View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Belarusian State Technological University

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

А. С. Кобайло, Н. А. Жиляк

АРИФМЕТИКО-ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН И АРХИТЕКТУРА КОМПЬЮТЕРОВ. ВВЕДЕНИЕ В АРХИТЕКТУРУ КОМПЬЮТЕРНЫХ СИСТЕМ

Учебно-методическое пособие по одноименной дисциплине для студентов IT-специальностей

Минск 2015

УДК 004.2(075.8)
ББК 32.97.я73
К55

Рассмотрено и рекомендовано к изданию редакционно-издательским советом Белорусского государственного технологического университета.

Рецензенты:
доцент кафедры информационных технологий автоматизированных систем БГУИР кандидат технических наук, доцент *О. В. Герман*;
доцент кафедры информатики и компьютерной графики БГТУ кандидат технических наук, доцент *А. А. Дятко*

Кобайло, А. С.

К55 Арифметико-логические основы цифровых вычислительных машин и архитектура компьютеров. Введение в архитектуру компьютерных систем : учеб.-метод. пособие по одноименной дисциплине для студентов IT-специальностей / А. С. Кобайло, Н. А. Жиляк. – Минск : БГТУ, 2015. – 64 с.

Учебно-методическое пособие содержит основные положения по архитектуре микропроцессоров и организации компьютерных систем, методам распараллеливания вычислительных процессов и основанных на них принципах организации параллельных вычислительных систем.

Издание предназначено для студентов, изучающих дисциплины «Арифметико-логические основы цифровых вычислительных машин и архитектура компьютеров», «Компьютерные системы и сети», а также может быть полезно студентам, магистрантам и аспирантам, изучающим организацию вычислительных систем и программирование на языках нижнего уровня (в частности, Ассемблере).

УДК 004.434 (075.8)
ББК 32.97я73

© УО «Белорусский государственный технологический университет, 2015
© Кобайло А. С., Жиляк Н. А., 2015

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Учебно-методическое пособие по дисциплине «Арифметико-логические основы цифровых вычислительных машин и архитектура компьютеров» предназначено для обучения студентов по одноименной дисциплине, а также дисциплине «Компьютерные системы и сети».

Отметим, что задачи, связанные с применением арифметических и логических основ ЦВМ, возникают в самых различных сферах информационных технологий. Сюда можно отнести системы автоматизированного проектирования (САПР), в которых осуществляется синтез цифровых узлов вычислительных систем на базе законов алгебры логики и логико-комбинаторного подхода к синтезу структуры сложных объектов, оптимизация синтезированных вариантов объекта, выбор из множества альтернативных вариантов структуры проектируемого объекта оптимальных. Широко используются методы, основанные на алгебре логики, при синтезе элементов и устройств вычислительной техники различного назначения, синтезе математических моделей для информационных систем, синтезе алгоритмов. Важную роль эти методы играют при проектировании систем управления и обработки информации, встроенных систем, других сложных технических систем, в том числе и издательско-полиграфических комплексов. Логичным продолжением арифметико-логических основ ЦВМ являются разделы по архитектурной организации компьютерных систем. Материалы по данной тематике содержатся в фундаментальных изданиях отечественных и зарубежных авторов, однако, большинство из этих литературных источников стало библиографической редкостью или является недоступным широкому кругу обучающихся. В данной связи издание пособий по арифметико-логическим основам ЦВМ и архитектуры компьютерных систем является актуальным.

Первые разделы курса, вынесенного в название данного учебно-методического пособия, изложены в электронном средстве обучения Кобайло А. С. «Арифметические и логические основы цифровых вычислительных машин: учебно-методическое пособие для студентов заочной формы обучения специальности «Информационные системы и технологии (издательско-полиграфический комплекс)», изданного в 2014 г.; схемотехнические основы ЦВМ как цифровой логический уровень компьютера рассматриваются в конспекте лекций Кобайло А. С. по дисциплине «Арифметико-логические основы цифровых

вычислительных машин и архитектура компьютеров». В данном учебно-методическом пособии содержатся основные сведения по важнейшим разделам соответствующих дисциплин, не вошедшим в указанные издания.

Пособие состоит из двух разделов: организация микропроцессорной системы; параллельные компьютерные системы.

В первом разделе рассматриваются вопросы организации компьютерных систем, архитектура ряда современных микропроцессоров, форматы и методы передачи данных в микропроцессорных системах, построение интерфейсов микропроцессорных систем, организация прерываний и прямого доступа к памяти.

Во втором разделе рассматриваются такие вопросы, как классификация параллельных вычислительных систем (ВС), различные типы архитектур, организация памяти, способы комплексирования в ВС.

Содержание пособия соответствует важнейшим разделам учебных программ учреждения высшего образования по указанной учебной дисциплине «Арифметико-логические основы цифровых вычислительных машин и архитектура компьютеров» для студентов специальностей 1-40 05 01-03 «Информационные системы и технологии (издательско-полиграфический комплекс)» и 1-98 01 03 «Программное обеспечение безопасности мобильных систем» и учебной дисциплине «Компьютерные системы и сети» для студентов специальности 1-40 01 01 «Программное обеспечение информационных технологий». Содержание пособия полностью соответствует содержанию одноименных тем учебной программы

Изучение студентами материала пособия будет способствовать углублению и расширению ими знаний по архитектурной организации компьютерных систем и окажет несомненную помощь при изучении практически всех специальных дисциплин согласно учебным планам соответствующих специальностей.

ГЛАВА 1. ОРГАНИЗАЦИЯ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

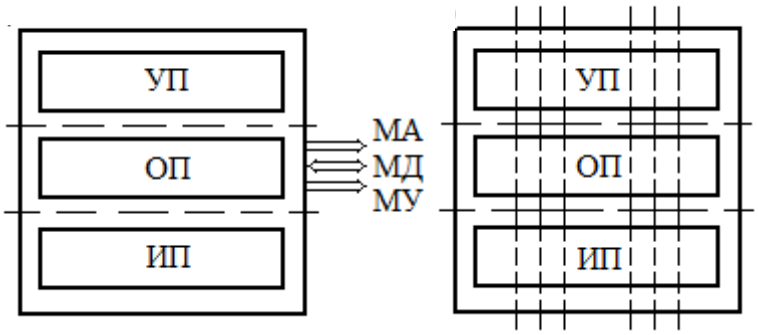
По числу больших интегральных схем (БИС) в микропроцессорном комплекте различают микропроцессоры однокристальные, многокристальные и многокристальные секционные.

Процессоры даже самых простых ЭВМ имеют сложную функциональную структуру, содержат большое количество электронных элементов и множество разветвленных связей. Изменять структуру процессора необходимо так, чтобы полная принципиальная схема или ее части имели количество элементов и связей, совместимое с возможностями БИС. При этом микропроцессоры приобретают внутреннюю магистральную архитектуру, т. е. в них к единой внутренней информационной магистрали подключаются все основные функциональные блоки (арифметико-логический, рабочих регистров, стека, прерываний, интерфейса, управления и синхронизации и др.).

