные характеристики МК подгруппы PIC16F8X приведены в табл. 5.1.

Все микроконтроллеры подгруппы PIC16F8X используют гарвардскую архитектуру с RISC-процессором, обладающую следующими основными особенностями:

- используются только 35 простых команд;
- все команды выполняются за один цикл (400 нс при частоте 10 МГц), кроме команд перехода, которые требуют 2 цикла;
- рабочая частота 0 Гц ... 10 МГц;
- раздельные шины данных (8 бит) и команд (14 бит);
- 512 x 14 или 1024 x 14 память программ, выполненная на ПЗУ или электрически перепрограммируемой Flash- памяти;
- 15 восьмиразрядных регистров специальных функций (SFR);
- восьмиуровневый аппаратный стек;
- прямая, косвенная и относительная адресация данных и команд;
- 36 или 68 восьмиразрядных регистров общего назначения (GPR) или ОЗУ;
- четыре источника прерывания:
  - внешний вход RB0/INT;
  - переполнение таймера TMR0;
  - изменение сигналов на линиях порта B;
  - завершение записи данных в память EEPROM;
- 64 x 8 электрически перепрограммируемая EEPROM память данных с возможностью выполнения 1000000 циклов стирания/записи;
- сохранение данных в EEPROM в течение как минимум 40 лет.

Таблица 5.1. Основные характеристики МК подгруппы PIC16F8X.

Параметр	PIC16F83	PIC16CR83	PIC16F84	PIC16CR84
Максимальная частота, МГц	10	10	10	10
Flash-память программ, слов	512	-	1K	-
ПЗУ программ, слов	-	512	-	1K
Память данных, байт	36	36	68	68
Память данных в РПЗУ (EEPROM), байт	64	64	64	64
Таймеры	TMR0	TMR0	TMR0	TMR0

Число источников прерываний	4	4	4	4
Число линий ввода/вывода	13	13	13	13
Диапазон напряжений питания, В	2,0 – 6,0	2,0 – 6,0	2,0 – 6,0	2,0 – 6,0
Число выводов и тип корпуса	18 DIP, SOIC	18 DIP, SOIC	18 DIP, SOIC	18 DIP, SOIC

Микроконтроллеры подгруппы PIC16F8X обладают развитыми возможностями ввода/вывода:

- 13 линий ввода-вывода с индивидуальной установкой направления обмена;
- высокий втекающий/вытекающий ток, достаточный для управления светодиодами:
- максимальный втекающий ток – 25 мА;
- максимальный вытекающий ток – 20 мА;
- 8-битный таймер/счетчик TMR0 с 8-битным программируемым предварительным делителем.

Специализированные микроконтроллерные функции включают следующие возможности:

- автоматический сброс при включении (Power-on-Reset);
- таймер включения при сбросе (Power-up Timer);
- таймер запуска генератора (Oscillator Start-up Timer);
- сторожевой (Watchdog) таймер WDT с собственным встроенным генератором, обеспечивающим повышенную надежность;
- EEPROM бит секретности для защиты кода;
- экономичный режим SLEEP;
- выбираемые пользователем биты для установки режима возбуждения встроенного генератора;
- последовательное встроенное устройство программирования Flash/EEPROM памяти программ и данных с использованием только двух выводов.

КМОП технология обеспечивает МК подгруппы PIC16F8X дополнительные преимущества:

- статический принцип работы;
- широкий диапазон напряжений питания: 2,0 ... 6,0 В;
- низкое энергопотребление:
- менее 2 мА при 5В и 4МГц;
- порядка 15 мкА при 2В и 32КГц;
- менее 1 мкА для SLEEP-режима при 2В.

Микроконтроллеры подгруппы PIC16F8X различаются между собой только объемом ОЗУ данных, а также объемом и типом памяти программ. Наличие в составе подгруппы МК с Flash-памятью программ облегчает создание и отладку прототипов промышленных образцов изделий.

5.2.2. Особенности архитектуры

Упрощенная структурная схема МК подгруппы PIC16F8X приведена на рис. 5.1.

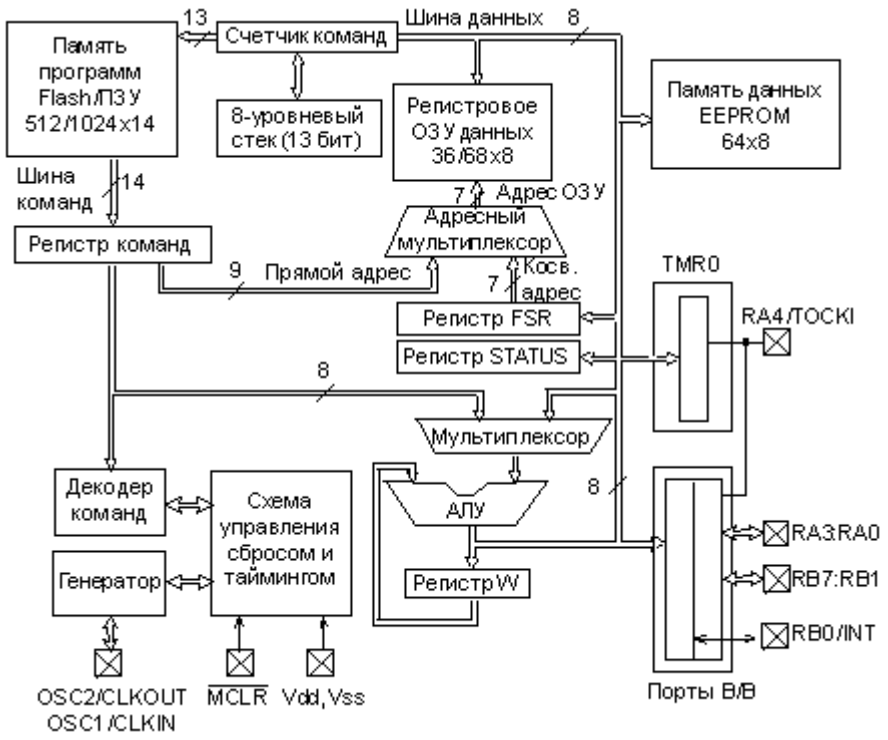

Рис. 5.1. Структурная схема МК подгруппы PIC16F8X.

Архитектура основана на концепции раздельных шин и областей памяти для данных и для команд (гарвардская архитектура). Шина данных и память данных (ОЗУ) – имеют ширину 8 бит, а программная шина и программная память (ПЗУ) имеют ширину 14 бит. Такая концепция обеспечивает простую, но мощную систему команд, разработанную так, что битовые, байтовые и регистровые операции работают с высокой скоростью и с перекрытием по времени выборок команд и циклов выполнения. 14-битовая ширина программной памяти обеспечивает выборку 14-битовой команды в один цикл. Двухступенчатый конвейер обеспечивает одновременную выборку и исполнение команды. Все команды выполняются за один цикл, исключая команды переходов.

Микроконтроллеры PIC16F83 и PIC16CR83 адресуют 512x14 памяти программ, а PIC16F84 и PIC16CR84 – 1Kx14 памяти программ. Вся память программ является внутренней.

Микроконтроллер может прямо или косвенно обращаться к регистрам или памяти данных. Все регистры специальных функций, включая счетчик команд, отображаются на память данных. Ортогональная (симметричная) система команд позволяет выполнять любую команду над любым регистром с использованием произвольного метода адресации. Ортогональная архитектура и отсутствие специальных исключений делает программирование МК группы PIC16F8X простым и эффективным.

Назначение выводов МК подгруппы PIC16F8X приведено в табл. 5.2.

Таблица 5.2. Назначение выводов МК подгруппы PIC16F8X.

Обозначение	Тип	Буфер	Описание
OSC1/CLKIN	I	ТШ/КМОП[1)]	Вход кристалла генератора, RC-цепочки или вход внешнего тактового сигнала
OSC2/CLKOUT	O	-	Выход кристалла генератора. В RC-режиме – выход 1/4 частоты OSC1
/MCLR	I/P	ТШ	Сигнал сброса/вход программирующего напряжения. Сброс низким уровнем.

RA0	I/O	ТТЛ	PORTA – двунаправленный порт ввода/вывода RA4/T0CKI может быть выбран как тактовый вход таймера/счетчика TMR0. Выход с открытым стоком.
RA1	I/O	ТТЛ	
RA2	I/O	ТТЛ	
RA3	I/O	ТТЛ	
RA4	I/O	ТШ	
/T0CKI			
RB0/INT	I/O	ТТЛ/ТШ[2)]	PORTB – двунаправленный порт ввода/вывода. Может быть запрограммирован в режиме внутренних активных нагрузок на линию питания по всем выводам. Вывод RB0/INT может быть выбран как внешний вход прерывания. Выводы RB4...RB7 могут быть программно настроены как входы прерывания по изменению состояния на любом из входов. При программировании МК RB6 используется как тактовый, а RB7 как вход/выход данных.
RB1	I/O	ТТЛ	
RB2	I/O	ТТЛ	
RB3	I/O	ТТЛ	
RB4	I/O	ТТЛ	
RB5	I/O	ТТЛ	
RB6	I/O	ТТЛ/ТШ[3)]	
RB7	I/O	ТТЛ/ТШ[4)]	

Таблица 5.2. Назначение выводов МК подгруппы PIC16F8X (продолжение).

Обозначение	Тип	Буфер	Описание
Vdd	P	-	Положительное напряжение питания
Vss	P	-	Общий провод (земля)

В таблице использованы следующие обозначения: I — вход; O — выход; I/O — вход/выход; P — питание; – — не используется; ТТЛ — ТТЛ вход; ТШ — вход триггера Шмитта.

Микроконтроллер содержит 8-разрядное АЛУ и рабочий регистр W. АЛУ является арифметическим модулем общего назначения и выполняет арифметические и логические функции над содержимым рабочего регистра и любого из регистров контроллера. АЛУ может выполнять операции сложения, вычитания, сдвига и логические операции. Если не указано иное, то арифметические операции выполняются в дополнительном двоичном коде.

В зависимости от результата операции, АЛУ может изменять значения бит регистра STATUS: C (Carry), DC (Digit carry) и Z (Zero).

5.2.3. Схема тактирования и цикл выполнения команды

Входная тактовая частота, поступающая с вывода OSC1/CLKIN, делится внутри на четыре, и из нее формируются четыре циклические не перекрывающиеся тактовые последовательности Q1, Q2, Q3 и Q4. Счетчик команд увеличивается в такте Q1, команда считывается из памяти программы и защел-

кивается в регистре команд в такте Q4. Команда декодируется и выполняется в течение последующего цикла в тактах Q1...Q4. Схема тактирования и выполнения команды изображена на рис. 5.2.

Цикл выполнения команды состоит из четырех тактов: Q1...Q4. Выборка команды и ее выполнение совмещены по времени таким образом, что выборка команды занимает один цикл, а выполнение – следующий цикл. Эффективное время выполнения команды составляет один цикл. Если команда изменяет счетчик команд (например, команда GOTO), то для ее выполнения потребуется два цикла, как показано на рис. 5.3.

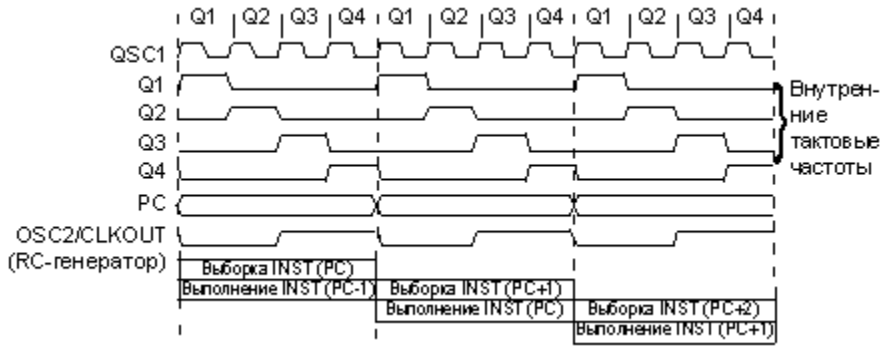

Рис. 5.2. Схема тактирования и выполнения команды.

Рис. 5.3. Выборка команд.

Цикл выборки начинается с увеличения счетчика команд в такте Q1. В цикле выполнения команды выбранная команда защелкивается в регистр команд в такте Q1. В течение тактов Q2, Q3 и Q4 происходит декодирование и выполнение команды. В такте Q2 считывается память данных (чтение операнда), а запись происходит в такте Q4.

5.2.4. Организация памяти программ и стека

Счетчик команд в МК PIC16F8X имеет ширину 13 бит и способен адресовать 8Кх14бит объема программной памяти. Однако физически на кристаллах PIC16F83 и PIC16CR83 имеется только 512x14 памяти (адреса 0000h-01FFh), а в МК PIC16F84 и PIC16CR84 – 1Кх14 памяти (адреса 0000h-03FFh). Обращение к адресам выше 1FFh (3FFh) фактически есть адресация в те же первые 512 адресов (первые 1К адресов).

Организация памяти программ и стека приведена на рис. 5.4.

В памяти программ есть выделенные адреса. Вектор сброса находится по адресу 0000h, вектор прерывания – по адресу 0004h. Обычно по адресу 0004h располагается подпрограмма идентификации и обработки прерываний, а по адресу 0000h – команда перехода на метку, расположенную за подпрограммой обработки прерываний.

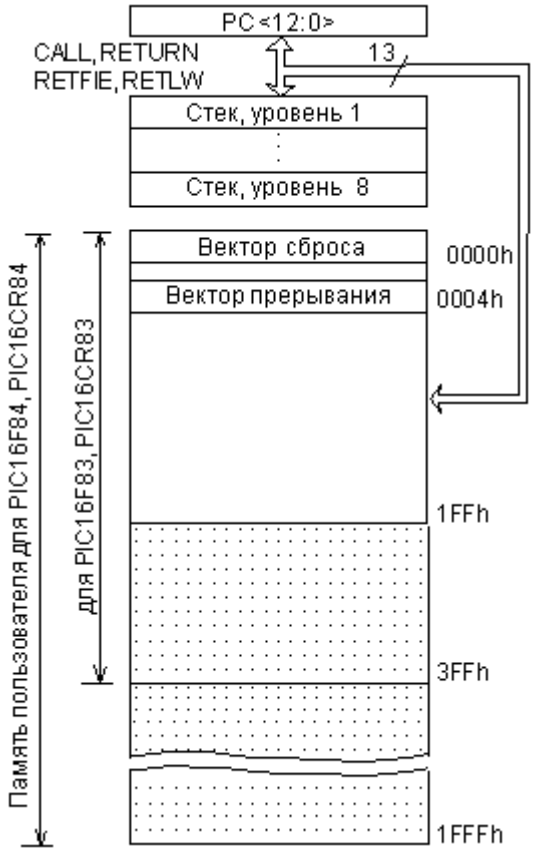

Рис. 5.4. Организация памяти программ и стека.

5.2.5. Организация памяти данных

Память данных МК разбита на две области. Первые 12 адресов – это область регистров специальных функций (SFR), а вторая – область регистров общего назначения (GPR). Область SFR управляет работой прибора.

Обе области разбиты в свою очередь на банки 0 и 1. Банк 0 выбирается обнулением бита RP0 регистра статуса (STATUS). Установка бита RP0 в единицу выбирает банк 1. Каждый банк имеет протяженность 128 байт. Однако для PIC16F83 и PIC16CR83 память данных существует только до адреса 02Fh, а для PIC16F84 и PIC16CR84 – до адреса 04Fh.

На рис. 5.5 изображена организация памяти данных.

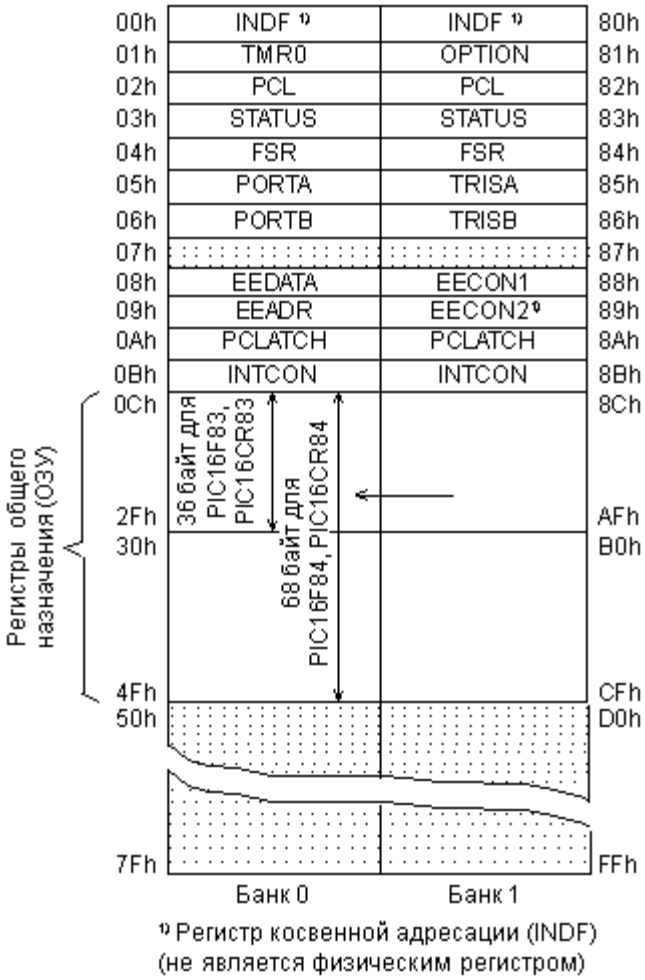

Рис. 5.5. Организация памяти данных.