Для обоснования классификации микропроцессоров по числу БИС надо распределить все аппаратные блоки процессора между основными тремя функциональными частями: операционной, управляющей и интерфейсной. Сложность операционной и управляющей частей процессора определяется их разрядностью, системой команд и требованиями к системе прерываний; сложность интерфейсной части разрядностью и возможностями подключения других устройств ЭВМ (памяти, внешних устройств, датчиков и исполнительных механизмов и др.). Интерфейс процессора содержит несколько десятков информационных шин данных (ШД), адресов (ША) и управления (ШУ).

Однокристальные микропроцессоры получаются при реализации всех аппаратных средств процессора в виде одной БИС или СБИС (сверхбольшой интегральной схемы). По мере увеличения степени интеграции элементов в кристалле и числа выводов корпуса параметры однокристальных микропроцессоров улучшаются. Однако возможности однокристальных микропроцессоров ограничены аппаратными ресурсами кристалла и корпуса. Для получения многокристального микропроцессора необходимо провести разбиение его логической структуры на функционально законченные части и реализовать их в виде БИС (СБИС). Функциональная законченность БИС многокристального микропроцессора означает, что его части выполняют заранее определенные функции и могут работать автономно.

На рис. 1.1, *а* показано функциональное разбиение структуры процессора при создании трехкристального микропроцессора (пунктирные линии), содержащего БИС операционного (ОП), управляющего (УП) и интерфейсного (ИП) процессоров.



а *б*

Рис. 1.1. Функциональная структура процессора (*а*) и ее разбиение для реализации процессора в виде комплекта секционных БИС (*б*)

Операционный процессор служит для обработки данных, управляющий процессор выполняет функции выборки, декодирования и вычисления адресов операндов и также генерирует последовательности микрокоманд. Автономность работы и большое быстродействие БИС УП позволяет выбирать команды из памяти с большей скоростью, чем скорость их исполнения БИС ОП. При этом в УП образуется очередь еще не исполненных команд, а также заранее подготавливаются те данные, которые потребуются ОП в следующих циклах работы. Такая опережающая выборка команд экономит время ОП на ожидание операндов, необходимых для выполнения команд программ. Интерфейсный процессор позволяет подключить память и периферийные средства к микропроцессору; он, по существу, является сложным контроллером для устройств ввода/вывода информации. БИС ИП выполняет также функции канала прямого доступа к памяти.

Выбираемые из памяти команды распознаются и выполняются каждой частью микропроцессора автономно и поэтому может быть обеспечен режим одновременной работы всех БИС МП, т. е. конвейерный поточный режим исполнения последовательности команд программы (выполнение последовательности с небольшим временным сдвигом). Такой режим работы значительно повышает производительность микропроцессора.

Многокристальные секционные микропроцессоры получаются в том случае, когда в виде БИС реализуются части (секции) логической

структуры процессора при функциональном разбиении ее вертикальными плоскостями (рис. 1.1, *б*). Для построения многоразрядных микропроцессоров при параллельном включении секций БИС в них добавляются средства «стыковки».

Для создания высокопроизводительных многоразрядных микропроцессоров требуется столь много аппаратных средств, не реализуемых в доступных БИС, что может возникнуть необходимость еще и в функциональном разбиении структуры микропроцессора горизонтальными плоскостями. В результате рассмотренного функционального разделения структуры микропроцессора на функционально и конструктивно законченные части создаются условия реализации каждой из них в виде БИС. Все они образуют комплект секционных БИС МП.

Таким образом, микропроцессорная секция – это БИС, предназначенная для обработки нескольких разрядов данных или выполнения определенных управляющих операций. Секционность БИС МП – определяет возможность «наращивания» разрядности обрабатываемых данных или усложнения устройств управления микропроцессора при «параллельном» включении большего числа БИС.

Однокристальные и трехкристальные БИС МП, как правило, изготовляют на основе микроэлектронных технологий униполярных полупроводниковых приборов, а многокристальные секционные БИС МП – на основе технологии биполярных полупроводниковых приборов. Использование многокристальных микропроцессорных высокоскоростных биполярных БИС, имеющих функциональную законченность при малой физической разрядности обрабатываемых данных и монтируемых в корпус с большим числом выводов, позволяет организовать разветвление связи в процессоре, а также осуществить конвейерные принципы обработки информации для повышения его производительности.

По назначению различают универсальные и специализированные микропроцессоры.

Универсальные микропроцессоры могут быть применены для решения широкого круга разнообразных задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач. Специализация МП, т. е. его проблемная ориентация на ускоренное выполнение определенных функций позволяет резко увеличить эффективную производительность при решении только определенных задач.

Среди специализированных микропроцессоров можно выделить различные микроконтроллеры, ориентированные на выполнение

сложных последовательностей логических операций, математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, например, матричных методов их выполнения, МП для обработки данных в различных областях применений и т. д. С помощью специализированных МП можно эффективно решать новые сложные задачи параллельной обработки данных. Например, конволюция позволяет осуществить более сложную математическую обработку сигналов, чем широко используемые методы корреляции. Последние в основном сводятся к сравнению и определению подобия всего двух серий данных: входных, передаваемых формой сигнала, и фиксированных опорных. Конволюция дает возможность в реальном масштабе времени находить соответствие для сигналов изменяющейся формы путем сравнения их с различными эталонными сигналами, что, например, может позволить эффективно выделить полезный сигнал на фоне шума.

Разработанные однокристальные конвольверы используются в устройствах опознавания образов в тех случаях, когда возможности сбора данных превосходят способности системы обрабатывать эти данные.

По виду обрабатываемых входных сигналов различают цифровые и аналоговые микропроцессоры. Сами микропроцессоры – цифровые устройства, однако могут иметь встроенные аналого-цифровые и цифро-аналоговые преобразователи. Поэтому входные аналоговые сигналы передаются в МП через преобразователь в цифровой форме, обрабатываются и после обратного преобразования в аналоговую форму поступают на выход. С архитектурной точки зрения такие микропроцессоры представляют собой аналоговые функциональные преобразователи сигналов и называются аналоговыми микропроцессорами. Они выполняют функции любой аналоговой схемы (например, производят генерацию колебаний, модуляцию, смещение, фильтрацию, кодирование и декодирование сигналов в реальном масштабе времени и т. д., заменяя сложные схемы, состоящие из операционных усилителей, катушек индуктивности, конденсаторов и т. д.). При этом применение аналогового микропроцессора значительно повышает точность обработки аналоговых сигналов и их воспроизводимость, а также расширяет функциональные возможности за счет программной «настройки» цифровой части микропроцессора на различные алгоритмы обработки сигналов.

Обычно в составе однокристальных аналоговых МП имеется несколько каналов аналого-цифрового и цифро-аналогового преобразо-

вания. В аналоговом микропроцессоре разрядность обрабатываемых данных достигает 24 бит и более, большое значение уделяется увеличению скорости выполнения арифметических операций.

Отличительная черта аналоговых микропроцессоров – способность к переработке большого объема числовых данных, т. е. к выполнению операций сложения и умножения с большой скоростью при необходимости даже за счет отказа от операций прерываний и переходов. Аналоговый сигнал, преобразованный в цифровую форму, обрабатывается в реальном масштабе времени и передается на выход обычно в аналоговой форме через цифро-аналоговый преобразователь. При этом согласно теореме Котельникова частота квантования аналогового сигнала должна вдвое превышать верхнюю частоту сигнала.

Сравнение цифровых микропроцессоров производится сопоставлением времени выполнения ими списков операций. Сравнение же аналоговых микропроцессоров производится по количеству эквивалентных звеньев аналого-цифровых фильтров рекурсивных фильтров второго порядка. Производительность аналогового микропроцессора определяется его способностью быстро выполнять операции умножения: чем быстрее осуществляется умножение, тем больше эквивалентное количество звеньев фильтра в аналоговом преобразователе и тем более сложный алгоритм преобразования цифровых сигналов можно задавать в микропроцессоре.

Одним из направлений дальнейшего совершенствования аналоговых микропроцессоров является повышение их универсальности и гибкости. Поэтому вместе с повышением скорости обработки большого объема цифровых данных будут развиваться средства обеспечения развитых вычислительных процессов обработки цифровой информации за счет реализации аппаратных блоков прерывания программ и программных переходов.

По характеру временной организации работы микропроцессоры делят на синхронные и асинхронные.

Синхронные микропроцессоры – микропроцессоры, в которых начало и конец выполнения операций задаются устройством управления (время выполнения операций в этом случае не зависит от вида выполняемых команд и величин операндов).

Асинхронные микропроцессоры позволяют начало выполнения каждой следующей операции определить по сигналу фактического окончания выполнения предыдущей операции. Для более эффективного использования каждого устройства микропроцессорной системы

в состав асинхронно работающих устройств вводят электронные цепи, обеспечивающие автономное функционирование устройств. Закончив работу над какой-либо операцией, устройство вырабатывает сигнал запроса, означающий его готовность к выполнению следующей операции. При этом роль естественного распределителя работ принимает на себя память, которая в соответствии с заранее установленным приоритетом выполняет запросы остальных устройств по обеспечению их командной информацией и данными.

По организации структуры микропроцессорных систем различают микроЭВМ одно- и многомагистральные.

В одномагистральных микроЭВМ все устройства имеют одинаковый интерфейс и подключены к единой информационной магистрали, по которой передаются коды данных, адресов и управляющих сигналов.

В многомагистральных микроЭВМ устройства группами подключаются к своей информационной магистрали. Это позволяет осуществить одновременную передачу информационных сигналов по нескольким (или всем) магистралям. Такая организация систем усложняет их конструкцию, однако увеличивает производительность.

По количеству выполняемых программ различают одно- и многопрограммные микропроцессоры.

В однопрограммных микропроцессорах выполняется только одна программа. Переход к выполнению другой программы происходит после завершения текущей программы.

В много- или мультипрограммных микропроцессорах одновременно выполняется несколько (обычно несколько десятков) программ. Организация мультипрограммной работы микропроцессорных управляющих систем позволяет осуществить контроль за состоянием и управлением большим числом источников или приемников информации.

1.1. Структура типового микропроцессора

Архитектура типичной небольшой вычислительной системы на основе микроЭВМ показана на рис. 1.2. Такая микроЭВМ содержит все пять основных блоков цифровой машины: устройство ввода информации, управляющее устройство (УУ), арифметико-логическое устройство (АЛУ) (входящее в состав микропроцессора), запоминающие устройства (ЗУ) и устройство вывода информации.

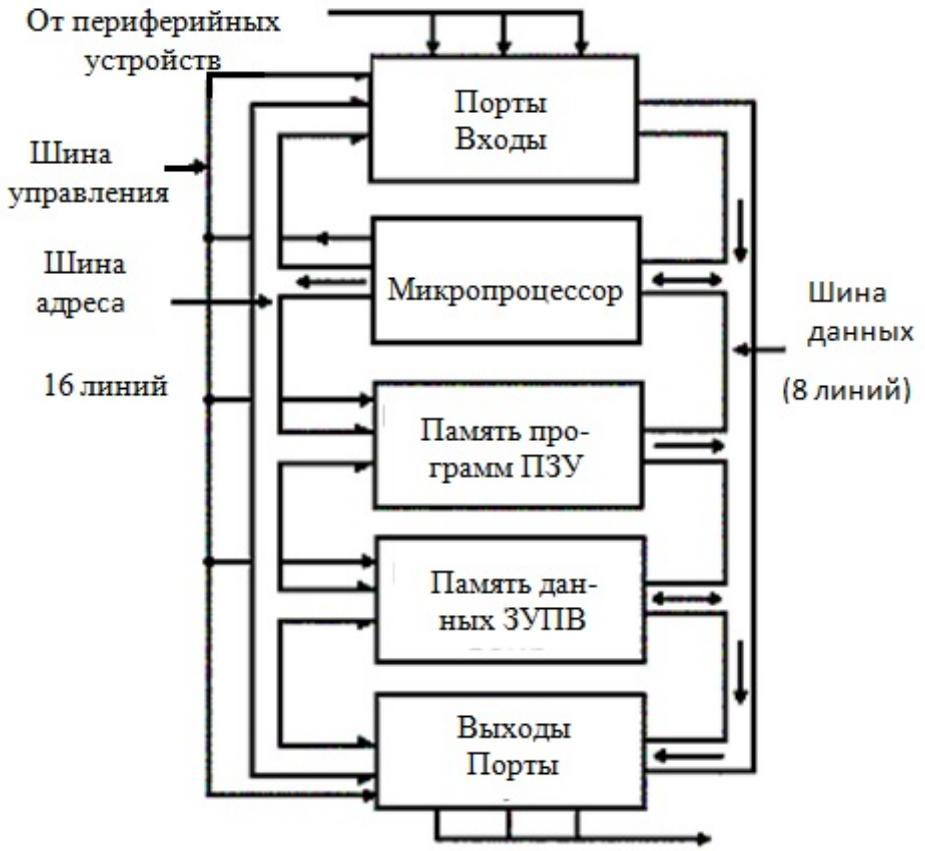


Рис. 1.2. Архитектура типового микропроцессора

Микропроцессор координирует работу всех устройств цифровой системы с помощью шины управления (ШУ). Помимо ШУ имеется 16-разрядная адресная шина (ША), которая служит для выбора определенной ячейки памяти, порта ввода или порта вывода. По 8-разрядной информационной шине или шине данных (ШД) осуществляется двунаправленная пересылка данных к микропроцессору и от микропроцессора. Важно отметить, что МП может посылать информацию в память микроЭВМ или к одному из портов вывода, а также получать информацию из памяти или от одного из портов ввода.

Постоянное запоминающее устройство (ПЗУ) в микроЭВМ содержит некоторую программу (на практике программу инициализации ЭВМ). Программы могут быть загружены в запоминающее устройство с произвольной выборкой (ЗУПВ) и из внешнего запоминающего устройства (ВЗУ). Это программы пользователя.

В качестве примера, иллюстрирующего работу микроЭВМ, рассмотрим процедуру, для реализации которой нужно выполнить следующую последовательность элементарных операций:

1. Нажать клавишу с буквой «А» на клавиатуре.
2. Поместить букву «А» в память микроЭВМ.
3. Вывести букву «А» на экран дисплея.

Это типичная процедура ввода-запоминания-вывода, рассмотрение которой дает возможность пояснить принципы использования некоторых устройств, входящих в микроЭВМ.