Некоторые регистры специального назначения продублированы в обоих банках, а некоторые расположены в банке 1 отдельно.

Регистры с адресами 0Ch-4Fh могут использоваться как регистры общего назначения, которые представляют собой статическое ОЗУ. Адреса регистров общего назначения банка 1 отображаются на банк 0. Следовательно, когда установлен банк 1, то обращение к адресам 8Ch-CFh фактически адресует банк 0.

В регистре статуса помимо бита RP0 есть еще бит RB1, что позволяет обращаться к четырем страницам (банкам) будущих модификаций этого кристалла.

К ячейкам ОЗУ можно адресоваться прямо, используя абсолютный адрес каждого регистра, или косвенно, через регистр указатель FSR. Косвенная адресация использует текущее значение разрядов RP1:RP0 для доступа к банкам. Это относится и к EEPROM памяти данных. В обоих случаях можно адресовать до 512 регистров.

5.2.6. Регистры специального назначения

Регистр статуса (STATUS) содержит признаки операции (арифметические флаги) АЛУ, состояние контроллера при сбросе и биты выбора страниц для памяти данных. Назначение бит регистра приведено в табл. 5.3.

Таблица 5.3. Назначение бит регистра STATUS (адрес 03h, 83h).

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	/TO	/PD	Z	DC	C
Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0

Бит 7: IRP: бит выбора страницы банка данных (используется при косвенной адресации)

0 = банк 0,1 (00h – FFh)

1 = банк 2,3 (100h – 1FFh)

Бит IRP не используется в МК подгруппы PIC16F8X

Биты 6-5: RP1:RP0: биты выбора страницы банка данных (используются при прямой адресации)

00 = банк 0 (00h – 7Fh)

01 = банк 1 (80h – FFh)

10 = банк 2 (100h – 17Fh)

11 = банк 3 (180h – 1FFh)

В МК подгруппы PIC16F8X используется только бит RP0

Бит 4: /TO: бит срабатывания сторожевого таймера

1 = после включения питания, а также командами CLRWDT и SLEEP

0 = по завершении выдержки сторожевого таймера

Бит 3: /PD: бит снижения потребляемой мощности

1 = после включения питания, а также командой CLRWDT

0 = по команде SLEEP

Бит 2: Z: бит нулевого результата

1 = результат арифметической или логической операции нулевой

0 = результат арифметической или логической операции ненулевой

Бит 1: DC: бит десятичного переноса/заема (для команд ADDWF и ADDLW)

1 = имеет место перенос из 4-го разряда

0 = нет переноса из 4-го разряда

Бит 0: C: бит переноса/заема (для команд ADDWF и ADDLW)

1 = имеет место перенос из самого старшего разряда

0 = нет переноса из самого старшего разряда

Примечание: вычитание осуществляется путем прибавления дополнительного кода второго операнда. При выполнении команд сдвига этот бит загружается из младшего или старшего разряда сдвигаемого источника.

Здесь и далее: R — читаемый бит; W — записываемый бит; S — устанавливаемый бит; U — неиспользуемый бит (читается как "0"); -n = 0 или 1 — значение бита после сброса.

Регистр статуса доступен для любой команды так же, как любой другой регистр. Однако если регистр STATUS является регистром назначения для команды, влияющей на биты Z, DC или C, то запись в эти три бита запрещается. Кроме того, биты /TO и /PD устанавливаются аппаратно и не могут быть записаны в статус программно. Это следует иметь в виду при выполнении команды с использованием регистра статуса. Например, команда CLRF STATUS обнулит все биты, кроме битов /TO и /PD, а затем установит бит Z=1. После выполнения этой команды регистр статуса может и не иметь нулевого значения (из-за битов /TO и /PD) STATUS=000uu1uu, где u – неизменяемое состояние. Поэтому рекомендуется для изменения регистра статуса использовать только команды битовой установки BCF, BSF, MOVWF, которые не изменяют остальные биты статуса. Воздействие всех команд на биты статуса рассматривается в разделе "Описание системы команд".

Регистр конфигурации (OPTION) является доступным по чтению и записи регистром, который содержит управляющие биты для конфигурации предварительного делителя (пределителя), внешних прерываний, таймера, а также резисторов "pull-up" на выводах PORTB. Назначение бит регистра приведено в табл. 5.4.

Таблица 5.4. Назначение бит регистра OPTION (адрес 81h).

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
/RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0

Бит 7: /RBPU: бит установки резисторов "pull-up" на выводах PORTB

0 = резисторы "pull-up" подключены

1 = резисторы "pull-up" отключены

Бит 6: INTEDG: бит выбора перехода сигнала прерывания

0 = прерывание по спаду сигнала на выводе RB0/INT

1 = прерывание по фронту сигнала на выводе RB0/INT

Бит 5: T0CS: бит выбора источника сигнала таймера TMR0

0 = внутренний тактовый сигнал (CLKOUT)

1 = переход на выводе RA4/T0CKI

Бит 4: T0SE: бит выбора перехода источника сигнала для TMR0

0 = приращение по фронту сигнала на выводе RA4/T0CKI

1 = приращение по спаду сигнала на выводе RA4/T0CKI

Бит 3: PSA: бит назначения пределителя

0 = предделитель подключен к TMR0

1 = предделитель подключен к сторожевому таймеру WDT

Биты 2-0: PS2:PS0: биты выбора коэффициента деления пределителя

Значения бит	Скорость TMR0	Скорость WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

В том случае, когда предделитель обслуживает сторожевой таймер WDT, таймеру TMR0 назначается коэффициент предварительного деления 1:1.

Регистр условий прерывания (INTCON) является доступным по чтению и записи регистром, который содержит биты доступа для всех источников прерываний. Назначение бит регистра приведено в табл. 5.5.

Таблица 5.5. Назначение бит регистра INTCON (адреса 0Bh, 8Bh).

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0

Бит 7: GIE: бит разрешения всех прерываний

0 = запрещены все прерывания

1 = разрешены все незамаскированные прерывания

Бит 6: EEIE: бит разрешения прерывания записи в EEPROM

0 = запрещены прерывания записи в EEPROM

1 = разрешены прерывания записи в EEPROM

Бит 5: T0IE: бит разрешения прерывания по переполнению TMR0

0 = запрещены прерывания от TMR0

1 = разрешены прерывания от TMR0

Бит 4: INTE: бит разрешения прерываний по входу RB0/INT

0 = запрещены прерывания по входу RB0/INT

1 = разрешены прерывания по входу RB0/INT
Бит 3: RBIE: бит разрешения прерываний по изменению PORTB 0 = запрещены прерывания по изменению PORTB 1 = разрешены прерывания по изменению PORTB
Бит 2: T0IF: бит запроса прерывания по переполнению TMR0 0 = прерывание по переполнению TMR0 отсутствует 1 = прерывание по переполнению TMR0 имеет место
Бит 1: INTF: бит запроса прерывания по входу RB0/INT 0 = прерывание по входу RB0/INT отсутствует 1 = прерывание по входу RB0/INT имеет место
Бит 0: RBTF: бит запроса прерывания по изменению PORTB 0 = ни на одном из входов RB7:RB4 состояние не изменилось 1 = хотя бы на одном из входов RB7:RB4 изменилось состояние

Бит разрешения всех прерываний GIE сбрасывается автоматически при следующих обстоятельствах:

- по включению питания;
- по внешнему сигналу /MCLR при нормальной работе;
- по внешнему сигналу /MCLR в режиме SLEEP;
- по окончанию задержки таймера WDT при нормальной работе;
- по окончанию задержки таймера WDT в режиме SLEEP.

Прерывание INT может вывести процессор из режима SLEEP, если перед входом в этот режим бит INTE был установлен в единицу. Состояние бита GIE также определяет: будет ли процессор переходить на подпрограмму прерывания после выхода из режима SLEEP.

Сброс битов – запросов прерываний – должен осуществляться соответствующей программой обработки.

5.2.7. Счетчик команд

Счетчик команд PCL и PCLATH имеет разрядность 13 бит. Младший байт счетчика (PCL) доступен для чтения и записи и находится в регистре 02h. Старший байт счетчика команд не может быть напрямую записан или считан и берется из регистра PCLATH (PC latch high), адрес которого 0Ah. Содержимое PCLATH передается в старший байт счетчика команд, когда он загружается новым значением.

В зависимости от того, загружается ли в счетчик команд новое значение во время выполнения команд CALL, GOTO, или в младший байт счетчика команд (PCL) производится запись, – старшие биты счетчика команд загружаются из PCLATH разными способами, как показано на рис. 5.6.

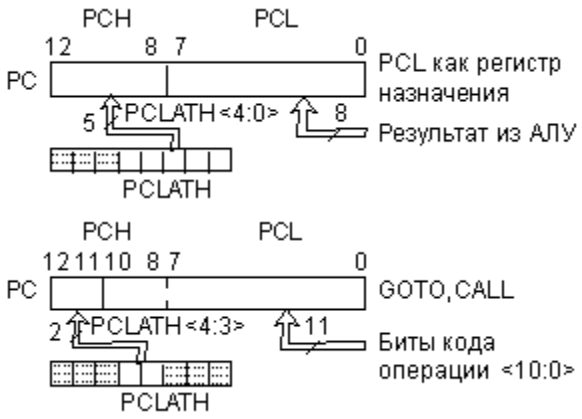

Рис. 5.6. Загрузка старших бит счетчика команд.

Команды CALL и GOTO оперируют 11-разрядным адресным диапазоном, достаточным для смещения в пределах страницы программной памяти объемом 2К слов. Для МК подгруппы PIC16F8X этого хватает. С целью обеспечения возможности расширения памяти команд для будущих моделей МК предусмотрена загрузка двух старших бит счетчика команд из регистра PCLATH<4:3>. При использовании команд CALL и GOTO пользователь должен убедиться в том, что эти страничные биты запрограммированы для выхода на нужную страницу. При выполнении команды CALL или выполнении прерывания весь 13-битный счетчик команд помещается в стек, поэтому для возвращения из подпрограммы не нужны манипуляции с разрядами PCLATH<4:3>.

Микроконтроллеры подгруппы PIC16F8X игнорируют значения бит PCLATH<4:3>, которые используются для обращения к страницам 1, 2 и 3 программной памяти. Однако применять биты PCLATH<4:3> в качестве ячеек памяти общего назначения не рекомендуется, так как это может повлиять на совместимость с будущими поколениями изделий.

Возможность выполнять арифметические операции непосредственно над счетчиком команд позволяет очень быстро и эффективно осуществлять табличные преобразования в PIC-контроллерах.

Микроконтроллеры подгруппы PIC16F8X имеют восьмиуровневый аппаратный стек шириной 13 бит (см. рис. 5.4). Область стека не принадлежит ни к программной области, ни к области данных, а указатель стека пользователю недоступен. Текущее значение счетчика команд посылается в стек, когда выполняется команда CALL или производится обработка прерывания. При выполнении процедуры возврата из подпрограммы (команды RETLW, RETFIE или RETURN) содержимое счетчика команд восстанавливается из стека. Регистр PCLATH при операциях со стеком не изменяется.

Стек работает как циклический буфер. Следовательно, после того как стек был загружен 8 раз, девятая загрузка перепишет значение первой. Десятая загрузка перепишет вторую и т.д. Если стек был выгружен 9 раз, счетчик команд становится таким же, как после первой выгрузки.

Признаков положения стека в контроллере не предусмотрено, поэтому пользователь должен самостоятельно следить за уровнем вложения подпрограмм.

5.2.8. Прямая и косвенная адресации

Когда производится прямая 9-битная адресация, младшие 7 бит берутся как прямой адрес из кода операции, а два бита указателя страниц (RP1, RP0) из регистра статуса, как показано на рис. 5.7.

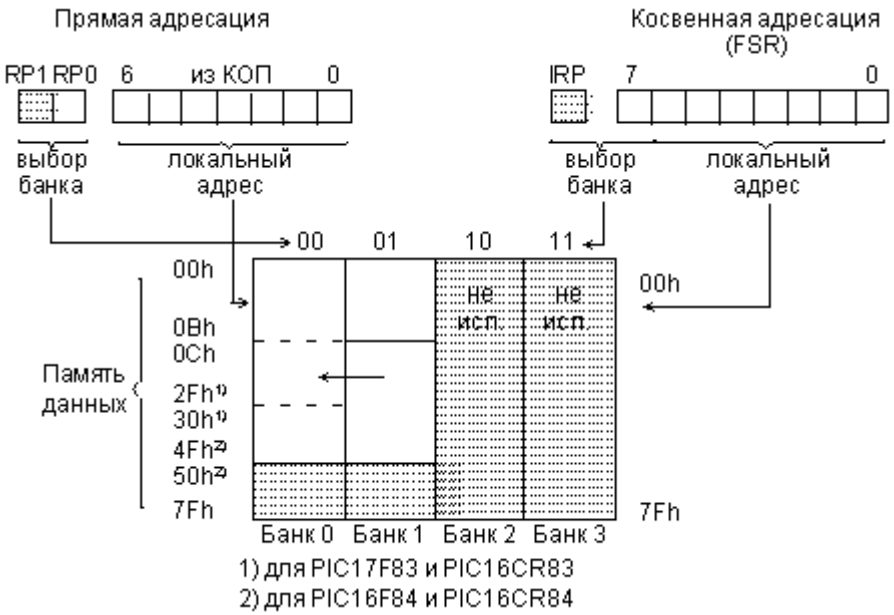

Рис. 5.7. Методы адресации данных.

Признаком косвенной адресации является обращение к регистру INDF. Любая команда, которая использует INDF (адрес 00h) в качестве регистра фактически обращается к указателю, который хранится в FSR (адрес 04h). Чтение косвенным образом самого регистра INDF даст результат 00h. Запись в регистр INDF косвенным образом будет выглядеть как NOP, но биты статуса могут быть изменены. Необходимый 9-битный адрес формируется объединением содержимого 8-битного FSR регистра и бита IRP из регистра статуса (см. рис. 5.7).

Обратите внимание, что некоторые регистры специальных функций располагаются в банке 1. Чтобы адресоваться к ним, нужно дополнительно установить в единицу бит RP0 в регистре статуса.

5.2.9. Порты ввода/вывода

Контроллеры подгруппы PIC16F8X имеют два порта: PORTA (5 бит) и PORTB (8 бит) с побитовой индивидуальной настройкой на ввод или на вывод.

Порт A (PORTA) представляет собой 5-битовый фиксатор, соответствующий выводам контроллера RA<4:0>. Линия RA4 имеет вход триггера Шмитта и выход с открытым стоком. Все остальные линии порта имеют ТТЛ входные уровни и КМОП выходные буферы. Адрес регистра порта A – 05h.

Каждой линии порта поставлен в соответствие бит направления передачи данных, который хранится в управляющем регистре TRISA, расположенном по адресу 85h. Если бит управляющего TRISA регистра имеет значение 1, то соответствующая линия будет устанавливаться на ввод. Ноль переключает линию на вывод и одновременно выводит на нее содержимое соответствующего регистра-фиксатора порта. При включении питания все линии порта по умолчанию настроены на ввод.

На рис. 5.8 дана схема линий RA<3:0> порта A.

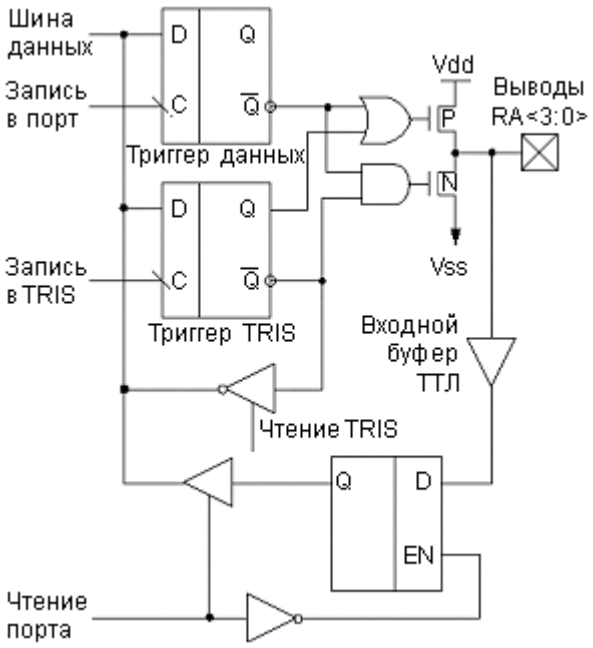

Рис. 5.8. Схема линий RA <3:0 порта A. Выводы порта имеют защитные диоды к Vdd и Vss

Операция чтения порта A считывает состояние выводов порта, в то время как запись в него изменяет состояние триггеров порта. Все операции с портом являются операциями типа "чтение-модификация-запись". Поэтому запись в порт предполагает, что состояние выводов порта вначале считывается, затем модифицируется и записывается в триггер-фиксатор. Вывод RA4 мультиплексирован с тактовым входом таймера TMR0. Схема линии RA4 порта A приведена на рис. 5.9.

Порт B (PORTB) – это двунаправленный 8-битовый порт, соответствующий выводам RB<7:0> контроллера и расположенный по адресу 06h. Относящийся к порту B управляющий регистр TRISB расположен на первой странице регистров по адресу 86h. Если бит управляющего TRISB регистра имеет значение 1, то соответствующая линия будет устанавливаться на ввод. Ноль переключает линию на вывод и одновременно выводит на нее содержимое соответствующего регистра защелки. При включении питания все линии порта по умолчанию настроены на ввод.