На рис. 1.3 приведена подробная диаграмма выполнения процедуры ввода-запоминания-вывода. Обратите внимание, что команды уже загружены в первые шесть ячеек памяти. Хранимая программа содержит следующую цепочку команд:

1. Ввести данные из порта ввода 1.
2. Запомнить данные в ячейке памяти 200.
3. Переслать данные в порт вывода 10.

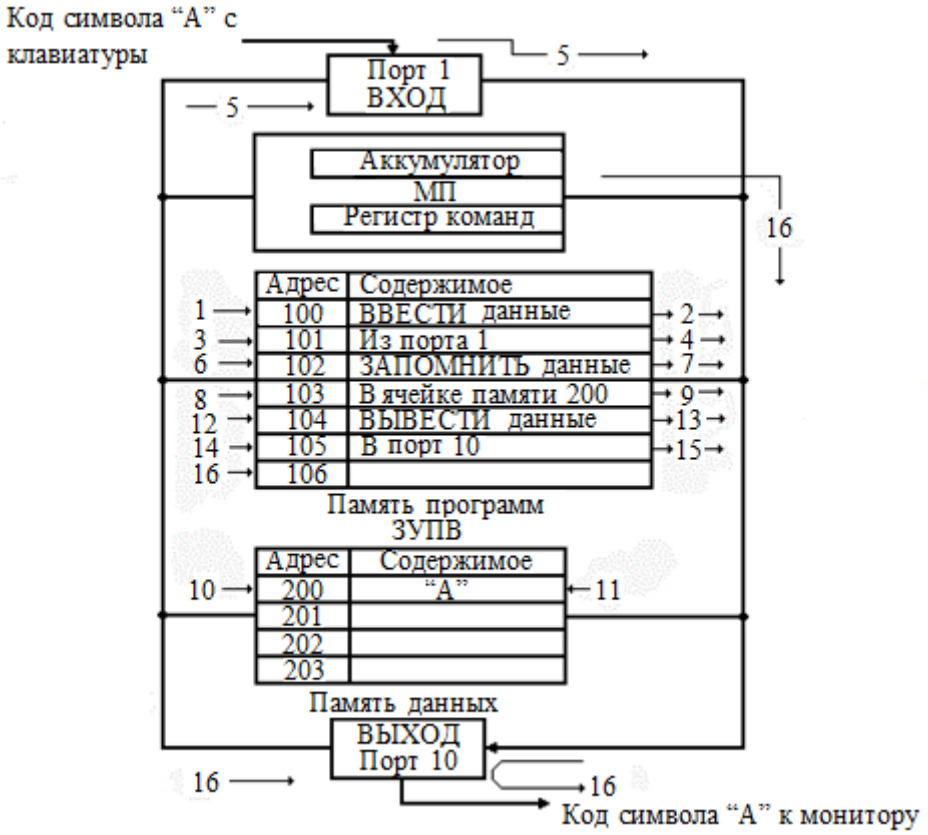


Рис. 1.3. Диаграмма выполнения процедуры ввода-запоминания-вывода

В данной программе всего три команды, хотя на рис. 1.3 может показаться, что в памяти программ записано шесть команд. Это связано с тем, что команда обычно разбивается на части. Первая часть команды 1 в приведенной выше программе – команда ввода данных.

Во второй части команды 1 указывается, откуда нужно ввести данные (из порта 1). Первая часть команды, предписывающая конкретное действие, называется кодом операции (КОП), а вторая часть – операндом. Коды операции и операнд размещаются в отдельных ячейках памяти программ. На рис. 1.3 КОП хранится в ячейке 100, а код операнда – в ячейке 101 (порт 1); последний указывает, откуда нужно взять информацию.

В МП на рис. 1.3 выделены еще два новых блока – регистры: аккумулятор и регистр команд.

Рассмотрим прохождение команд и данных внутри микроЭВМ с помощью занумерованных кружков на диаграмме. Напомним, что микропроцессор – это центральный узел, управляющий перемещением всех данных и выполнением операций.

Итак, при выполнении типичной процедуры ввода-запоминания-вывода в микроЭВМ происходит следующая последовательность действий:

1. МП выдает адрес 100 на шину адреса. По шине управления поступает сигнал, устанавливающий память программ (конкретную микросхему) в режим считывания.

2. ЗУ программ пересылает первую команду («Ввести данные») по шине данных, и МП получает это закодированное сообщение. Команда помещается в регистр команд. МП декодирует (интерпретирует) полученную команду и определяет, что для команды нужен операнд.

3. МП выдает адрес 101 на ША; ШУ используется для перевода памяти программ в режим считывания.

4. Из памяти программ на ШД пересылается операнд «Из порта 1». Этот операнд находится в программной памяти в ячейке 101. Код операнда (содержащий адрес порта 1) передается по ШД к МП и направляется в регистр команд. МП теперь декодирует полную команду («Ввести данные из порта 1»).

5. МП, используя ША и ШУ, связывающие его с устройством ввода, открывает порт 1. Цифровой код буквы «А» передается в аккумулятор внутри МП и запоминается. Важно отметить, что при обработке каждой программной команды МП действует согласно микропроцедуре выборки-декодирования-исполнения.

6. МП обращается к ячейке 102 по ША. ШУ используется для перевода памяти программ в режим считывания.

7. Код команды «Запомнить данные» подается на ШД и пересылается в МП, где помещается в регистр команд.

8. МП дешифрирует эту команду и определяет, что для нее нужен операнд. МП обращается к ячейке памяти 103 и приводит в активное состояние вход считывания микросхем памяти программ.

9. Из памяти программ на ШД пересылается код сообщения «В ячейке памяти 200». МП воспринимает этот операнд и помещает его в регистр команд. Полная команда «Запомнить данные в ячейке памяти 200» выбрана из памяти программ и декодирована.

10. Теперь начинается процесс выполнения команды. МП пересылает адрес 200 на ША и активизирует вход записи, относящийся к памяти данных.

11. МП направляет хранящуюся в аккумуляторе информацию в память данных. Код буквы «А» передается по ШД и записывается в ячейку 200 этой памяти. Выполнена вторая команда. Процесс запоминания не разрушает содержимого аккумулятора. В нем по-прежнему находится код буквы «А».

12. МП обращается к ячейке памяти 104 для выбора очередной команды и переводит память программ в режим считывания.

13. Код команды вывода данных пересылается по ШД к МП, который помещает ее в регистр команд, дешифрирует и определяет, что нужен операнд.

14. МП выдает адрес 105 на ША и устанавливает память программ в режим считывания.

15. Из памяти программ по ШД к МП поступает код операнда «В порт 10», который далее помещается в регистр команд.

16. МП дешифрирует полную команду «Вывести данные в порт 10». С помощью ША и ШУ, связывающих его с устройством вывода, МП открывает порт 10, пересылает код буквы «А» (все еще находящийся в аккумуляторе) по ШД. Буква «А» выводится через порт 10 на экран дисплея.

В большинстве микропроцессорных систем (МПС) передача информации осуществляется способом, аналогичным рассмотренному выше. Наиболее существенные различия возможны в блоках ввода и вывода информации.