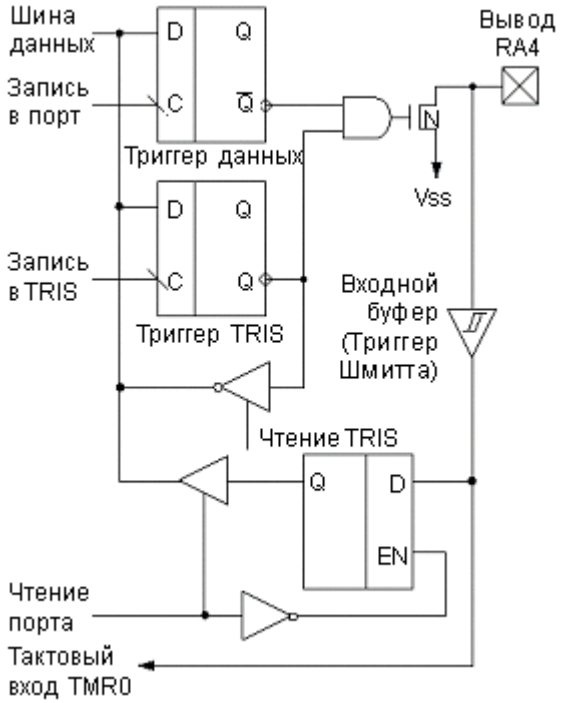

Рис. 5.9. Схема линии RA4 порта A. Вывод порта имеет защитный диод только к Vss.

У каждой ножки порта B имеется небольшая активная нагрузка (около 100мкА) на линию питания (pull-up). Она автоматически отключается, если эта ножка запрограммирована как вывод. Более того, управляющий бит /RBPU регистра OPTION<7> может отключить (при RBPU=1) все нагрузки. Сброс при включении питания также отключает все нагрузки.

Четыре линии порта B (RB<7:4>) могут вызвать прерывание при изменении значения сигнала на любой из них. Если эти линии настроены на ввод, то они опрашиваются и защелкиваются в цикле чтения Q1. Новая величина входного сигнала сравнивается со старой в каждом командном цикле. При несовпадении значения сигнала на ножке и в фиксаторе генерируется высокий уровень. Выходы детекторов "несовпадений" RB4, RB5, RB6, RB7 объединяются по ИЛИ и генерируют прерывание RBIF (запоминаемое в регистре INTCON<0>). Любая линия, настроенная как вывод, в этом сравнении не участвует. Прерывание может вывести кристалл из режима SLEEP. В подпрограмме обработки прерывания следует сбросить запрос прерывания одним из следующих способов:

- прочитать (или записать в) порт B. Это завершит состояние сравнения;
- обнулить бит RBIF регистра INTCON<0>.

При этом необходимо иметь в виду, что условие "несовпадения" будет продолжать устанавливать признак RBIF. Только чтение порта B может устранить "несовпадение" и позволит обнулить бит RBIF.

Прерывание по несовпадению и программно устанавливаемые внутренние активные нагрузки на этих четырех линиях могут обеспечить простой интерфейс, например, с клавиатурой, с выходом из режима SLEEP по нажатию клавиш.

Схемы линий порта B приведены на рис. 5.10 и 5.11.

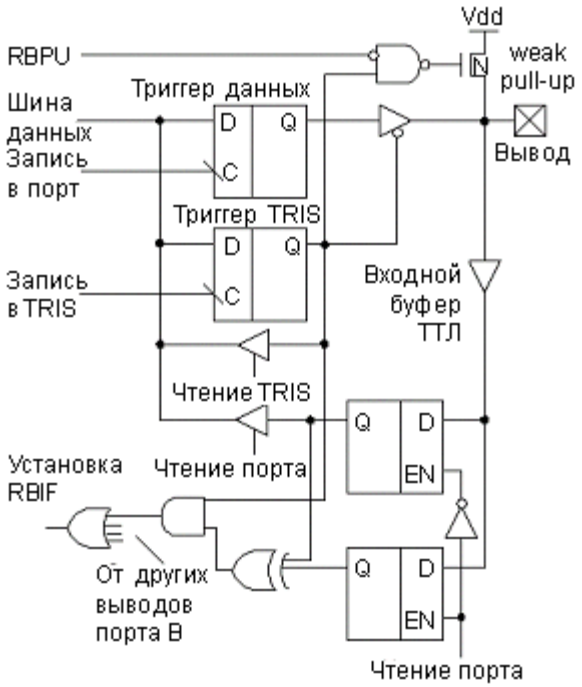

Рис. 5.10. Схема линий RB <7:4 порта B. Выводы порта имеют защитные диоды к Vdd и Vss.

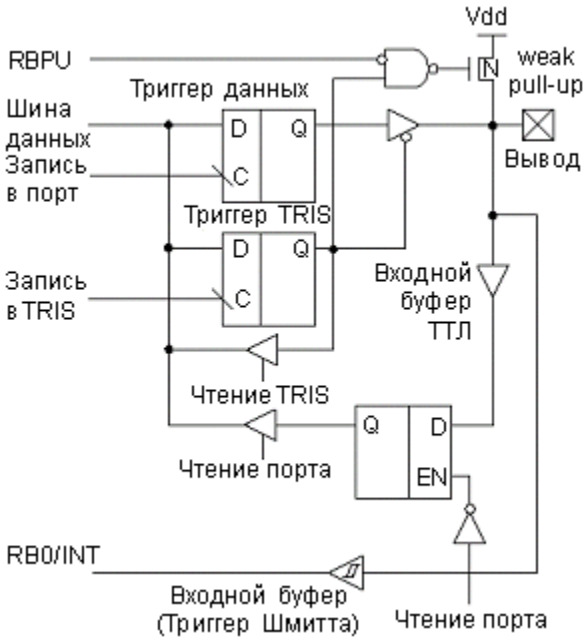

Рис. 5.11. Схема линий RB <3:0 порта B. Выводы порта имеют защитные диоды к Vdd и Vss

При организации двунаправленных портов необходимо учитывать особенности организации ввода/вывода данных МК. Любая команда, которая осуществляет запись, выполняет ее внутри как "чтение-модификация-запись". Например, команды BCF и BSF считывают порт целиком, модифицируют один бит и выводят результат обратно. Здесь необходима осторожность. В частности, команда BSF PORTB, 5 (установить в единицу бит 5 порта B) сначала считывает все реальные значения сигналов, присутствующие в данный момент на выводах порта. Затем выполняются действия над битом 5, и новое значение байта целиком записывается в выходные фиксаторы. Если другой бит регистра PORTB используется в качестве двунаправленного ввода/вывода (скажем, бит 0), и в данный момент он определен как входной, то входной сигнал на этом выводе будет считан и записан обратно в выходной триггер-фиксатор этого же вывода, стирая предыдущее состояние. До тех пор, пока эта ножка остается в режиме ввода, никаких проблем не возникает. Однако если позднее линия 0 переключится в режим вывода, ее состояние будет неопределенным.

На ножку, работающую в режиме вывода, не должны нагружаться внешние источники токов ("монтажное И", "монтажное ИЛИ"). Большие результирующие токи могут повредить кристалл.

Необходимо выдерживать определенную последовательность обращения к портам ввода/вывода. Запись в порт вывода происходит в конце командного цикла. Но при чтении данные должны быть стабильны в начале командного цикла. Будьте внимательны в операциях чтения, следующих сразу за записью в тот же порт. Здесь надо учитывать инерционность установления напряжения на выводах. Может потребоваться программная задержка, чтобы напряжение на ножке (которое зависит от нагрузки) успело стабилизироваться до начала исполнения следующей команды чтения.

5.2.10. Модуль таймера и регистр таймера

Структура модуля таймера/счетчика TIMER0 и его взаимосвязь с регистрами TMR0 и OPTION показаны на рис. 5.12. TIMER0 является программируемым модулем и содержит следующие компоненты:

- 8-разрядный таймер/счетчик TMR0 с возможностью чтения и записи как регистр;
- 8-разрядный программно управляемый предварительный делитель (предделитель);
- мультиплексор входного сигнала для выбора внутреннего или внешнего тактового сигнала;
- схему выбора фронта внешнего тактового сигнала;
- формирователь запроса прерывания по переполнению регистра TMR0 с FFh до 00h.

Режим таймера выбирается путем сбрасывания в ноль бита T0CS регистра OPTION <5>. В режиме таймера TMR0 инкрементируется каждый командный цикл (без пределителя). После записи информации в TMR0 инкрементирование его начнется после двух командных циклов. Это происходит со всеми командами, которые производят запись или чтение-модификацию-запись TMR0 (например, MOVF TMR0, CLRF TMR0). Избежать этого можно при помощи записи в TMR0 скорректированного значения. Если TMR0 нужно проверить на равенство нулю без останова счета, следует использовать инструкцию MOVF TMR0,W.

Режим счетчика выбирается путем установки в единицу бита T0CS регистра OPTION<5>. В этом режиме регистр TMR0 будет инкрементироваться либо нарастающим, либо спадающим фронтом на выводе RA4/T0CKI от внешних событий. Направление фронта определяется управляющим битом T0SE в регистре OPTION<4>. При T0SE = 0 будет выбран нарастающий фронт.

Предделитель может использоваться или совместно с TMR0, или со сторожевым (Watchdog) таймером. Вариант подключения делителя контролируется битом PSA регистра OPTION<3>. При PSA=0 делитель будет подсоединен к TMR0. Содержимое делителя программе недоступно. Коэффициент деления пределителя программируется битами PS2…PS0 регистра OPTION<2:0>.

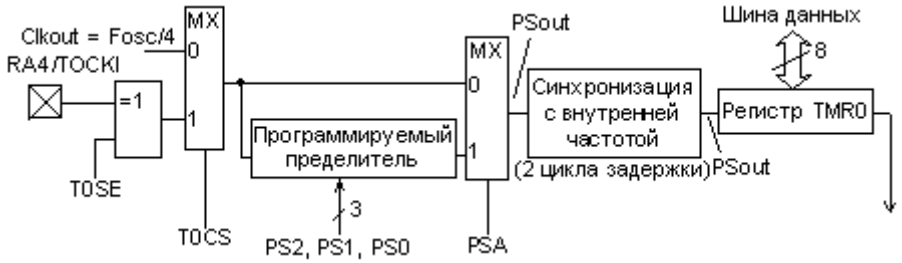

Рис. 5.12. Структурная схема таймера/счетчика TMR0.

Прерывание по TMR0 вырабатывается тогда, когда происходит переполнение регистра таймера/счетчика при переходе от FFh к 00h. Тогда устанавливается бит запроса T0IF в регистре INTCON<2>. Данное прерывание можно замаскировать битом T0IE в регистре INTCON<5>. Бит запроса T0IF должен быть сброшен программно при обработке прерывания. Прерывание по TMR0 не может вывести процессор из режима SLEEP потому, что таймер в этом режиме не функционирует.

При PSA=1 делитель будет подсоединен к сторожевому таймеру как постделитель (делитель на выходе). Возможные варианты использования пределителя показаны на рис. 5.13.

При использовании пределителя совместно с TMR0, все команды, изменяющие содержимое TMR0, обнуляют предделитель. Если предделитель используется совместно с WDT, команда CLRWDT обнуляет содержимое пределителя вместе с WDT.

При использовании модуля TIMER0 в режиме счетчика внешних событий необходимо учитывать то, что внешний тактовый сигнал синхронизируется внутренней частотой Fosc. Это приводит к появлению задержки во времени фактического инкрементирования содержимого TMR0.

Синхронизация происходит по окончании 2-го и 4-го тактов работы МК, поэтому, если предделитель не используется, то для фиксации входного события необходимо, чтобы длительности высокого и низкого состояний сигнала на входе RA4/T0CKI были бы не менее 2 периодов тактовой частоты Tosc плюс некоторая задержка (~ 20 нс).

Если модуль TIMER0 используется совместно с пределителем, то частота входного сигнала делится асинхронным счетчиком так, что сигнал на выходе пределителя становится симметричным. При этом необходимо, чтобы длительности высокого и низкого уровней сигнала на входе RA4/T0CKI были бы не менее 10 нс. Синхронизация сигнала происходит на выходе пределителя, поэтому существует небольшая задержка между фронтом внешнего сигнала и временем фактического инкремента тайме-

ра/счетчика. Эта задержка находится в диапазоне от 3 до 7 периодов колебаний тактового генератора. Таким образом, измерение интервала между событиями будет выполнено с точностью ±4 · Tosc.

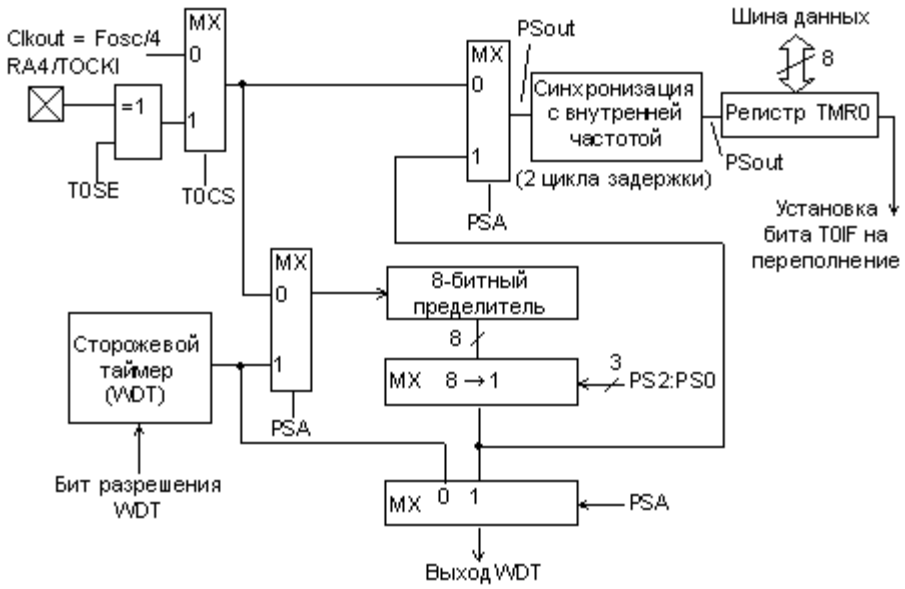

Рис. 5.13. Структура и возможные варианты использования пределителя.

5.2.11. Память данных в РПЗУ (EEPROM)

Микроконтроллеры подгруппы PIC6F8X имеют энергонезависимую память данных 64x8 EEPROM бит, которая допускает запись и чтение во время нормальной работы (во всем диапазоне питающих напряжений). Эта память не принадлежит области регистровой памяти ОЗУ. Доступ к ней осуществляется посредством косвенной адресации через регистры специальных функций: EEDATA <08h>, который содержит 8-битовые данные для чтения/записи и EEADR <09h>, включающий адрес ячейки, к которой идет обращение. Для управления процессом чтения/записи используются два регистра: EECON1 <88h> и EECON2 <89h>.

При записи байта автоматически стирается предыдущее значение, и записываются новые данные (стирание перед записью). Все эти операции производит встроенный автомат записи EEPROM. Содержимое ячеек этой памяти при выключении питания сохраняется.

Регистр EEADR может адресовать до 256 байт данных EEPROM. В МК подгруппы PIC6F8X используются только первые 64 байта, адресуемые шестью младшими битами EEADR<5:0>. Однако старшие два бита также декодируются. Поэтому эти два бита должны быть установлены в ‘0’, чтобы адрес попал в доступные 64 бита адресного пространства.

Назначение бит регистра EECON1 приведено в табл. 5.6.

Таблица 5.6. Назначение бит регистра EECON1 (адреса 88h).

U	U	U	R/W-0	R/W-x	R/W-0	R/S-0	R/S-x
-	-	-	EEIF	WRERR	WREN	WR	RD
Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
Биты 7:5 не используются (читаются как ‘0’)							

Бит 4: EEIF: бит запроса прерывания по записи в EEPROM

0 = операция записи не завершена или не начиналась

1 = операция записи завершена (должен быть сброшен программно)

Бит 3: WRERR: бит признака ошибки записи в EEPROM

0 = операция записи завершена

1 = операция записи прервана преждевременно (сбросом по /MCLR или сбросом от WDT)

Бит 2: WREN: бит разрешения записи в EEPROM

0 = запрещена запись в EEPROM

1 = разрешены циклы записи

Бит 1: WR: бит управления записью

0 = цикл записи данных в EEPROM завершен

1 = инициирует цикл записи (сбрасывается аппаратно по завершении записи. Бит WR может быть только установлен (но не сброшен) программно)

Бит 0: RD: бит управления чтением

0 = чтение данных EEPROM не инициировано

1 = инициирует чтение данных EEPROM (чтение занимает один цикл. Бит RD сбрасывается аппаратно. Бит RD может быть только установлен (но не сброшен) программно)

Регистр EECON2 не является физическим регистром. Он используется исключительно при организации записи данных в EEPROM. Чтение регистра EECON2 дает нули.

При считывании данных из памяти EEPROM необходимо записать нужный адрес в EEADR регистр и затем установить бит RD EECON1<0> в единицу. Данные появятся в следующем командном цикле в регистре EEDATA и могут быть прочитаны. Данные в регистре EEDATA фиксируются.

При записи в память EEPROM необходимо сначала записать адрес в EEADR-регистр и данные в EEDATA-регистр. Затем следует выполнить специальную последовательность команд, производящую непосредственную запись:

```
movlw 55h
movwf EECON2
movlw AAh
movwf EECON2
bsf EECON1,WR;установить WR бит, начать запись
```

Во время выполнения этого участка программы все прерывания должны быть запрещены, для точного выполнения временной диаграммы. Время записи – примерно 10 мс. Фактическое время записи может изменяться в зависимости от напряжения, температуры и индивидуальных свойств кристалла. В конце записи бит WR автоматически обнуляется, а флаг завершения записи EEIF, он же запрос на прерывание, устанавливается.