Подчеркнем еще раз, что именно микропроцессор является ядром системы и осуществляет управление всеми операциями. Его работа представляет последовательную реализацию микропроцедур выборки-дешифрации-исполнения. Однако фактическая последовательность операций в МПС определяется командами, записанными в памяти программ.

Таким образом, в МПС микропроцессор выполняет следующие функции:

– выборку команд программы из основной памяти;

– дешифрацию команд;

– выполнение арифметических, логических и других операций, закодированных в командах;

– управление пересылкой информации между регистрами и основной памятью, между устройствами ввода/вывода;

– отработку сигналов от устройств ввода/вывода, в том числе реализацию прерываний с этих устройств;

– управление и координацию работы основных узлов МП.

1.2. Логическая структура микропроцессора

Логическая структура микропроцессора, т. е. конфигурация составляющих микропроцессор логических схем и связей между ними, определяется функциональным назначением. Именно структура задает состав логических блоков микропроцессора и то, как эти блоки должны быть связаны между собой, чтобы полностью отвечать архитектурным требованиям. Срабатывание электронных блоков микропроцессора в определенной последовательности приводит к выполнению заданных архитектурой микропроцессора функций, т. е. к реализации вычислительных алгоритмов. Одни и те же функции можно выполнить в микропроцессорах со структурой, отличающейся набором, количеством и порядком срабатывания логических блоков. Различные структуры микропроцессоров, как правило, обеспечивают их различные возможности, в том числе и различную скорость обработки данных. Логические блоки микропроцессора с развитой архитектурой показаны на рис. 1.4.

При проектировании логической структуры микропроцессоров необходимо рассмотреть:

1) номенклатуру электронных блоков, необходимую и достаточную для реализации архитектурных требований;

2) способы и средства реализации связей между электронными блоками;

3) методы отбора наиболее рациональных вариантов логических структур из возможного числа структур с отличающимся составом блоков и конфигурацией связей между ними.

При проектировании микропроцессора приводятся в соответствие внутренняя сложность кристалла и количество выводов корпуса.

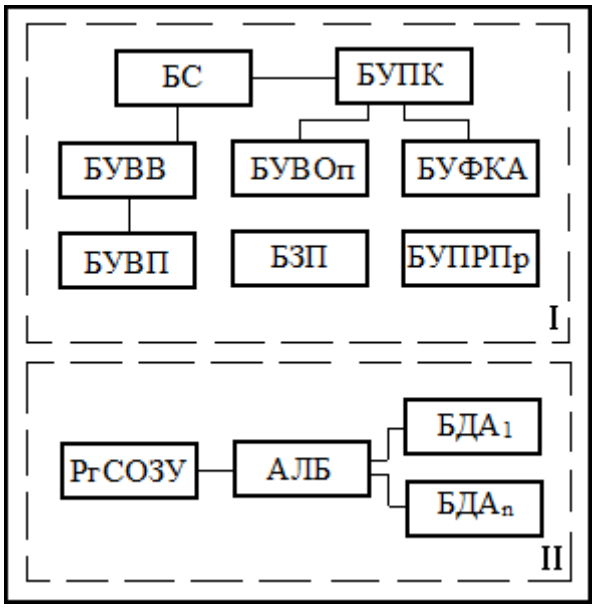


Рис. 1.4. Общая логическая структура микропроцессора:
I – управляющая часть, II – операционная часть;
БУПК – блок управления последовательно-стью команд;
БУВОп – блок управления выполнением операций;
БУФКА – блок управления формированием кодов адресов;
БУВП – блок управления виртуальной памятью; БЗП – блок защиты памяти;
БУПРПр – блок управления прерыванием работы процессора;
БУВВ – блок управления вводом/выводом;
РгСОЗУ – регистровое сверхоперативное запоминающее устройство;
АЛБ – арифметико-логический блок; БДА – блок дополнительной арифметики;
БС – блок синхронизации

Относительный рост числа элементов по мере развития микроэлектронной технологии во много раз превышает относительное увеличение числа выводов корпуса, поэтому проектирование БИС в виде конечного автомата, а не в виде набора схем, реализующих некоторый набор логических переключательных функций и схем памяти, дает возможность получить функционально законченные блоки и устройства ЭВМ.

Использование микропроцессорных комплектов БИС позволяет создать микроЭВМ для широких областей применения вследствие программной адаптации микропроцессора к конкретной области применения: изменяя программу работы микропроцессора, изменяют функции информационно-управляющей системы. Поэтому за счет составления программы работы микропроцессоров в конкретных условиях работы определенной системы можно получить оптимальные характеристики последней.

Если уровень только программной «настройки» микропроцессоров не позволит получить эффективную систему, доступен следующий уровень проектирования – микропрограммный. За счет изменения содержимого ПЗУ или программируемой логической матрицы (ПЛМ) можно «настроиться» на более специфичные черты системы обработки информации. В этом случае частично за счет изменения микропрограмм затрагивается аппаратный уровень системы. Технико-экономические последствия здесь связаны лишь с ограниченным вмешательством в технологию изготовления управляющих блоков микроЭВМ.

Изменение аппаратного уровня информационно-управляющей микропроцессорной системы, включающего в себя функциональные БИС комплекта, одновременно с конкретизацией микропрограммного и программного уровней позволяет наилучшим образом удовлетворить требованиям, предъявляемым к системе.

Решение задач управления в конкретной системе чисто аппаратными средствами (аппаратная логика) дает выигрыш в быстродействии, однако приводит к сложностям при модификации системы. Микропроцессорное решение (программная логика) является более медленным, но более гибким решением, позволяющим развивать и модифицировать систему. Изменение технических требований к информационно-управляющей микропроцессорной системе ведет лишь к необходимости перепрограммирования работы микропроцессора. Именно это качество обеспечивает высокую логическую гибкость микропроцессоров, определяет возможность их широкого использования, а значит и крупносерийного производства.

1.3. Типы архитектур

Существует несколько подходов к классификации микропроцессоров по типу архитектуры. Так, выделяют МП с CISC (CompleteInstructionSetComputer) архитектурой, характеризуемой полным набором команд, и RISC (ReduceInstructionSetComputer) архитектурой, которая определяет систему с сокращенным набором команд одинакового формата, выполняемых за один такт МП.

Определяя в качестве основной характеристики МП разрядность, выделяют следующие типы МП архитектуры:

– с фиксированной разрядностью и списком команд (однокристальные);

– с наращиваемой разрядностью (секционные) и микропрограммным управлением.

Анализируя адресные пространства программ и данных, определяют МП с архитектурой фон Неймана (память программ и память данных находятся в едином пространстве и нет никаких признаков, указывающих на тип информации в ячейке памяти) и МП с архитектурой Гарвардской лаборатории (память программ и память данных разделены, имеют свои адресные пространства и способы доступа к ним).

Мы рассмотрим более подробно основные типы архитектурных решений, выделяя связь со способами адресации памяти. Регистровая архитектура определяется наличием достаточно большого регистрового файла внутри МП. Команды получают возможность обратиться к операндам, расположенным в одной из двух запоминающих сред: оперативной памяти или регистрах.

Размер регистра обычно фиксирован и совпадает с размером слова, физически реализованного в оперативной памяти. К любому регистру можно обратиться непосредственно, поскольку регистры представлены в виде массива запоминающих элементов – регистрового файла. Типичным является выполнение арифметических операций только в регистре, при этом команда содержит два операнда (оба операнда в регистре или один операнд в регистре, а второй в оперативной памяти).

К данному типу архитектуры относится микропроцессор фирмы Zilog (рис. 1.5). Процессор Z80 – детище фирмы Zilog – помимо расширенной системы команд, одного номинала питания и способности исполнять программы, написанные для i8080, имел архитектурные «изюминки». В дополнение к основному набору РОН, в кристалле был реализован второй комплект аналогичных регистров. Это значительно упрощало работу при вызове подпрограмм или процедур обслуживания прерываний, поскольку программист мог использовать для них альтернативный набор регистров, избегая сохранения в стеке содержимого РОНов для основной программы с помощью операций PUSH. Кроме того, в систему команд был включен ряд специальных инструкций, ориентированных на обработку отдельных битов, а для поддержки регенерации динамической памяти в схему процессора введены соответствующие аппаратные средства. Z80 применялся в машинах SinclairZX, SinclairSpectrum, TandyTRS80.

Предельный вариант – архитектура с адресацией посредством аккумуляторов (меньший набор команд).

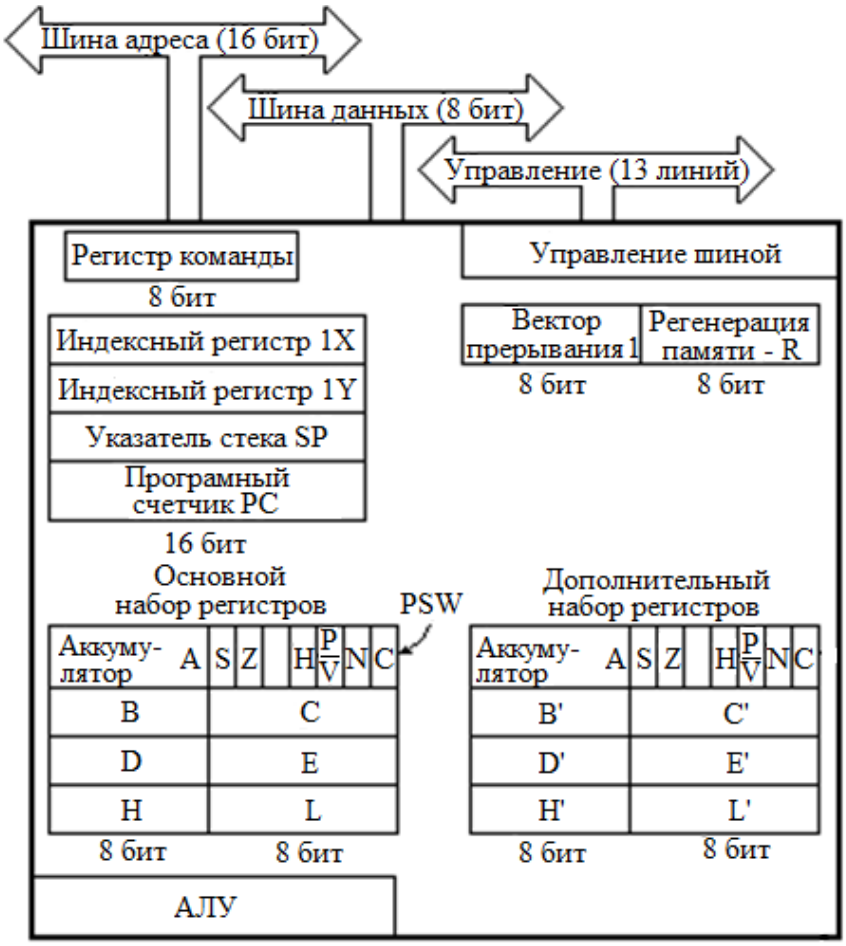


Рис. 1.5. Микропроцессор Z80 фирмы Zilog

МП фирмы Motorola имел ряд существенных преимуществ. Прежде всего, кристалл MC6800 требовал для работы одного номинала питания, а система команд оказалась весьма прозрачной для программиста. Архитектура МП также имела ряд особенностей (рис. 1.6).

Микропроцессор MC 6800 содержал два аккумулятора, и результат операции АЛУ мог быть помещен в любой из них.

Но самым ценным качеством структуры MC 6800 было автоматическое сохранение в стеке содержимого всех регистров процессора при обработке прерываний (Z80 требовалось для этого несколько команд PUSH). Процедура восстановления РОН из стека тоже выполнялась аппаратно.

1. Стековая архитектура дает возможность создать поле памяти с упорядоченной последовательностью записи и выборки информации.

2. В общем случае команды неявно адресуются к элементу стека, расположенному на его вершине, или к двум верхним элементам стека.

3. Архитектура МП, ориентированная на оперативную память (типа «память-память»), обеспечивает высокую скорость работы и большую информационную емкость рабочих регистров и стека при их организации в оперативной памяти.

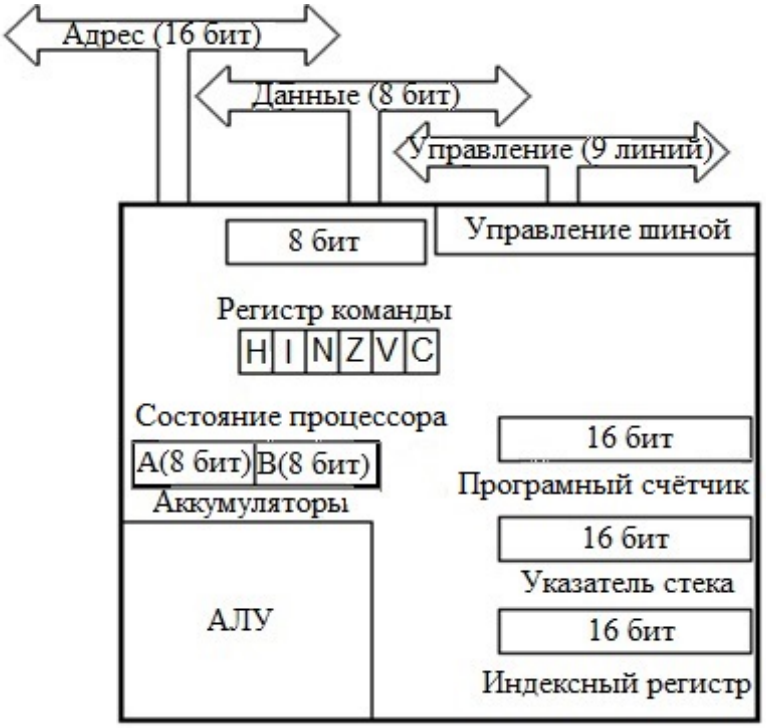


Рис. 1.6. Микропроцессор MC6800 фирмы Motorola

Архитектура этого типа не предполагает явного определения аккумулятора, регистров общего назначения или стека; все операнды команд адресуются к области основной памяти.

С точки зрения важности для пользователя-программиста под архитектурой в общем случае понимают совокупность следующих компонентов и характеристик:

– разрядности адресов и данных;
– состава, имен и назначения программно-доступных регистров;
– форматов и системы команд;
– режимов адресации памяти;
– способов машинного представления данных разного типа;
– структуры адресного пространства;
– способа адресации внешних устройств и средств выполнения операций ввода/вывода;
– классов прерываний, особенностей инициирования и обработки прерываний.