Для предотвращения случайных записей в память данных предусмотрен специальный бит WREN в регистре EECON1. Рекомендуется держать бит WREN выключенным, кроме тех случаев, когда нужно обновить память данных. Более того, кодовые сегменты, которые устанавливают бит WREN, и те, которые выполняют запись, следует хранить на различных адресах, чтобы избежать случайного выполнения их обоих при сбое программы.

5.2.12. Организация прерываний

МК подгруппы PIC16F8X имеют четыре источника прерываний:

- внешнее прерывание с вывода RB0/INT;
- прерывание от переполнения счетчика/таймера TMR0;
- прерывание от изменения сигналов на линиях порта RB<7:4>;
- прерывание по окончании записи данных в EEPROM.

Все прерывания имеют один и тот же вектор/адрес – 0004h. Однако в управляющем регистре прерываний INTCON соответствующим битом-признаком записывается, от какого именно источника поступил запрос прерывания. Исключение составляет прерывание по завершении записи в EEPROM, признак которого находится в регистре EECON1.

Бит общего разрешения/запрещения прерывания GIE (INTCON <7>) разрешает (если = 1) все индивидуально незамаскированные прерывания или запрещает их (если = 0). Каждое прерывание в отдельности может быть дополнительно разрешено/запрещено установкой/сбросом соответствующего бита в регистре INTCON.

Бит GIE при сбросе обнуляется. Когда начинает обрабатываться прерывание, бит GIE обнуляется, чтобы запретить дальнейшие прерывания, адрес возврата посылается в стек, а в программный счетчик загружается адрес 0004h. Время реакции на прерывание для внешних событий, таких как прерывание от ножки INT или порта B, составляет приблизительно пять циклов. Это на один цикл меньше, чем для внутренних событий, таких как прерывание по переполнению от таймера TMR0. Время реакции всегда одинаковое.

В подпрограмме обработки прерывания источник прерывания может быть определен по соответствующему биту в регистре признаков. Этот флаг-признак должен быть программно сброшен внутри подпрограммы. Признаки запросов прерываний не зависят от соответствующих маскирующих битов и бита общего маскирования GIE.

Команда возврата из прерывания RETFIE завершает прерывающую подпрограмму и устанавливает бит GIE, чтобы опять разрешить прерывания.

Логика прерываний контроллера изображена на рис. 5.14.

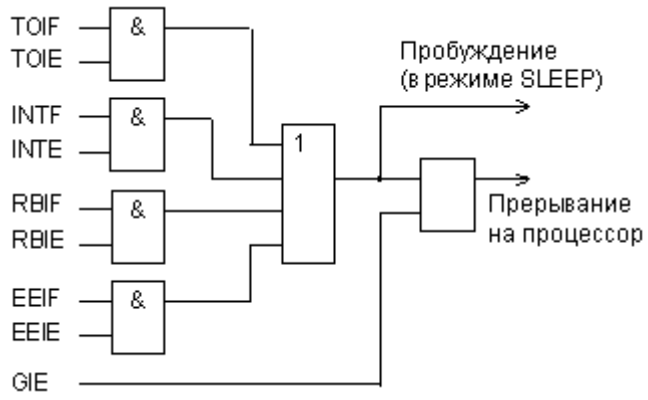

Рис. 5.14. Логика прерываний микроконтроллера.

Внешнее прерывание на ножке RB0/INT осуществляется по фронту: либо по нарастающему (если в регистре OPTION бит INTEDG=1), либо по спадающему (если INTEDG=0). Когда фронт обнаруживается на ножке INT, бит запроса INTF устанавливается в единицу (INTCON <1>). Это прерывание может быть замаскировано сбросом управляющего бита INTE в ноль (INTCON <4>). Бит запроса INTF необходимо очистить прерывающей программой перед тем, как опять разрешить это прерыва-

ние. Прерывание INT может вывести процессор из режима SLEEP, если перед входом в этот режим бит INTE был установлен в единицу. Состояние бита GIE также определяет, будет ли процессор переходить на подпрограмму прерывания после выхода из режима SLEEP.

Переполнение счетчика TMR0 (FFh->00h) устанавливает в единицу бит запроса T0IF (INTCON<2>). Это прерывание может быть разрешено/запрещено установкой/сбросом бита маски T0IE (INTCON<5>). Сброс запроса T0IF – дело программы обработки.

Любое изменение сигнала на одном из четырех входов порта RB<7:4> устанавливает в единицу бит RBIF (INTCON<0>). Это прерывание может быть разрешено/запрещено установкой/сбросом бита маски RBIE (INTCON<3>). Сброс запроса RBIF – дело программы обработки.

Признак запроса прерывания по завершении записи в EEPROM, EEIF (EECON1<4>) устанавливается в единицу по окончании автоматической записи данных в EEPROM. Это прерывание может быть замаскировано сбросом бита EEIE (INTCON<6>). Сброс запроса EEIF – дело программы обработки.

11. Лекция: Особенности разработки цифровых устройств на основе микроконтроллеров

В этой лекции рассказывается об основных этапах проектирования и разработки цифровых устройств и систем на основе микроконтроллеров, а также приводится обзор методов совместной отладки аппаратных и программных средств.

6.1. Разработка микропроцессорной системы на основе микроконтроллера

6.1.1. Основные этапы разработки

МПС на основе МК используются чаще всего в качестве встроенных систем для решения задач управления некоторым объектом. Важной особенностью данного применения является работа в реальном времени, т.е. обеспечение реакции на внешние события в течение определенного временного интервала. Такие устройства получили название контроллеров.

Технология проектирования контроллеров на базе МК полностью соответствует принципу неразрывного проектирования и отладки аппаратных и программных средств, принятому в микропроцессорной технике. Это означает, что перед разработчиком такого рода МПС стоит задача реализации полного цикла проектирования, начиная от разработки алгоритма функционирования и заканчивая комплексными испытаниями в составе изделия, а, возможно, и сопровождением при производстве. Сложившаяся к настоящему времени методология проектирования контроллеров может быть представлена так, как показано на рис. 6.1.

В техническом задании формулируются требования к контроллеру с точки зрения реализации определенной функции управления. Техническое задание включает в себя набор требований, который определяет, что пользователь хочет от контроллера и что разрабатываемый прибор должен делать. Техническое задание может иметь вид текстового описания, не свободного в общем случае от внутренних противоречий.

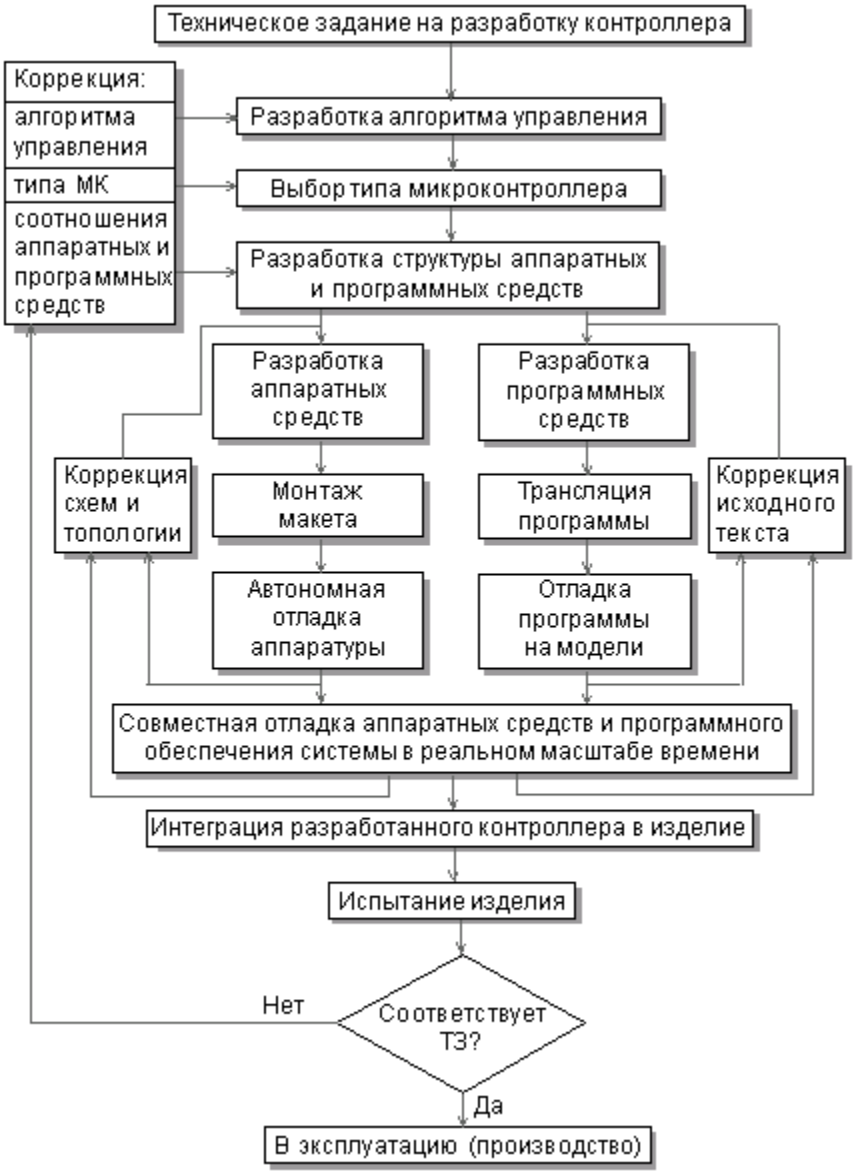

Рис. 6.1. Основные этапы разработки контроллера.

На основании требований пользователя составляется функциональная спецификация, которая определяет функции, выполняемые контроллером для пользователя после завершения проектирования, уточняя тем самым, насколько устройство соответствует предъявляемым требованиям. Она включает в себя описания форматов данных, как на входе, так и на выходе, а также внешние условия, управляющие действиями контроллера.

Функциональная спецификация и требования пользователя являются критериями оценки функционирования контролера после завершения проектирования. Может потребоваться проведение нескольких итераций, включающих обсуждение требований и функциональной спецификации с потенциальными пользователями контроллера, и соответствующую коррекцию требований и спецификации. Требования к типу используемого МК формулируются на данном этапе чаще всего в неявном виде.

Этап разработки алгоритма управления является наиболее ответственным, поскольку ошибки данного этапа обычно обнаруживаются только при испытаниях законченного изделия и приводят к необходимости дорогостоящей переработки всего устройства. Разработка алгоритма обычно сводится к выбору одного из нескольких возможных вариантов алгоритмов, отличающихся соотношением объема программного обеспечения и аппаратных средств.

При этом необходимо исходить из того, что максимальное использование аппаратных средств упрощает разработку и обеспечивает высокое быстродействие контроллера в целом, но сопровождается, как правило, увеличением стоимости и потребляемой мощности. Связано это с тем, что увеличение

доли аппаратных средств достигается либо путем выбора более сложного МК, либо путем использования специализированных интерфейсных схем. И то, и другое приводит к росту стоимости и энергопотребления. Увеличение удельного веса программного обеспечения позволяет сократить число элементов контроллера и стоимость аппаратных средств, но это приводит к снижению быстродействия, увеличению необходимого объема внутренней памяти МК, увеличению сроков разработки и отладки программного обеспечения. Критерием выбора здесь и далее является возможность максимальной реализации заданных функций программными средствами при минимальных аппаратных затратах и при условии обеспечения заданных показателей быстродействия и надежности в полном диапазоне условий эксплуатации. Часто определяющими требованиями являются возможность защиты информации (программного кода) контроллера, необходимость обеспечения максимальной продолжительности работы в автономном режиме и другие. В результате выполнения этого этапа окончательно формулируются требования к параметрам используемого МК.

При выборе типа МК учитываются следующие основные характеристики:

- разрядность;
- быстродействие;
- набор команд и способов адресации;
- требования к источнику питания и потребляемая мощность в различных режимах;
- объем ПЗУ программ и ОЗУ данных;
- возможности расширения памяти программ и данных;
- наличие и возможности периферийных устройств, включая средства поддержки работы в реальном времени (таймеры, процессоры событий и т.п.);
- возможность перепрограммирования в составе устройства;
- наличие и надежность средств защиты внутренней информации;
- возможность поставки в различных вариантах конструктивного исполнения;
- стоимость в различных вариантах исполнения;
- наличие полной документации;
- наличие и доступность эффективных средств программирования и отладки МК;
- количество и доступность каналов поставки, возможность замены изделиями других фирм.

Список этот не является исчерпывающим, поскольку специфика проектируемого устройства может перенести акцент требований на другие параметры МК. Определяющими могут оказаться, например, требования к точности внутреннего компаратора напряжений или наличие большого числа выходных каналов ШИМ.

Номенклатура выпускаемых в настоящее время МК исчисляется тысячами типов изделий различных фирм. Современная стратегия модульного проектирования обеспечивает потребителя разнообразием моделей МК с одним и тем же процессорным ядром. Такое структурное разнообразие открывает перед разработчиком возможность выбора оптимального МК, не имеющего функциональной избыточности, что минимизирует стоимость комплектующих элементов.

Однако для реализации на практике возможности выбора оптимального МК необходима достаточно глубокая проработка алгоритма управления, оценка объема исполняемой программы и числа линий сопряжения с объектом на этапе выбора МК. Допущенные на данном этапе просчеты могут впоследствии привести к необходимости смены модели МК и повторной разводки печатной платы макета контроллера. В таких условиях целесообразно выполнять предварительное моделирование основных элементов прикладной программы с использованием программно-логической модели выбранного МК.

При отсутствии МК, обеспечивающего требуемые по ТЗ характеристики проектируемого контроллера, необходим возврат к этапу разработки алгоритма управления и пересмотр выбранного соотношения между объемом программного обеспечения и аппаратных средств. Отсутствие подходящего МК чаще всего означает, что для реализации необходимого объема вычислений (алгоритмов управления) за отведенное время нужна дополнительная аппаратная поддержка. Отрицательный результат поиска МК с требуемыми характеристиками может быть связан также с необходимостью обслуживания

большого числа объектов управления. В этом случае возможно использование внешних схем обрамления МК.

На этапе разработки структуры контроллера окончательно определяется состав имеющихся и подлежащих разработке аппаратных модулей, протоколы обмена между модулями, типы разъемов. Выполняется предварительная проработка конструкции контроллера. В части программного обеспечения определяются состав и связи программных модулей, язык программирования. На этом же этапе осуществляется выбор средств проектирования и отладки.

Возможность перераспределения функций между аппаратными и программными средствами на данном этапе существует, но она ограничена характеристиками уже выбранного МК. При этом необходимо иметь в виду, что современные МК выпускаются, как правило, сериями (семействами) контроллеров, совместимых программно и конструктивно, но различающихся по своим возможностям (объем памяти, набор периферийных устройств и т.д.). Это дает возможность выбора структуры контроллера с целью поиска наиболее оптимального варианта реализации.

Нельзя не упомянуть здесь о новой идеологии разработки устройств на базе МК, предложенной фирмой «Scenix». Она основана на использовании высокоскоростных RISC-микроконтроллеров серии SX с тактовой частотой до 100 МГц. Эти МК имеют минимальный набор встроенной периферии, а все более сложные периферийные модули эмулируются программными средствами. Такие модули программного обеспечения называются «виртуальными периферийными устройствами», они обеспечивают уменьшение числа элементов контроллера, времени разработки, увеличивают гибкость исполнения. К настоящему времени разработаны целые библиотеки виртуальных устройств, содержащие отлаженные программные модули таких устройств как модули ШИМ и ФАПЧ, последовательные интерфейсы, генераторы и измерители частоты, контроллеры прерываний и многие другие.

6.1.2. Разработка и отладка аппаратных средств

После разработки структуры аппаратных и программных средств дальнейшая работа над контроллером может быть распараллелена. Разработка аппаратных средств включает в себя разработку общей принципиальной схемы, разводку топологии плат, монтаж макета и его автономную отладку. Время выполнения этих этапов зависит от имеющегося набора апробированных функционально-топологических модулей, опыта и квалификации разработчика. На этапе ввода принципиальной схемы и разработки топологии используются, как правило, распространенные системы проектирования типа «ACCEL EDA» или «OrCad».

Автономная отладка аппаратуры на основе МК с открытой архитектурой предполагает контроль состояния многоразрядных магистралей адреса и данных с целью проверки правильности обращения к внешним ресурсам памяти и периферийным устройствам. Закрытая архитектура МК предполагает реализацию большинства функций разрабатываемого устройства внутренними средствами микроконтроллера. Поэтому разрабатываемый контроллер будет иметь малое число периферийных ИС, а обмен с ними будет идти преимущественно по последовательным интерфейсам. Здесь на первый план выйдут вопросы согласования по нагрузочной способности параллельных портов МК и отладка алгоритмов обмена по последовательным каналам.

6.1.3. Разработка и отладка программного обеспечения

Содержание этапов разработки программного обеспечения, его трансляции и отладки на моделях существенно зависит от используемых системных средств. В настоящее время ресурсы 8-разрядных МК достаточны для поддержки программирования на языках высокого уровня. Это позволяет использовать все преимущества структурного программирования, разрабатывать программное обеспечение с использованием раздельно транслируемых модулей. Одновременно продолжают широко использоваться языки низкого уровня типа ассемблера, особенно при необходимости обеспечения контролируемых интервалов времени. Задачи предварительной обработки данных часто требуют использования вычислений с плавающей точкой, трансцендентных функций.