1.4. Организация ввода/вывода в микопроцессорной системе

Вводом/выводом (ВВ) называется передача данных между ядром ЭВМ, включающим в себя микропроцессор и основную память, и внешними устройствами (ВУ). Это единственное средство взаимодействия ЭВМ с «внешним миром», и архитектура ВВ (режимы работы, форматы команд, особенности прерываний, скорость обмена и др.) непосредственно влияет на эффективность всей системы. За время эволюции ЭВМ подсистема ВВ претерпела наибольшие изменения благодаря расширению сферы применения ЭВМ и появлению новых внешних устройств. Особенно важную роль средства ВВ играют в управляющих ЭВМ. Разработка аппаратных средств и программного обеспечения ВВ является наиболее сложным этапом проектирования новых систем на базе ЭВМ, а возможности ВВ серийных машин представляют собой один из важных параметров, определяющих выбор машины для конкретного применения.

1.5. Программная модель внешнего устройства

Подключение внешних устройств к системной шине осуществляется посредством электронных схем, называемых контроллерами ВВ (интерфейсами ВВ). Они согласуют уровни электрических сигналов, а также преобразуют машинные данные в формат, необходимый устройству, и наоборот. Обычно контроллеры ВВ конструктивно оформляются вместе с процессором в виде интерфейсных плат.

В процессе ввода/вывода передается информация двух видов: управляющие данные (слова) и собственно данные, или данные-сообщения. Управляющие данные от процессора, называемые также командными словами или приказами, инициируют действия, не связанные непосредственно с передачей данных, например запуск устройства, запрещение прерываний и т. п. Управляющие данные от внешних устройств называются словами состояния; они содержат информацию об определенных признаках, например о готовности устройства к передаче данных, о наличии ошибок при обмене и т. п. Состояние обычно представляется в декодированной форме – один бит для каждого признака.

Регистр, содержащий группу бит, к которой процессор обращается в операциях ВВ, образует порт ВВ. Таким образом, наиболее общая

программная модель внешнего устройства, которое может выполнять ввод и вывод, содержит четыре регистра ВВ: регистр выходных данных (выходной порт), регистр входных данных (входной порт), регистр управления и регистр состояния (рис. 1.7). Каждый из этих регистров должен иметь однозначный адрес, который идентифицируется дешифратором адреса.

В зависимости от особенностей устройства общая модель конкретизируется, например, отдельные регистры состояния и управления объединяются в один регистр, в устройстве ввода (вывода) имеется только регистр входных (выходных) данных, для ввода и вывода используется двунаправленный порт.

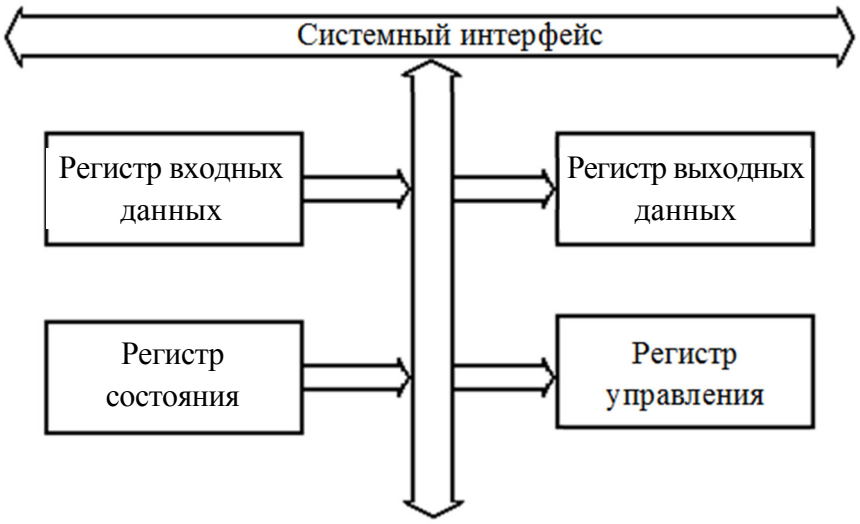


Рис. 1.7. Программная модель внешнего устройства

Непосредственные действия, связанные с вводом/выводом, реализуются одним из двух способов, различающихся адресацией регистров ВВ.

Интерфейс с изолированными шинами характеризуется раздельной адресацией памяти и внешних устройств при обмене информацией. Изолированный ВВ предполагает наличие специальных команд ввода/вывода, общий формат которых показан на рис. 1.8. При выполнении команды ввода IN содержимое адресуемого входного регистра PORT передается во внутренний регистр REG процессора, а при выполнении команды OUT содержимое регистра REG передается в выходной порт PORT. В процессоре могут быть и другие команды, относящиеся к ВВ и связанные с проверкой и модификацией содержимого регистра управления и состояния.

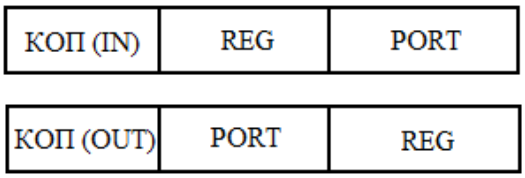


Рис. 1.8. Команды ввода/вывода (общий формат)

Нетрудно заметить, что в этом способе адресное пространство портов ввода и вывода изолировано от адресного пространства памяти, т. е. в ЭВМ один и тот же адрес могут иметь порт ВВ и ячейка памяти. Разделение адресных пространств осуществляется с помощью управляющих сигналов, относящихся к системам ВВ и памяти (MEMRD# – считывание данных из памяти, MEMWR# – запись данных в память, IORD# – чтение порта ВВ, IOWR# – запись в порт ВВ) (# – активный низкий уровень сигналов).

В ЭВМ, рассчитанной на изолированный ВВ, нетрудно перейти к ВВ, отображенному на память. Если, например, адресное пространство памяти составляет 64 Кбайт, а для программного обеспечения достаточно 32 Кбайт, то область адресов от 0 до 32 К-1 используется для памяти, от 32 К до 64 К-1 – для ввода/вывода. При этом признаком, дифференцирующим обращения к памяти и портам ВВ, может быть старший бит адреса.

Таким образом, интерфейс с общими шинами (ввод/вывод с отображением на память) имеет организацию, при которой часть общего адресного пространства отводится для внешних устройств, регистры которых адресуются так же, как и ячейки памяти. В этом случае для адресации портов ВВ используются полные адресные сигналы: READ – чтение, WRITE – запись.

В операционных системах ЭВМ имеется набор подпрограмм (драйверов ВВ), управляющих операциями ВВ стандартных внешних устройств. Благодаря им пользователь может не знать многих особенностей ВУ и интерфейсов ВВ, а применять четкие программные протоколы.

1.6. Форматы передачи данных

Рассмотрим некоторые общие вопросы, связанные с обменом данными между ВУ и микроЭВМ. Существуют два способа передачи слов информации по линиям данных: параллельный, когда одновременно пересылаются все биты слова, и последовательный, когда биты

слова пересылаются поочередно, начиная, например, с его младшего разряда.