В настоящее время самым мощным средством разработки программного обеспечения для МК являются интегрированные среды разработки, имеющие в своем составе менеджер проектов, текстовый редактор и симулятор, а также допускающие подключение компиляторов языков высокого уровня типа Паскаль или Си. При этом необходимо иметь в виду, что архитектура многих 8-разрядных МК вследствие малого количества ресурсов, страничного распределения памяти, неудобной индексной адресации и некоторых других архитектурных ограничений не обеспечивает компилятору возможности генерировать эффективный код. Для обхода этих ограничений разработчики ряда компиляторов вынуждены были перекладывать на пользователя заботу об оптимизации кода программы.

Для проверки и отладки программного обеспечения используются так называемые программные симуляторы, предоставляющие пользователю возможность выполнять разработанную программу на программно-логической модели МК. Программные симуляторы распространяются, как правило, бесплатно и сконфигурированы сразу на несколько МК одного семейства. Выбор конкретного типа МК среди моделей семейства обеспечивает соответствующая опция меню конфигурации симулятора. При этом моделируется работа ЦП, всех портов ввода/вывода, прерываний и другой периферии. Карта памяти моделируемого МК загружается в симулятор автоматически, отладка ведется в символьных обозначениях регистров.

Загрузив программу в симулятор, пользователь имеет возможность запускать ее в пошаговом или непрерывном режимах, задавать условные или безусловные точки останова, контролировать и свободно модифицировать содержимое ячеек памяти и регистров симулируемого МК.

6.1.4. Методы и средства совместной отладки аппаратных и программных средств

Этап совместной отладки аппаратных и программных средств в реальном масштабе времени является самым трудоемким и требует использования инструментальных средств отладки. К числу основных инструментальных средств отладки относятся:

- внутрисхемные эмуляторы;
- платы развития (оценочные платы);
- мониторы отладки;
- эмуляторы ПЗУ.

Внутрисхемный эмулятор – программно-аппаратное средство, способное заменить эмулируемый МК в реальной схеме. Стыковка внутрисхемного эмулятора с отлаживаемой системой производится при помощи кабеля со специальной эмуляционной головкой, которая вставляется вместо МК в отлаживаемую систему. Если МК нельзя удалить из отлаживаемой системы, то использование эмулятора возможно, только если этот микроконтроллер имеет отладочный режим, при котором все его выводы находятся в третьем состоянии. В этом случае для подключения эмулятора используют специальный адаптер-клипсу, который подключается непосредственно к выводам эмулируемого МК.

Внутрисхемный эмулятор – это наиболее мощное и универсальное отладочное средство, которое делает процесс функционирования отлаживаемого контроллера прозрачным, т.е. легко контролируемым, произвольно управляемым и модифицируемым.

Платы развития, или, как принято их называть в зарубежной литературе, оценочные платы (Evaluation Boards), являются своего рода конструкторами для макетирования электронных устройств. Обычно это печатная плата с установленным на ней МК и всей необходимой ему стандартной периферией. На этой плате также устанавливают схемы связи с внешним компьютером. Как правило, там же имеется свободное поле для монтажа прикладных схем пользователя. Иногда предусмотрена уже готовая разводка для установки дополнительных устройств, рекомендуемых фирмой. Например, ПЗУ, ОЗУ, ЖКИ-дисплей, клавиатура, АЦП и др. Кроме учебных или макетных целей, такие доработанные пользователем платы можно использовать в качестве одноплатных контроллеров, встраиваемых в малосерийную продукцию.

Для большего удобства платы развития комплектуются еще и простейшим средством отладки на базе **монитора** отладки. Используются два типа мониторов отладки: один для МК, имеющих внешнюю шину, а второй – для МК, не имеющих внешней шины.

В первом случае отладочный монитор поставляется в виде микросхемы ПЗУ, которая вставляется в специальную розетку на плате развития. Плата также имеет ОЗУ для программ пользователя и канал связи с внешним компьютером или терминалом. Во втором случае плата развития имеет встроенные схемы программирования внутреннего ПЗУ МК, которые управляются от внешнего компьютера. При этом программа монитора просто заносится в ПЗУ МК совместно с прикладными кодами пользователя. Прикладная программа должна быть специально подготовлена: в нужные места необходимо вставить вызовы отладочных подпрограмм монитора. Затем осуществляется пробный прогон. Чтобы внести в программу исправления, пользователю надо стереть ПЗУ и произвести повторную запись. Готовую прикладную программу получают из отлаженной путем удаления всех вызовов мониторных функций и самого монитора отладки. Возможности отладки, предоставляемые комплектом «плата развития плюс монитор», не столь универсальны, как возможности внутрисхемного эмулятора, да и некоторая часть ресурсов МК в процессе отладки отбирается для работы монитора. Тем не менее, наличие набора готовых программно-аппаратных средств, позволяющих без потери времени приступить к монтажу и отладке проектируемой системы, во многих случаях является решающим фактором. Особенно если учесть, что стоимость такого комплекта несколько меньше, чем стоимость более универсального эмулятора.

Эмулятор ПЗУ – программно-аппаратное средство, позволяющее замещать ПЗУ на отлаживаемой плате, и подставляющее вместо него ОЗУ, в которое может быть загружена программа с компьютера через один из стандартных каналов связи. Это устройство позволяет пользователю избежать многократных циклов перепрограммирования ПЗУ. Эмулятор ПЗУ нужен только для МК, которые могут обращаться к внешней памяти программ. Это устройство сравнимо по сложности и по стоимости с платами развития и имеет одно большое достоинство: универсальность. Эмулятор ПЗУ может работать с любыми типами МК.

Эмулируемая память доступна для просмотра и модификации, но контроль над внутренними управляющими регистрами МК был до недавнего времени невозможен.

В последнее время появились модели интеллектуальных эмуляторов ПЗУ, которые позволяют «заглядывать» внутрь МК на плате пользователя. Интеллектуальные эмуляторы представляют собой гибрид из обычного эмулятора ПЗУ, монитора отладки и схем быстрого переключения шины с одного на другой. Это создает эффект, как если бы монитор отладки был установлен на плате пользователя и при этом он не занимает у МК никаких аппаратных ресурсов, кроме небольшой зоны программных шагов, примерно 4К.

Этап совместной отладки аппаратных и программных средств в реальном масштабе времени завершается, когда аппаратура и программное обеспечение совместно обеспечивают выполнение всех шагов алгоритма работы системы. В конце этапа отлаженная программа заносится с помощью программатора в энергонезависимую память МК, и проверяется работа контроллера без эмулятора. При этом используются лабораторные источники питания. Часть внешних источников сигналов может моделироваться.

Этап интеграции разработанного контроллера в изделие заключается в повторении работ по совместной отладке аппаратуры и управляющей программы, но при работе в составе изделия, питании от штатного источника и с информацией от штатных источников сигналов и датчиков.

Состав и объем испытаний разработанного и изготовленного контроллера зависит от условий его эксплуатации и определяется соответствующими нормативными документами. Проведение испытаний таких функционально сложных изделий, как современные контроллеры, может потребовать разработки специализированных средств контроля состояния изделия во время испытаний.

Глава 7. Организация персонального компьютера

13. Лекция: Архитектура и процессоры персональных компьютеров

В этой лекции рассказывается об особенностях архитектуры персональных компьютеров семейства IBM PC, о процессорах, применяемых в персональных компьютерах, об их функциях, характеристиках, этапах их эволюции.

Как уже отмечалось, персональный компьютер представляет собой наиболее развитый вид микропроцессорных систем. На основе персональных компьютеров можно строить самые сложные контрольно-измерительные, управляющие, вычислительные и информационные системы. Имеющиеся в персональном компьютере аппаратные и программные средства делают его универсальным инструментом для самых разных задач.

В случае вычислительных и информационных систем персональный компьютер не нуждается в подключении нестандартной аппаратуры, все сводится к подбору или написанию необходимого программного обеспечения. В случае же контрольно-измерительных и управляющих систем персональный компьютер оснащается набором инструментов для сопряжения с внешними устройствами и соответствующими программными средствами. Во многих случаях строить систему на основе персонального компьютера оказывается гораздо проще, быстрее и даже дешевле, чем проектировать ее с нуля на базе какого-то микропроцессора, микропроцессорного комплекта или микроконтроллера.

Конечно, в большинстве случаев система на основе персонального компьютера оказывается сильно избыточной, это плата за универсальность. Но в то же время один и тот же компьютер может решать самые разнообразные задачи. Например, в системе управления технологическими процессами или научными установками он может математически моделировать происходящие процессы, выдавать в реальном времени управляющие сигналы, принимать в реальном времени ответные сигналы, накапливать информацию, обрабатывать ее, обмениваться информацией с другими компьютерами и т.д. Развитый интерфейс пользователя (видеомонитор, полноразмерная клавиатура, мышь) делают работу с персональным компьютером комфортной и эффективной. А стоимость персональных компьютеров вследствие большого объема выпуска постоянно снижается. Поэтому их использование не только удобно, но и экономически выгодно.

Но чтобы грамотно и полноценно использовать персональный компьютер в составе любых систем, надо иметь представление о его архитектуре, об основных принципах построения, об устройствах, входящих в его состав, наконец, о внешних интерфейсах.

7.1. Архитектура персонального компьютера

Персональный компьютер типа IBM PC имеет довольно традиционную архитектуру микропроцессорной системы и содержит все обычные функциональные узлы: процессор, постоянную и оперативную память, устройства ввода/вывода, системную шину, источник питания (рис.7.1). Основные особенности архитектуры персональных компьютеров сводятся к принципам компоновки аппаратуры, а также к выбранному набору системных аппаратных средств.

Рис. 7.1. Архитектура персонального компьютера типа IBM PC.

Функции основных узлов компьютера следующие:

- **Центральный процессор** — это микропроцессор со всеми необходимыми вспомогательными микросхемами, включая внешнюю кэш-память и контроллер системной шины. (О кэш-памяти подробнее будет рассказано в следующих разделах). В большинстве случаев именно центральный процессор осуществляет обмен по системной шине.
- **Оперативная память** может занимать почти все адресуемое пространство памяти процессора. Однако чаще всего ее объем гораздо меньше. В современных персональных компьютерах стандартный объем системной памяти составляет, как правило, от 64 до 512 Мбайт. Оперативная память компьютера выполняется на микросхемах динамической памяти и поэтому требует регенерации.
- **Постоянная память (ROM BIOS — Base Input/Output System)** имеет небольшой объем (до 64 Кбайт), содержит программу начального запуска, описание конфигурации системы, а также драйверы (программы нижнего уровня) для взаимодействия с системными устройствами.
- **Контроллер прерываний** преобразует аппаратные прерывания системной магистрали в аппаратные прерывания процессора и задает адреса векторов прерывания. Все режимы функционирования контроллера прерываний задаются программно процессором перед началом работы.
- **Контроллер прямого доступа к памяти** принимает запрос на ПДП из системной магистрали, передает его процессору, а после предоставления процессором магистрали производит пересылку данных между памятью и устройством ввода/вывода. Все режимы функционирования контроллера ПДП задаются программно процессором перед началом работы. Использование встроенных в компьютер контроллеров прерываний и ПДП позволяет существенно упростить аппаратуру применяемых плат расширения.
- **Контроллер регенерации** осуществляет периодическое обновление информации в динамической оперативной памяти путем проведения по шине специальных циклов регенерации. На время циклов регенерации он становится хозяином (задатчиком) шины.
- **Перестановщик байтов данных** помогает производить обмен данными между 16-разрядным и 8-разрядным устройствами, пересылать целые слова или отдельные байты.
- **Часы реального времени и таймер-счетчик** — это устройства для внутреннего контроля времени и даты, а также для программной выдержки временных интервалов, программного задания частоты и т.д.
- Системные устройства ввода/вывода — это те устройства, которые необходимы для работы компьютера и взаимодействия со стандартными внешними устройствами по параллельному и последовательному интерфейсам. Они могут быть выполнены на материнской плате, а могут располагаться на платах расширения.
- **Платы расширения** устанавливаются в слоты (разъемы) системной магистрали и могут содержать оперативную память и устройства ввода/вывода. Они могут обмениваться данными с другими устройствами на шине в режиме программного обмена, в режиме прерываний и в режиме ПДП. Предусмотрена также возможность захвата шины, то есть полного отключения от шины всех системных устройств на некоторое время.

Важная особенность подобной архитектуры — ее открытость, то есть возможность включения в компьютер дополнительных устройств, причем как системных устройств, так и разнообразных плат расширения. Открытость предполагает также возможность простого встраивания программ пользователя на любом уровне программного обеспечения компьютера.

Первый компьютер семейства, получивший широкое распространение, IBM PC XT, был выполнен на базе оригинальной системной магистрали PC XT-Bus. В дальнейшем (начиная с IBM PC AT) она была доработана до магистрали, ставшей стандартной и получившей название ISA (Industry Standard Architecture). До недавнего времени ISA оставалась основой компьютера. Однако, начиная с появления процессоров i486 (в 1989 году), она перестала удовлетворять требованиям производительности, и ее стали дублировать более быстрыми шинами: VLB (VESA Local Bus) и PCI (Peripheral Component Interconnect bus) или заменять совместимой с ISA магистралью EISA (Enhanced ISA). Постепенно шина PCI вытеснила конкурентов и стала фактическим стандартом, а начиная с 1999 года в новых компьютерах рекомендуется полностью отказываться от магистрали ISA, оставляя только PCI. Правда, при этом приходится отказываться от применения плат расширения, разработанных за долгие годы для подключения к магистрали ISA.

Другое направление совершенствования архитектуры персонального компьютера связано с максимальным ускорением обмена информацией с системной памятью. Именно из системной памяти компьютер читает все исполняемые команды, и в системной же памяти он хранит данные. То есть больше всего обращений процессор совершает именно к памяти. Ускорение обмена с памятью приводит к существенному ускорению работы всей системы в целом. Но при использовании для обмена с памятью системной магистрали приходится учитывать скоростные ограничения магистрали. Системная магистраль должна обеспечивать сопряжение с большим числом устройств, поэтому она должна иметь довольно большую протяженность; она требует применения входных и выходных буферов для согласования с линиями магистрали. Циклы обмена по системной магистрали сложны, и ускорять их нельзя. В результате существенного ускорения обмена процессора с памятью по магистрали добиться невозможно.

Разработчиками был предложен следующий подход. Системная память подключается не к системной магистрали, а к специальной высокоскоростной шине, находящейся «ближе» к процессору, не требующей сложных буферов и больших расстояний. В таком случае обмен с памятью идет с максимально возможной для данного процессора скоростью, и системная магистраль не замедляет его. Особенно актуальным это становится с ростом быстродействия процессора (сейчас тактовые частоты процессоров персональных компьютеров достигают 1—3 ГГц).

Таким образом, структура персонального компьютера из одношинной, применявшейся только в первых компьютерах, становится трехшинной (рис. 7.2).

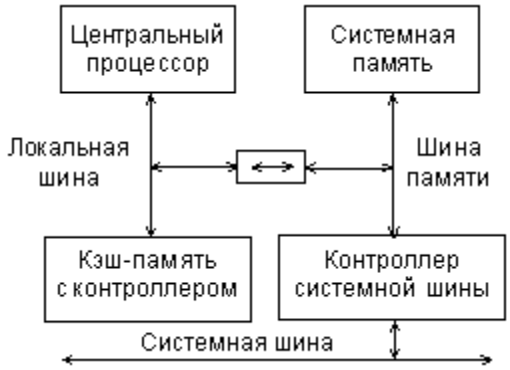

Рис. 7.2. Организация связей в случае трехшинной структуры.

Назначение шин следующее:

- к локальной шине подключаются центральный процессор и кэш-память (быстрая буферная память);
- к шине памяти подключается оперативная и постоянная память компьютера, а также контроллер системной шины;

- к системной шине (магистрали) подключаются все остальные устройства компьютера.

Все три шины имеют адресные линии, линии данных и управляющие сигналы. Но состав и назначение линий этих шин не совпадают между собой, хотя они и выполняют одинаковые функции. С точки зрения процессора, системная шина (магистраль) в системе всего одна, по ней он получает данные и команды и передает данные как в память, так и в устройства ввода/вывода.

Временные задержки между системной памятью и процессором в данном случае минимальны, так как локальная шина и шина памяти соединены только простейшими быстродействующими буферами. Еще меньше задержки между процессором и кэш-памятью, подключаемой непосредственно к локальной шине процессора и служащей для ускорения обмена процессора с системной памятью.

Если в компьютере применяются две системные шины, например, ISA и PCI, то каждая из них имеет свой собственный контроллер шины, и работают они параллельно, не влияя друг на друга. Тогда получается уже четырехшинная, а иногда и пятишинная структура. Пример такой структуры компьютера приведен на рис. 7.3.

Рис. 7.3. Пример многошинной структуры.

В наиболее распространенных настольных компьютерах класса Desk-top в качестве конструктивной основы используется системная или материнская плата (motherboard), на которой располагаются все основные системные узлы компьютера, а также несколько разъемов (слотов) системной шины для подключения дочерних плат — плат расширения (интерфейсных модулей, контроллеров, адаптеров). Как правило, современные системные платы допускают замену процессора, выбор его тактовой частоты, замену и наращивание оперативной памяти, выбор режимов работы других узлов.

На системной плате сейчас обычно располагаются также основные средства внешнего интерфейса, служащие для присоединения как встроенных устройств (например, дисковых носителей), так и внешних устройств компьютера (например, клавиатуры, мыши, принтера, сканера, модема). Для подключения видеомонитора, как правило, используется специальная плата расширения (контроллер дисплея), вставляемая в один из слотов. Это позволяет заменять ее более мощной при необходимости установки нового монитора.

Отметим, что для получающих все более широкое распространение портативных персональных компьютеров класса ноутбуков (notebook) применяются несколько иные конструктивные решения. В частности, в них отсутствуют слоты расширения системной шины, а практически все узлы компьютера выполняются на одной плате. Но мы в основном будем говорить о компьютерах типа desktop (настольных), так как именно они наиболее приспособлены для построения сложных систем, допускают довольно простую модернизацию (upgrade) и настройку на конкретные нужды пользователя.

7.2. Процессоры персональных компьютеров

Несмотря на то, что первый персональный компьютер был выпущен фирмой Apple, сейчас персональными компьютерами называют в основном IBM PC-совместимые компьютеры. Это связано, прежде всего, с тем, что фирма IBM выбрала правильную рыночную политику: она не скрывала принципов устройства своих компьютеров и не патентовала основных решений. В результате многие производители стали выпускать совместимые компьютеры, и они быстро стали фактическим стандартом. Из-за большого объема выпуска персональные компьютеры начали быстро дешеветь. К тому же для IBM-совместимых персональных компьютеров стали разрабатывать множество программных средств, что еще больше способствовало их распространению. Поэтому, несмотря на некоторые существенные архитектурные недостатки, IBM-совместимые персональные компьютеры сейчас уверенно занимают первое место на рынке.

С самого начала фирма IBM ориентировалась на процессоры Intel. У этих процессоров были очень мощные конкуренты, например, процессоры фирм Motorola или Zilog, превосходившие процессоры Intel по многим параметрам, но именно благодаря персональным компьютерам процессоры Intel смогли выйти победителями в конкурентной борьбе. К тому же еще некоторые фирмы (например, AMD, VIA, Cyrix) выпускают Intel-совместимые процессоры. Поэтому мы рассмотрим основные особенности процессоров фирмы Intel. Это позволит нам также проследить основные тенденции в развитии процессоров за последние десятилетия.

Свой первый 16-разрядный процессор i8086 фирма Intel выпустила в 1978 году. Он мог адресовать 1 Мбайт памяти (то есть имел 20-разрядную шину адреса), производительность его при тактовой частоте 5 МГц составляла 0,33 MIPS, но вскоре появились процессоры с тактовой частотой 8 и 10 МГц. Чуть позже (через год) была выпущена упрощенная версия процессора i8086 — i8088, который отличался только 8-разрядной внешней шиной данных. За счет этого он был медленнее, чем i8086, на 20—60% при той же тактовой частоте, но зато заметно дешевле. Именно на его основе был собран очень популярный персональный компьютер IBM PC XT.

16-разрядный процессор i80286, выпущенный в 1982 году, был использован в персональных компьютерах IBM PC AT. Принципиально новым было в нем то, что он мог адресовать до 16 Мбайт памяти и имел помимо реального режима, аналогичного i8086, еще и так называемый защищенный режим, позволяющий более гибко управлять памятью. Производительность этого процессора при тактовой частоте 8 МГц составляла 1,2 MIPS.

Важным шагом стало появление в 1985 году полностью 32-разрядного процессора i80386, способного адресовать до 4 Гбайт памяти (32-разрядная адресная шина). Он имел еще более развитую систему управления памятью MMU (Memory Management Unit). Производительность его при тактовой частоте 16 МГц составляла 6 MIPS. С появлением этого процессора стала бурно развиваться операционная система MS Windows, существенно изменившая процесс работы с компьютерами типа IBM PC.

Еще одним принципиальным шагом стало создание в 1989 году процессора Intel 486DX, в котором появились встроенный математический сопроцессор, существенно ускоривший выполнение арифметических операций, и внутренняя кэш-память, ускоряющая обмен с оперативной памятью. Максимальный объем адресуемой памяти этого процессора — 4 Гбайт. На тактовой частоте 25 МГц производительность была 16,5 MIPS. Начиная с процессора 486, получило распространение так называемое умножение тактовой частоты, то есть внутреннее удвоение и даже учетверение внешней тактовой частоты (обозначается 486DX2, 486DX4).

В 1995 году появились первые процессоры Pentium, открывшие новый этап в развитии семейства. Они были 32-разрядными внутри, но имели 64-разрядную внешнюю шину данных. Принципиальным отличием было использование в них так называемой суперскалярной архитектуры, следствием чего стало более высокое быстродействие при той же тактовой частоте, что и i486DX. При тактовой частоте 66 МГц производительность процессора достигала 112 MIPS. В 1996 году тактовая частота Pentium была доведена до 200 МГц, а стоимость снизилась настолько, что он стал рядовым процессором персональных компьютеров семейства IBM PC.

В 1997 году Pentium был дополнен технологией MMX, призванной ускорять выполнение мультимедийных приложений (обработку изображений и звука). И в этом же году появился процессор Pentium II, который включает в себя технологию MMX и имеет более высокое быстродействие. Возможная тактовая частота достигла 400 МГц.

В последние годы появились процессоры Pentium III и Pentium IV, имеющие еще более развитую архитектуру и тактовую частоту, превышающую 1 ГГц у Pentium III и 3 ГГц у Pentium IV.

Таким образом, содружество компании Intel и производителей IBM-совместимых персональных компьютеров успешно продолжается. Другие фирмы, выпускающие процессоры, вряд ли существенно потеснят в ближайшие годы Intel. Поэтому рассмотрим чуть подробнее характерные особенности процессоров для персональных компьютеров этой компании.

7.2.1. Особенности процессоров 8086/8088

Описание микропроцессоров фирмы Intel мы начнем с процессоров i8086/8088. Именно заложенные в них архитектурные решения во многом определили архитектуру последующих моделей семейства Intel, поддерживающих совместимость с более ранними моделями. В том числе и с недостатками и ограничениями предыдущих моделей.

Процессор i8086 имеет совмещенную (мультиплексированную) 20-разрядную внешнюю шину адреса/данных. Данные передаются по 16 разрядам, адрес — по 20 разрядам. Шина управления имеет 16 разрядов (в частности, в нее входят строб адреса и стробы обмена с памятью и устройствами ввода/вывода). Среднее время выполнения команды занимает 12 тактов синхронизации, один цикл обмена по внешней шине требует 4 тактов (без учета тактов ожидания, вводимых при асинхронном обмене). У процессора i8088 внешняя шина данных 8-разрядная.

Одна из характерных особенностей процессоров i8086/8088 — принцип сегментирования памяти. То есть вся память представляется не в виде непрерывного пространства, а в виде нескольких кусков — сегментов заданного размера (по 64 Кбайта), положение которых в пространстве памяти можно программно изменять. Об этом уже говорилось в разделе 3.1.2 (см. рис. 3.5 и 3.6).

Процессор 8086/8088 имеет 14 регистров разрядностью по 16 бит. Об их назначении также уже говорилось в разделе 3.2.

Для ускорения выборки команд из памяти в процессоре 8086 предусмотрен внутренний 6-байтный конвейер (в процессоре 8088 — 4-байтный). Конвейер заполняется читаемыми из памяти командами во время выполнения предыдущей команды и сбрасывается (считается пустым) при выполнении любой команды перехода (даже если это команда перехода на следующий адрес).

Система команд процессора включает в себя 133 команды, поддерживающие 24 метода адресации операндов. Такое большое число команд может рассматриваться как достоинство (можно гибко выбирать команду, оптимально подходящую для каждого конкретного случая), но оно же заметно усложняет структуру процессора.

Каждая команда содержит 1, 2 или 4 байта кода команды, за которыми могут следовать 1, 2 или 4 байта операнда.

В процессоре предусмотрены программные и аппаратные прерывания, разделение внешней шины с другими процессорами или с контроллером прямого доступа к памяти, а также возможность подключения математического сопроцессора i8087, существенно увеличивающего производительность вычислений.

При старте процессора (по внешнему сигналу RESET) он переходит в адрес памяти FFFF0 и начинает выполнение программы, которая размещается начиная с этого адреса.

Процессор может обрабатывать 256 типов прерываний: внешних (аппаратных), программных и внутренних. Векторы прерываний представляют собой двойное слово (два слова по 16 разрядов), определяющее сегмент и смещение начального адреса программы обработки прерываний. Для векторов прерываний отведена область памяти с адресами 00000…003FF. Внутренние прерывания вырабатываются при особых ситуациях:

- прерывание 0 соответствует переполнению при делении на нуль;
- прерывание 1 вырабатывается после каждой команды при установленном флаге трассировки TF в регистре состояния процессора (см. раздел 3.2);
- прерывание 4 вырабатывается по специальной команде INTO, если установлен флаг переполнения OF в регистре состояния процессора (это условное прерывание по переполнению).

Особое место занимает немаскируемое прерывание NMI (Non-Masked Interrupt), которое вырабатывается при поступлении внешнего сигнала NMI и не зависит от состояния флага разрешения аппаратных прерываний IF. В компьютере оно используется для контроля четности памяти, контроля корректности обмена с памятью и устройствами ввода/вывода, а также для обработки так называемых **исключений**, то есть особых условий, возникающих в процессе работы. Немаскируемым оно называется именно потому, что его нельзя запретить.

Важная отличительная особенность процессора — разделение операций обмена с устройствами ввода/вывода и с памятью. Для обмена с устройствами ввода/вывода используются как отдельные команды ввода и вывода, так и специальные управляющие сигналы на шине управления. Адреса и данные как при обмене с памятью, так и при обмене с устройствами ввода/вывода передаются по одним и тем же шинам. Но если для обмена с памятью используются все 20 разрядов шины адреса (адресуется 1 Мбайт — адреса 00000 ... FFFFF), то в циклах обмена с устройствами ввода/вывода — только 16 разрядов шины адреса (адресуется 64 Кбайта — адреса 00000 ... 0FFFF). Такой подход имеет как свои преимущества (например, упрощение реализации прямого доступа к памяти), так и недостатки (усложнение системы команд, увеличение количества управляющих сигналов).

Микропроцессоры i8086/8088 выполнены в виде интегральной микросхемы в 40-выводном корпусе. Отличие в назначении выводов микросхемы между ними только одно: адрес в процессоре 8088 не мультиплексирован с данными (передается по отдельным линиям), а в процессоре 8086 — мультиплексирован.

Процессор работает от одного источника питания напряжением +5В и требует внешнего тактирующего сигнала с частотой, определяемой номером модели (от 4,77 МГц до 10 МГц).

Специальный управляющий сигнал MN/MX определяет минимальный или максимальный режим работы процессора. В минимальном режиме процессор сам вырабатывает сигналы управления для внешней шины. Этот режим используется для построения простейших систем. Для работы в составе компьютера применяется максимальный режим, при котором сигналы управления внешней шиной вырабатываются специальной микросхемой контроллера шины i8288.

7.2.2. Особенности процессора 80286

Несмотря на то, что процессор 80286 остался 16-разрядным, как и его предшественник 8086, он представлял собой новое поколение процессоров, что определило его высокую популярность и обеспечило персональному компьютеру на его основе (IBM PC AT) довольно долгую жизнь. Этот процессор отличается тем, что он имеет специальные средства для работы в многопользовательских и многозадачных системах.

Наиболее существенное отличие от процессора 8086/8088 — это механизм управления адресацией памяти, который обеспечивает четырехуровневую систему защиты и поддержку виртуальной памяти. (Виртуальная память — это внешняя память большого объема, с которой процессор может взаимодействовать как со своей системной памятью, но с некоторыми ограничениями). Специальные средства предусмотрены также для поддержки механизма переключения задач (Task switching). То

есть процессор способен выполнять несколько задач одновременно, переключаясь время от времени между ними. В процессоре 80286 также расширена система команд за счет добавления команд управления защитой и нескольких новых команд общего назначения.

Процессор 80286 может работать в двух режимах:

- Реальный режим (8086 Real Address Mode — режим реальной адресации), полностью совместимый с процессором 8086/8088. В этом режиме возможна адресация только в пределах 1 Мбайта физической памяти. Он используется для обеспечения программной преемственности с процессором 8086/8088.
- Защищенный режим (Protected Virtual Address Mode — защищенный режим виртуальной адресации). В этом режиме возможна адресация в пределах 16 Мбайт физической памяти. Такое решение связано с необходимостью построения компьютеров с большим объемом памяти, которые обеспечивали бы поддержку более сложных программ. В защищенном режиме система команд включает набор команд 8086, расширенный для обеспечения аппаратной поддержки многозадачного режима и виртуальной памяти.

Переключение в защищенный режим осуществляется одной командой (с предварительно подготовленными таблицами дескрипторов, описывающих параметры режима). Естественно, это довольно быстрый процесс. Обратное переключение в реальный режим гораздо сложнее: оно возможно только через аппаратный сброс процессора (по сигналу RESET), что требует гораздо больше времени.

В составе компьютера под управлением операционной системы MS DOS процессор 80286 работает в реальном режиме, а защищенный режим используют операционные системы типа UNIX, OS/2, NetWare286, а также операционные системы семейства MS Windows. Подробнее особенности этих режимов будут рассмотрены в следующем разделе.

Как и процессор 8086, 80286 имеет 16-разрядную внешнюю шину данных и 6-байтный конвейер команд. Однако быстродействие процессора 80286 при тактовой частоте 12,5 МГц примерно в 6 раз выше, чем у 8086 с тактовой частотой 5 МГц. Это достигается за счет усовершенствованной архитектуры и снижения количества тактов на одну команду. Для ускорения выполнения математических операций предусмотрено подключение к процессору 80286 микросхемы математического сопроцессора 80287.

Назначение внутренних регистров процессора 80286 такое же, как у процессора 8086/8088. Но в слове состояния процессора (PSW) добавлены три используемых разряда, и, кроме того, появился еще один внутренний регистр — регистр управления со словом состояния машины (MSW — Machine State Word), в котором используется только четыре бита (рис. 7.4).

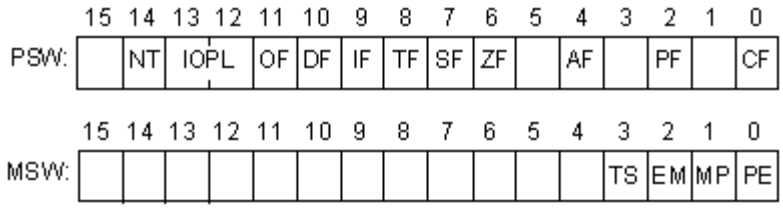

Рис. 7.4. Форматы регистров признаков (PSW) и управления (MSW) процессора 80286.

Дополнительные биты слова состояния процессора PSW имеют следующее назначение (подробнее о них — в следующем разделе):

- IOPL (Input/Output Privilege Level) — два бита, определяющие уровень привилегий ввода/вывода;
- NT (Nested Task flag) — флаг вложенной задачи.

Регистр MSW управляет режимом процессора. Для изменения его содержимого и сохранения его в памяти предназначены специальные команды. Назначение его битов следующее:

- PE (Protection Enable) — разрешение защиты. Установка этого флага переводит процессор в защищенный режим. Но очистка флага не переводит в реальный режим (требуется аппаратный сброс процессора).
- MP (Monitor Processor extension) — мониторинг внешнего математического сопроцессора.
- EM (Processor Extension Emulated) — эмуляция математического сопроцессора.
- TS (Task Switch) — переключение задач. Как и два предыдущих разряда, этот разряд управляет сопроцессором.

Сочетание MP=0 EM=0 TS=0, устанавливаемое по аппаратному сбросу (по сигналу RESET), обеспечивает совместимость с 8086/8088. Сочетание MP=1 EM=0 используется при совместном включении с сопроцессором 80287, а сочетание MP=0 EM=1 применяется в случае программной эмуляции сопроцессора, при которой функции сопроцессора выполняются основным процессором, но гораздо медленнее.

Остановимся подробнее на организации памяти процессора 80286. 24-разрядная внешняя шина адреса позволяет адресовать 16 Мбайт физической памяти, но в реальном режиме доступен только 1 Мбайт, начинающийся с нулевого адреса (000000…0FFFFF). Так же, как и в 8086, применяется сегментация памяти, но управление сегментацией в реальном и защищенном режимах различно.

В реальном режиме процессор 80286, в отличие от 8086, имеет средства контроля перехода через границу сегмента. Например, при попытке обращения к слову, имеющему смещение FFFF (его старший байт выходит за границу сегмента), или при выполнении команды, все байты которой не умещаются в данном сегменте, процессор вырабатывает специальное прерывание — Segment Overrun Interrupt.

В защищенном режиме существуют отличия от 8086, касающиеся определения сегментов:

- Сегментные регистры CS, DS, SS и ES хранят не сами базовые (начальные) адреса сегментов, а селекторы, определяющие адреса в памяти, по которым хранятся дескрипторы (описатели) сегментов. Область памяти с дескрипторами называется таблицей дескрипторов.
- Каждый дескриптор сегмента содержит базовый адрес сегмента, размер сегмента (от 1 до 64 Кбайт) и его атрибуты.
- Базовый адрес сегмента имеет разрядность 24 бит, что и обеспечивает адресацию 16 Мбайт физической памяти.
- Селекторы, загружаемые в 16-битные сегментные регистры, имеют три поля: RPL (Requested Privilege Level) — запрашиваемый уровень привилегий, TI (Table Indicator) — индикатор использования локальной или глобальной таблицы дескрипторов, INDEX — номер дескриптора в таблице (формат показан на рис. 7.5).

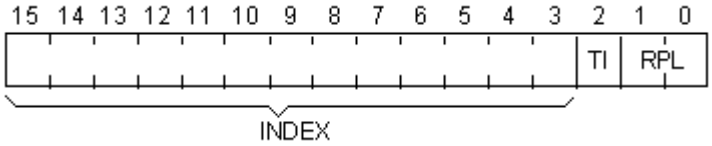

Рис. 7.5. Формат селектора сегмента процессора 80286.

О порядке вычисления адреса памяти в защищенном режиме процессора 80286 уже говорилось в разделе 3.1.2 (см. рис.3.7). На сумматор, вычисляющий физический адрес памяти, подается не содержимое сегментного регистра, а базовый адрес сегмента из таблицы дескрипторов.

Дескрипторы хранятся в памяти и занимают по четыре смежных 16-разрядных слова (то есть 8 байт). При загрузке нового значения селектора дескрипторы считываются из памяти и сохраняются во внутренних программно недоступных (и невидимых) регистрах процессора. До смены значения селектора при обращениях к памяти используются значения дескрипторов только из этих невидимых регистров (их называют кэш-регистрами).

В защищенном режиме команды ввода/вывода процессора являются привилегированными. Это означает, что они могут выполняться задачами только с определенным уровнем привилегий, определяе-

мых полем IOPL регистра признаков. Несанкционированная попытка выполнения этих команд вызывает прерывание по нарушению защиты.

Как и 8086, процессор 80286 может обрабатывать до 256 типов прерываний. Прерывания подразделяются на аппаратные (маскируемые или немаскируемые), вызываемые сигналами на входах процессора, программные, вызываемые командой INT, и **исключения инструкций**. При этом аппаратные и программные прерывания работают точно так же, как в 8086/8088.

Исключения инструкций (Instruction Exceptions) или просто исключения случаются при возникновении особых условий при выполнении операций (в 8086 аналогом исключений являлись внутренние прерывания процессора). Обработка исключений проводится аналогично обработке прерываний.

Каждому номеру прерывания соответствует свой элемент в таблице дескрипторов прерываний IDT (Interrupt Descriptor Table). В реальном режиме эта таблица организована так же, как у 8086/8088, то есть содержит двойные слова, определяющие адрес начала процедур обработки прерываний. В защищенном режиме таблица содержит 8-байтные дескрипторы прерываний. Ее размер может быть от 32 до 256 дескрипторов, и располагаться она может в любом месте памяти.

Система команд процессора 80286 включает, помимо полного набора8086, ряд дополнительных команд, например:

- сохранение константы в стеке, сохранение в стеке и восстановление из стека всех регистров одной командой;
- целочисленное умножение на константу;
- сдвиги (включая циклические) на заданное в константе количество шагов;
- вход и выход из процедур;
- команды управления защитой.

Попытка выполнения недействительной команды (или попытка выполнения в реальном режиме команды, предназначенной только для защищенного режима) вызывает специальное исключение.

Процессор 80286 выпускался в 68-выводных корпусах. Внешние шины адреса и данных были разделены. Напряжение питания процессора составляет +5В.

7.2.3. Особенности процессора 80386

32-разрядный процессор i80386 открыл новый этап в истории микропроцессоров Intel и персональных компьютеров типа IBM PC. Естественно, он сохранял полную совместимость со своими 16-разрядными предшественниками, чтобы не отказываться от разработанного для них программного обеспечения. Но именно в 80386 преодолено жесткое ограничение на длину непрерывного сегмента памяти в 64 Кбайт, что являлось пережитком прошлого и следствием не самых удачных архитектурных решений 8086.

В защищенном режиме 80386 длина сегмента может достигать 4 Гбайт, то есть всего объема физически адресуемой памяти. Таким образом, память фактически стала непрерывной. Кроме того, 80386 обеспечивает поддержку виртуальной памяти объемом до 64 Тбайт (1 Тбайт = 1024 Гбайт). Встроенный блок управления памятью поддерживает механизмы сегментации и страничной трансляции адресов (Paging). Обеспечивается четырехуровневая система защиты памяти и ввода/вывода, а также переключение задач.

Процессор 80386, как и 80286, может работать в двух режимах:

- Реальный режим, который полностью совместим с 8086.
- Защищенный режим. В этом режиме возможна адресация до 4 Гбайт физической памяти (32 разряда), через которые при использовании механизма страничной адресации может отображаться до 16 Тбайт виртуальной памяти каждой задачи.

Переключение между этими двумя режимами в обе стороны, в отличие от 80286, производится достаточно быстро, с помощью простой последовательности команд, и аппаратного сброса процессора не требуется.

Процессор может оперировать с 8, 16, 32-битными операндами, строками байт, слов и двойных слов, а также с битами, битовыми полями и строками бит.

В архитектуру процессора введены средства отладки и тестирования.

Разрядность регистров данных (AX, BX, CX, DX) и адресов (SI, DI, BP, SP) увеличена до 32. При этом в их обозначении появилась приставка E (Extended — расширенный), например, EAX, ESI. Отсутствие приставки в имени означает ссылку на младшие 16 разрядов соответствующего регистра. Регистры данных и адресов объединены в группу регистров общего назначения, которые иногда могут заменять друг друга. Это может рассматриваться как отход от идеологии специализации всех регистров.

Регистры селекторов оставлены 16-разрядными, но добавлено два новых регистра FS и GS для задания дополнительных сегментов данных. Также расширен до 32 разрядов регистр-указатель (счетчик) команд EIP. 32-разрядным стал и регистр флагов EFLAGS. Его биты, определенные для 8086 и 80286, остались прежними, но добавлены 6 новых бит (рис. 7.6). Такой же формат используется и в процессорах 80486 и Pentium.

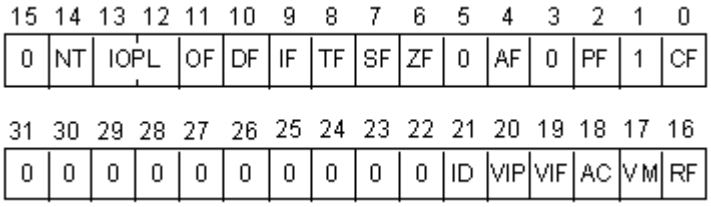

Рис. 7.6. Регистр признаков EFLAGS процессора 80386.

Регистры сегментов процессора содержат 16-битные указатели (в реальном режиме) или селекторы (в защищенном режиме) шести сегментов. С каждым из шести сегментных регистров связаны программно недоступные регистры дескрипторов, как и в случае 80286. В защищенном режиме в регистры дескрипторов загружается 32-битный базовый адрес сегмента, 32-битный лимит и атрибуты сегментов.

Помимо упомянутых регистров в состав процессора входят еще четыре управляющих регистра (CR0, CR1, CR2, CR3), которые хранят признаки состояния процессора, общие для всех задач. В процессоре Pentium к ним добавлен еще и регистр CR4. Кроме того, процессор содержит еще системные адресные регистры для ссылок на сегменты и таблицы в защищенном режиме, регистры отладки и регистры тестирования. Как видим, от модели к модели количество регистров процессора постоянно возрастает.

Процессор позволяет выделять в памяти сегменты и страницы. Сегменты в реальном режиме имеют фиксированный размер, в защищенном — переменный. Страницы, которых не было в предыдущих моделях, представляют собой области логической памяти размером 4 Кбайт, каждая из которых может отображаться на любую область физической памяти. Если сегменты используются на прикладном уровне, то страницы применяются на системном.

Применительно к памяти различают три адресных пространства: логическое, линейное и физическое. О принципах адресации памяти у процессора 80386 уже говорилось в разделе 3.1.2 (см. рис. 3.8).

Процессор 80386 может использовать режимы 32-битной или 16-битной адресации. Режим 16-битной адресации соответствует режимам процессоров 8086 и 80286, при этом в качестве компонентов адреса используются младшие 16 бит соответствующих регистров. Отличие 32-битной адресации отражено в табл. 7.1.

Таблица 7.1. Различия режимов адресации 80386.		
Компонент адресация	**16-битная адресация**	**32-битная**
Базовый регистр	BX или BP	Любой 32-битный общего назначения
Индексный регистр	SI или DI	Любой 32-битный общего назначения, кроме ESP
Масштаб	Нет (всегда 1)	1,2, 4 или 8
Смещение	0, 8 или 16 бит	0, 8 или 32 бита

Так же, как и предшественники, процессор 80386 обрабатывает все виды прерывания: аппаратные (маскируемые и немаскируемые) и программные, которые в данном случае обрабатываются как разновидность исключений, и собственно исключения. Исключения подразделяются на отказы, ловушки и аварийные завершения.

- **Отказ (Fault)** — это исключение, которое обнаруживается и обслуживается до выполнения команды, вызывающей ошибку.
- **Ловушка (Trap)** — это исключение, которое обнаруживается и обслуживается после выполнения команды, вызывающей это исключение. К классу ловушек относятся и программные прерывания.
- **Аварийное завершение (Abort)** — это исключение, которое не позволяет точно установить команду, вызвавшую его. Оно используется для сообщения о серьезной ошибке, такой как аппаратная ошибка или повреждение системных таблиц.

Каждому номеру прерывания (0…255) или исключения соответствует элемент в таблице дескрипторов прерываний IDT (Interrupt Descriptor Table). В защищенном режиме IDT может иметь размер от 32 до 256 дескрипторов, каждый из которых состоит из 8 байт.

Отличия от предшествующего процессора 80286 в выполнении операций ввода/вывода сводятся к добавлению возможностей обращения к 32-битным портам. Важно отметить, что строковые команды процессора 80386 обеспечивают блочный ввод/вывод с большей скоростью, чем стандартный контроллер прямого доступа к памяти.

Процессор выпускался в 100-выводном корпусе. Была предусмотрена возможность подключения внешнего сопроцессора 80387.

А теперь остановимся чуть подробнее на защищенном режиме, который используется на полную мощность именно начиная с 32-разрядных процессоров, так как процессор 80286 имел существенные ограничения и в большинстве случаев работал все-таки в реальном режиме.

Защищенный режим был предложен для обеспечения независимости одновременного выполнения нескольких задач (как системных, так и прикладных). Для этого предусмотрена защита ресурсов каждой задачи от действий других задач. Под ресурсами здесь понимается память с данными, программами, системными таблицами, а также используемая задачей аппаратура. Защита основывается на сегментации памяти, причем, в отличие от реального режима, задача не может переопределять положения своих сегментов в памяти и использует только сегменты, определенные для нее операционной системой. Сегмент определяется дескриптором сегмента, который задает положение сегмента в памяти, его размер (или лимит), назначение и характеристики защиты.

Защита с помощью сегментации не позволяет:

- использовать сегменты не по назначению, например, трактовать область данных как область программы;

- нарушать права доступа (например, пытаться записывать информацию в сегмент, предназначенный только для чтения, или обращаться к сегменту, не имея достаточных привилегий);
- адресоваться к элементам, выходящим за границу сегмента;
- изменять дескрипторы сегментов, не имея достаточных привилегий.

Защищенный режим предусматривает средства переключения задач. Состояние каждой задачи (то есть состояние всех регистров процессора) хранится в специальном сегменте состояния задачи, на который указывает селектор в регистре задачи. При переключении задачи достаточно загрузить в регистр задачи новый селектор, и состояние предыдущей задачи автоматически сохранится, а в процессор загрузится состояние новой (или ранее прерванной) задачи. Это развитие идеи стека.

В защищенном режиме предусматривается иерархическая четырехуровневая (уровни 0, 1, 2, 3) система привилегий, предназначенная для управления выполнением привилегированных команд и доступом к дескрипторам (рис. 7.7). Уровень 0 соответствует неограниченным правам доступа и отводится ядру операционной системы. Уровень 3 дает минимальные права и отводится прикладным задачам. Уровни привилегий относятся к дескрипторам, селекторам и задачам. Кроме того, в регистре флагов имеется двухбитовое поле привилегий ввода/вывода (см. рис. 7.4 и 7.6), управляющее доступом к командам ввода/вывода и флагом прерываний.

Механизм виртуальной памяти, используемый в защищенном режиме, позволяет любой задаче использовать логическое пространство размером до 64 Тбайт (16К сегментов по 4 Гбайта). Для этого каждый сегмент в своем дескрипторе имеет специальный бит, указывающий на присутствие данного сегмента в оперативной памяти в текущий момент. Неиспользуемый сегмент может быть выгружен из оперативной памяти во внешнюю память (обычно — на диск), о чем делается пометка в его дескрипторе. На освободившееся место из внешней памяти может закачиваться другой сегмент (это называется свопингом или подкачкой). При обращении задачи к отсутствующему в оперативной памяти сегменту вырабатывается специальное исключение, которое и выполняет свопинг. С точки зрения выполняемой программы, виртуальная память ничем не отличается от реальной (говорят, что виртуальная память прозрачна), не считая задержки на процесс перекачки информации на диск и с диска.

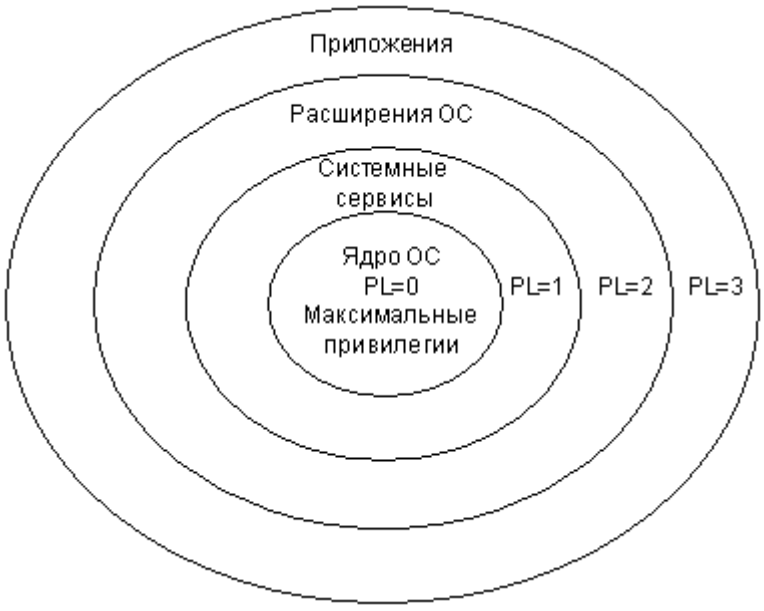

Рис. 7.7. Уровни привилегий 32-разрядных процессоров.

Реальное использование системы защиты и виртуальной памяти возлагается на операционную систему, которая в идеале должна обеспечивать работоспособность даже в случае некорректного выполнения прикладных задач.

В памяти существует три типа таблиц дескрипторов: локальная таблица дескрипторов LDT, глобальная таблица дескрипторов GDT и таблица дескрипторов прерываний IDT. Каждой таблице соответствует свой регистр процессора (соответственно, LDTR, GDTR и IDTR), где хранятся дескрипторы сегментов. Глобальная таблица содержит дескрипторы, доступные всем задачам, а локальная может

быть для каждой задачи своя. Дескрипторы состоят из 8 байтов (как и у 80286). Однако назначение байтов различно. Для примера на рис. 7.8 показаны форматы дескрипторов сегмента программ и данных процессоров 80286 и 80386.

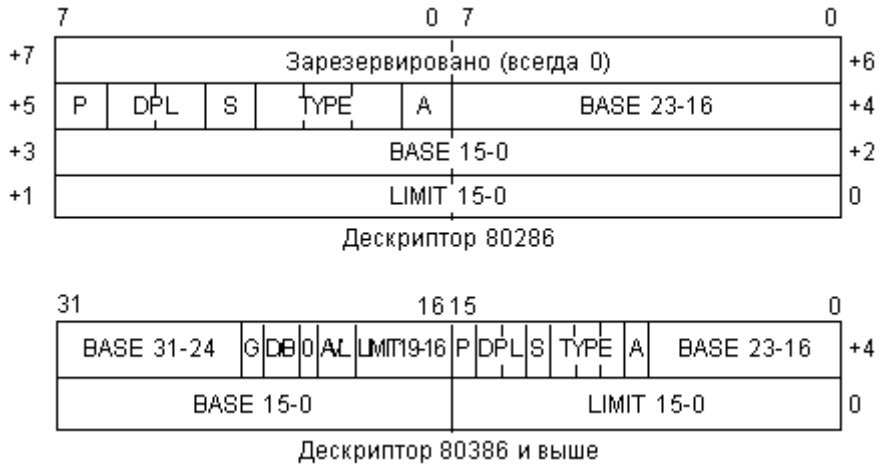

Рис. 7.8. Дескрипторы сегментов программы и данных.

Существуют также системные сегменты, предназначенные для хранения локальных таблиц дескрипторов и таблиц состояния задач. Их дескрипторы (тоже 8-байтные) определяют базовый адрес, лимит сегмента, права доступа (чтение, чтение/запись, только исполнение или исполнение/чтение) и присутствие сегмента в оперативной памяти.

Задачи, дескрипторы и селекторы имеют свои уровни привилегий. Привилегии задач действуют на выполнение команд и использование дескрипторов. Текущий уровень привилегии задач определяется двумя младшими битами регистра CS. Привилегии дескриптора описываются полем DPL (рис. 7.8). DPL определяет наименьший уровень привилегий, с которым возможен доступ к данному дескриптору. Привилегии селектора задаются полем RPL (см. рис. 7.5). Привилегии проверяются при попытках записи в сегментные регистры, а также при выполнении некоторых команд.

Таким образом, начиная с процессора 80386, появляются средства обслуживания многозадачного режима. Естественно, процессор не может обрабатывать несколько задач одновременно, выполняя по несколько команд сразу. Он только периодически переключается между задачами. Но с точки зрения пользователя получается, что компьютер параллельно работает с несколькими задачами.

7.2.4. Особенности процессора 486

Процессор 486 является представителем второго поколения 32-разрядных процессоров. Он сохраняет основные принципы архитектуры процессора 80386, а также обеспечивает полную совместимость со своими предшественниками. Но в то же время он имеет ряд преимуществ.

- В процессор введена внутренняя кэш-память 1-го уровня (Internal cache Level 1) размером 8 Кбайт и предусмотрены средства для двухуровневого кэширования.
- В процессор введен математический сопроцессор (в модели процессора 486SX сопроцессор отсутствует).
- Повышена производительность обмена по внешней шине — введены так называемые пакетные циклы, передающие одно слово за один такт шины.
- В архитектуре процессора применено скоростное RISC-ядро, которое позволяет наиболее часто встречающиеся команды выполнять за один такт.
- В структуру введены буферы отложенной записи.
- В отдельных моделях предусмотрено внутреннее умножение тактовой частоты (на 2, 2,5 или 3).

Все это обеспечило существенное увеличение быстродействия. А усовершенствованный защищенный режим дает некоторые дополнительные возможности.

Рассмотрим подробнее принцип действия кэш-памяти.

Кэш-память (или просто кэш, от англ. Cache — склад, тайник) предназначена для промежуточного хранения информации из системной памяти с целью ускорения доступа к ней. Ускорение достигается за счет использования более быстрой памяти и более быстрого доступа к ней. При этом в кэш-памяти хранится постоянно обновляемая копия некоторой области основной памяти.

Необходимость введения кэша связана с тем, что системная память персонального компьютера выполняется на микросхемах динамической памяти, которая характеризуется меньшей стоимостью, но и более низким быстродействием, по сравнению со статической памятью. Идея состоит в том, что благодаря введению быстрой буферной, промежуточной статической памяти можно ускорить обмен с медленной динамической памятью. По сути, кэш-память делает то же, что и применявшийся ранее конвейер команд, но на более высоком уровне. В кэш-памяти хранится копия некоторой части системной памяти, и процессор может обмениваться с этой частью памяти гораздо быстрее, чем с системной памятью. Причем в кэш-памяти могут храниться как команды, так и данные.

Выигрыш в быстродействии от применения кэша связан с тем, что процессор в большинстве случаев обращается к адресам памяти, расположенным последовательно, один за другим, или же близко друг к другу. Поэтому высока вероятность того, что информация из этих адресов памяти окажется внутри небольшой кэш-памяти. Если же процессор обращается к адресу, расположенному далеко от тех, к которым он обращался ранее, кэш оказывается бесполезным и требует перезагрузки, что может даже замедлить обмен по сравнению со структурой без кэш-памяти.

В принципе кэш-память может быть как внутренней (входить в состав процессора), так и внешней. Внутренний кэш называется кэшем первого уровня, внешний — кэшем второго уровня. Объем внутреннего кэша обычно невелик — типовое значение 32 Кбайт. Объем внешнего кэша может достигать нескольких мегабайт. Но принцип функционирования у них один и тот же.

Кэш первого уровня процессора 486 имеет четырехканальную структуру (рис. 7.9). Каждый канал состоит из 128 строк по 16 байт в каждой. Одноименные строки всех четырех каналов образуют 128 наборов из четырех строк, каждый из которых обслуживает свои адреса памяти. Каждой строке соответствует 21-разрядная информация об адресе скопированного в нее блока системной памяти. Эта информация называется **тегом** (Tag) строки.

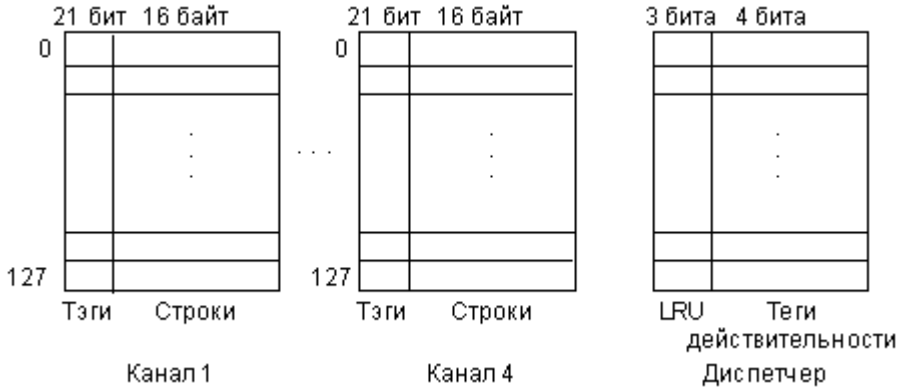

Рис. 7.9. Структура внутреннего кэша процессора 486.

Кроме того, в состав кэша входит так называемый диспетчер, то есть область памяти с организацией 128 x 7, в которой хранятся 4-битные теги действительности (достоверности) для каждого из 128 наборов и 3-битные коды LRU (Least Recently Used) для каждого из 128 наборов. Тег действительности набора включает в себя 4 бита достоверности каждой из 4 строк, входящих в данный набор. Бит достоверности, установленный в единицу, говорит о том, что соответствующая строка заполнена; если он сброшен в нуль, то строка пуста. Биты LRU говорят о том, как давно было обращение к данному набору. Это нужно для того, чтобы обновлять наименее используемые наборы.

Адресация кэш-памяти осуществляется с помощью 28 разрядов адреса. Из них 7 младших разрядов выбирают один из 128 наборов, а 21 старший разряд сравнивается с тегами всех 4 строк выбранного

набора. Если теги совпадают с разрядами адреса, то получается ситуация **кэш-попадания**, а если нет, то ситуация **кэш-промаха**.

В случае цикла чтения при кэш-попадании байт или слово читаются из кэш-памяти. При кэш-промахе происходит обновление (перезагрузка) одной из строк кэш-памяти.

В случае цикла записи при кэш-попадании производится запись как в кэш-память, так и в основную системную память. При кэш-промахе запись производится только в системную память, а обновление строки кэш-памяти не производится. Эта строка становится недостоверной (ее бит достоверности сбрасывается в нуль).

Такая политика записи называется **сквозной** или **прямой записью** (Write Through). В более поздних моделях процессоров применяется и **обратная запись** (Write Back), которая является более быстрой, так как требует гораздо меньшего числа обращений по внешней шине.

При использовании обратной записи в основную память записываемая информация отправляется только в том случае, когда нужной строки в кэше нет. В случае же попадания модифицируется только кэш. В основную память измененная информация попадет только при перезаписи новой строки в кэш. Прежняя строка при этом целиком переписывается в основную память, и тем самым восстанавливается идентичность содержимого кэша и основной памяти.

В случае, когда требуемая строка в кэше не представлена (ситуация кэш-промаха), запрос на запись направляется на внешнюю шину, а запрос на чтение обрабатывается несколько сложнее. Если этот запрос относится к кэшируемой области памяти, то выполняется цикл заполнения целой строки кэша (16 байт из памяти переписывается в одну из строк набора, обслуживающего данный адрес). Если затребованные данные не укладываются в одной строке, то заполняется и соседняя строка. Заполнение строки процессор старается выполнить самым быстрым способом — пакетным циклом, однако внешний контроллер памяти может потребовать использования более медленных пересылок.

Внутренний запрос процессора на данные удовлетворяется сразу, как только данные считываются из памяти, а дальнейшее заполнение строки может идти параллельно с обработкой данных. Если в наборе, который обслуживает данный адрес памяти, имеется свободная строка, заполнена будет именно она. Если же свободных строк нет, заполняется строка, к которой дольше всех не обращались. Для этого используются биты LRU, которые модифицируются при каждом обращении к строке данного набора.

Кроме того, существует возможность аннулирования строк (объявления их недостоверными) и очистки всей кэш-памяти. При сквозной записи очистка кэша проводится специальным внешним сигналом процессора, программным образом с помощью специальных команд, а также при начальном сбросе – по сигналу RESET. При обратной записи очистка кэша подразумевает также выгрузку всех модифицированных строк в основную память.

Отметим, что в пространстве памяти персонального компьютера имеются области, для которых кэширование принципиально недопустимо (например, разделяемая память аппаратурных адаптеров — плат расширения).

Режим пакетной передачи (Burst Mode), впервые появившийся в процессоре 486, предназначен для быстрых операций со строками кэша. Пакетный цикл обмена (Burst Cycle) отличается тем, что для пересылки всего пакета адрес по внешней шине адреса передается только один раз — в начале пакета, а затем в каждом следующем такте передаются только данные. Адрес для каждого следующего кода данных вычисляется из начального адреса по правилам, установленным как передатчиком данных, так и их приемником. Например, адрес каждого следующего слова данных вычисляется как инкрементированный адрес предыдущего. В результате время передачи одного слова данных значительно сокращается. Понятно, что обмен пакетными циклами возможен только с устройствами, изначально способными обслуживать такой цикл. Допустимая длина пакета не слишком велика, например, при чтении размер пакета ограничен одной строкой кэша.

Режим внутреннего умножения тактовой частоты процессора был предложен для того, чтобы повысить быстродействие процессора, но при этом устанавливать его в системные платы, рассчитанные на невысокие тактовые частоты. Например, модель процессора 486DX2-66 работает в системной плате с тактовой частотой 33, но эту частоту внутри себя преобразует в удвоенную частоту — 66 МГц. Это позволяет уменьшить общую стоимость системы, так как снижает требования к элементам системной платы.

Процессор 486 выпускался в 168- или 169-выводных корпусах. Напряжение питания — 5 В или 3,3 В. Введение пониженного напряжения питания 3,3 В связано с необходимостью снижения величины рассеиваемой мощности. Растущая тактовая частота и усложнение структуры процессоров приводят к тому, что рассеиваемая ими мощность достигает нескольких ватт. Для современных процессоров уже обязательно применение вентиляторов на корпусе процессора.

7.2.5. Особенности процессоров Pentium

Процессоры Pentium относятся к пятому поколению процессоров или к третьему поколению 32-разрядных процессоров. По своим основным архитектурным принципам они совместимы с процессорами 386 и 486. Но имеются существенные отличия, позволяющие говорить о новом поколении:

- Суперскалярная архитектура процессора, то есть процессор имеет два пятиступенчатых параллельно работающих конвейера обработки информации, благодаря чему он способен одновременно выполнять две команды за один такт. Необходимо отметить, что преимущества такой архитектуры проявляются только в случае специальной компиляции программного обеспечения, позволяющей осуществлять параллельную обработку.
- Внешняя 64-разрядная шина данных для повышения производительности. Это требует соответствующей организации памяти. Из-за такой особенности процессор иногда неправильно называют 64-разрядным (хотя внутри он все-таки остался 32-разрядным). Внешняя шина адреса процессора — 32-разрядная.
- Применение технологии динамического предсказания ветвлений (переходов).
- Раздельный кэш для команд и данных объемом 8 Кбайт каждый. Длина строки кэша — 32 байта. Оба кэша работают в режиме обратной записи.
- Повышенная в 2—10 раз по сравнению с процессором 486 производительность встроенного математического сопроцессора. В нем применена восьмиступенчатая конвейеризация и специальные блоки сложения, умножения и деления, что позволяет выполнять операции с плавающей точкой за один такт процессора.
- Сокращено время (число тактов) выполнения команд.
- Предусмотрена возможность построения двухпроцессорных систем.
- Введены средства управления энергопотреблением и тестирования.

Предсказание ветвлений позволяет продолжать выборку и декодирование потока команд после выборки команды ветвления (перехода), не дожидаясь проверки условия перехода. В прежних моделях в данном случае приходилось приостанавливать конвейер. Динамическое предсказание основывается на анализе предыдущей программы и накапливании статистики поведения. Исходя из этого анализа предсказывается наиболее вероятное условие каждого встречающегося в программе перехода.

В дополнение к базовой архитектуре 32-разрядных процессоров Pentium имеет набор регистров MSR (Model Specific Registers). В него входит группа тестовых регистров (TR1 — TR12), средства слежения за производительностью, регистры-фиксаторы адреса и данных цикла, вызвавшего срабатывание контроля машинной ошибки. Название этой группы регистров указывает на их уникальность для каждой модели процессоров Pentium.

Средства для слежения за производительностью (мониторинга производительности) включают в себя таймер реального времени и счетчики событий. Таймер представляет собой 64-разрядный счетчик, инкрементируемый с каждым тактом процессора. Два счетчика событий имеют разрядность по 40 бит и программируются на подсчет событий различных классов, связанных с шинными операциями, исполнением команд, связанных с работой кэша, контролем точек останова и т.д. Сравнивая состояния таймера и счетчиков событий, можно сделать вывод о производительности процессора.

Тестовые регистры позволяют управлять большинством функциональных узлов процессора, обеспечивая возможность подробного тестирования их работоспособности. Специальные биты регистра TR12 позволяют отключить новые архитектурные свойства (предсказание и трассировку ветвлений, параллельное выполнение инструкций), а также работу первичного кэша.

Средства для построения двухпроцессорных систем позволяют на одной локальной шине устанавливать два процессора с объединением почти всех одноименных выводов. Это дает возможность использовать симметричную мультипроцессорную обработку (SMP — Symmetric Multi-Processing) или строить функционально избыточные системы (FRC — Functional Redundancy Checking).

В режиме SMP каждый процессор выполняет свою задачу, порученную ему операционной системой, что поддерживается такими системами, как Novell NetWare, Windows NT, Unix. Оба процессора разделяют общие ресурсы компьютера, включая память и устройства ввода/вывода. В каждый момент шиной управляет один процессор, по определенным правилам они меняются ролями. В идеальном случае производительность системы увеличивается вдвое (без учета обращений к шине и времени на переключение процессоров).

В конфигурации FRC два процессора выступают как один логический процессор. Основной процессор (Master) работает в обычном однопроцессорном режиме. Проверочный процессор (Checker) выполняет все те же операции внутри себя, не управляя внешней шиной, и сравнивает сигналы основного процессора с теми, которые генерирует сам. В случае несовпадения формируется сигнал ошибки, обрабатываемый как прерывание. То есть в данном случае увеличивается надежность системы в целом (в идеале — вдвое).

Развитием Pentium стало добавление технологии MMX, рассчитанной на мультимедийное, графическое и коммуникационное применение. Основная идея MMX заключается в одновременной обработке нескольких элементов данных за одну команду (SIMD, Single Instruction — Mutiple Data). Расширение MMX использует новые типы упакованных 64-битных данных:

- упакованные байты — восемь байт;
- упакованные слова — четыре слова;
- упакованные двойные слова (два двойных слова);
- учетверенное слово (одно слово).

Эти типы данных могут обрабатываться в восьми дополнительных 64-разрядных регистрах MMX0 — MMX7. В систему команд для поддержки MMX введено 57 дополнительных команд для одновременной обработки нескольких единиц данных (команды пересылки, арифметические, логические команды и команды преобразования форматов данных). Команды MMX доступны из любого режима процессора.

Кроме того, в процессорах Pentium MMX увеличен объем кэша данных и программ (до 16 Кбайт каждый), увеличено число ступеней конвейеров и введено еще несколько усовершенствований, повышающих производительность обычных (не мультимедийных) операций.

От процессора Pentium Pro принято отсчитывать шестое поколение процессоров. Pentium Pro по сравнению с Pentium имеет следующие усовершенствования:

- Динамическое исполнение команд предполагает, что команды, не зависящие от результатов предыдущих операций, могут выполняться в измененном порядке (последующие раньше предыдущих), однако последовательность обмена с внешними устройствами (памятью и устройствами ввода/вывода) будет соответствовать программе. То есть процессор сам выбирает удобный ему порядок выполнения команд. Это позволяет повысить производительность процессора без увеличения тактовой частоты.
- Архитектура двойной независимой шины повышает суммарную пропускную способность. Одна шина (системная) служит для обмена с основной памятью и устройствами ввода/вывода, а другая (локальная) предназначена только для обмена с вторичным кэшем (см. рис. 7.3).
- В процессор введен кэш второго уровня объемом 256—512 Кбайт.
- Возможно построение многопроцессорных систем (до четырех микропроцессоров).

Процессор Pentium II сочетает в себе архитектуру Pentium Pro с технологией MMX.

Размер первичных кэшей данных и команд составляет 16 Кбайт, вторичного кэша — до 512 Кбайт. Кэш второго уровня несколько медленнее, чем кэш первого уровня, зато он имеет больший объем. Такая двухуровневая организация позволяет достигать компромисса между быстродействием кэш-памяти и ее объемом.

Шина адреса имеет 36 разрядов (то есть максимально допустимый объем памяти — 64 Гбайта).

Процессоры Pentium III и Pentium 4 отличаются значительно возросшей предельной тактовой частотой (до 3 ГГц у Pentium 4, а в перспективе и до 5 ГГц), увеличенным объемом кэша (от 512 Кбайт до нескольких мегабайт) и дальнейшим совершенствованием архитектуры Pentium. Размер внутреннего конвейера у Pentium 4 доведен до 20 ступеней.

Уже появились и полностью 64-разрядные процессоры. Правда, полное использование возможностей 64-разрядной архитектуры потребует существенного изменения программного обеспечения.

Надо отметить, что в составе персональных компьютеров практически никогда не используются все возможности процессоров семейства Pentium. Например, многопроцессорные системы встречаются достаточно редко, а объем системной памяти лишь иногда превышает 512 Мбайт.

Для портативных компьютеров были предложены упрощенные версии процессоров Pentium III и Pentium 4, продаваемые под маркой Celeron. Они отличаются уменьшенной тактовой частотой и сокращенным объемом кэша второго уровня. Их тактовая частота сейчас доходит до 2 ГГц. Надо учитывать, что рассеиваемая мощность процессора Celeron остается практически такой же, как у процессоров Pentium III и Pentium 4.