Так как между отдельными проводниками шины для параллельной передачи данных существует электрическая емкость, то при изменении сигнала, передаваемого по одному из проводников, возникает помеха (короткий выброс напряжения) на других проводниках. С увеличением длины шины (увеличением емкости проводников) помехи возрастают и могут восприниматься приемником как сигналы. Поэтому рабочее расстояние для шины параллельной передачи данных ограничивается длиной 1–2 м, и только за счет существенного удорожания шины или снижения скорости передачи длину шины можно увеличить до 10–20 м.

Указанное обстоятельство и желание использовать для дистанционной передачи информации телеграфные и телефонные линии обусловили широкое распространение способа последовательного обмена данными между ВУ и микроЭВМ и между несколькими микроЭВМ. Возможны два режима последовательной передачи данных: синхронный и асинхронный.

При синхронной последовательной передаче каждый передаваемый бит данных сопровождается импульсом синхронизации, информирующим приемник о наличии на линии информационного бита. Следовательно, между передатчиком и приемником должны быть протянуты минимум три провода: два для передачи импульсов синхронизации и бит данных, а также общий заземленный проводник. Если же передатчик (например, микроЭВМ) и приемник (например, дисплей) разнесены на несколько метров, то каждый из сигналов (информационный и синхронизирующий) придется посылать либо по экранированному (телевизионному) кабелю, либо с помощью витой пары проводов, один из которых заземлен или передает сигнал, инверсный основному.

Синхронная последовательная передача начинается с пересылки в приемник одного или двух символов синхронизации (не путать с импульсами синхронизации). Получив такой символ (символы), приемник начинает прием данных и их преобразование в параллельный формат. Естественно, что при такой организации синхронной последовательной передачи она целесообразна лишь для пересылки массивов слов, а не отдельных символов. Это обстоятельство, а также необходимость использования для обмена сравнительно дорогих (четырехпроводных или кабельных) линий связи помешало широкому распространению синхронной последовательности передачи данных.

Асинхронная последовательная передача данных означает, что у передатчика и приемника нет общего генератора синхроимпульсов и что синхронизирующий сигнал не посылается вместе с данными. Как же в таком случае приемник будет узнавать о моментах начала и завершения передачи бит данных. Опишем простую процедуру, которую можно использовать, если передатчик и приемник асинхронной последовательной передачи данных согласованы по формату и скорости передачи.

Стандартный формат асинхронной последовательной передачи данных, используемый в ЭВМ и ВУ, содержит *n* пересылаемых бит информации (при пересылке символов *n* равно 7 или 8 битам) и 3–4 дополнительных бита: стартовый бит, бит контроля четности (или нечетности) и 1 или 2 стоповых бита (рис. 1.9, *а*). Бит четности (или нечетности) может отсутствовать. Когда передатчик бездействует (данные не посылаются на линию), на линии сохраняется уровень сигнала, соответствующий логической 1.

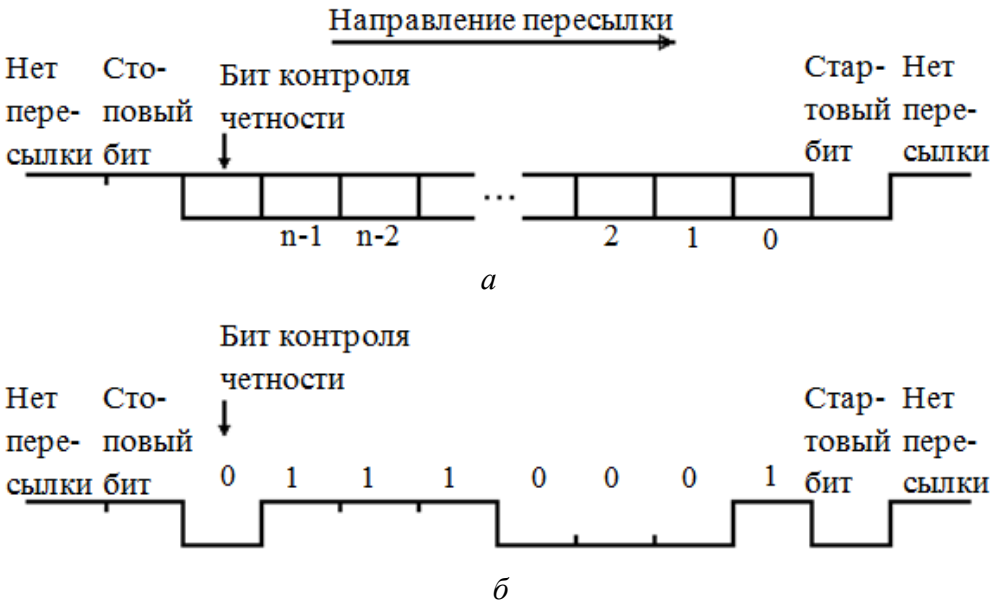


Рис. 1.9. Формат асинхронной последовательной передачи данных:
а – направление пересылки; *б* – бит контроля

Передатчик может начать пересылку символа в любой момент времени посредством генерирования стартового бита, т. е. перевода линии в состояние логического 0 на время, точно равное времени передачи бита. Затем происходит передача битов символа, начиная с младшего значащего бита, за которым следует дополнительный бит

контроля по четности или нечетности. Далее с помощью стопового бита линия переводится в состояние логической 1 (рис. 1.9, *б*). При единичном бите контроля стоповый бит не изменяет состояния сигнала на линии. Состояние логической 1 должно поддерживаться в течение промежутка времени, равного 1 или 2 временам передачи бита.

Промежуток времени от начала стартового бита до конца стопового бита (стоповых бит) называется кадром. Сразу после стоповых бит передатчик может посылать новый стартовый бит, если имеется другой символ для передачи; в противном случае уровень логической 1 может сохраняться на протяжение всего времени, пока бездействует передатчик. Новый стартовый бит может быть послан в любой момент времени после окончания стопового бита, например, через промежуток времени, равный 0,43 или 1,5 времени передачи бита.

В линиях последовательной передачи данных передатчик и приемник должны быть согласованы по всем параметрам формата, изображенного на рис. 1.9, включая номинальное время передачи бита. Для этого в приемнике устанавливается генератор синхроимпульсов, частота которого должна совпадать с частотой аналогичного генератора передатчика. Кроме того, для обеспечения оптимальной защищенности сигнала от искажения, шумов и разброса частоты синхроимпульсов приемник должен считывать принимаемый бит в середине его длительности. Рассмотрим работу приемника с того момента, когда он закончил прием символа данных и перешел в режим обнаружения стартового бита следующего слова.

Если линия перешла в состояние логического нуля и находится в этом состоянии в течение времени, не меньшего половины временного интервала передачи бита, то приемник переводится в режим считывания бит информации. В противном случае приемник остается в режиме обнаружения, так как вероятнее всего это был не стартовый бит, а шумовая помеха. В новом режиме приемник вырабатывает сигналы считывания через интервалы, равные времени передачи бита, т. е. выполняет считывание и сохранение принимаемых бит примерно на середине их передачи. Аналогичным образом будут считаны бит контроля четности и сигнал логической единицы (стоповый бит). Если оказалось, что на месте стопового бита обнаружен сигнал логического нуля, то произошла «Ошибка кадра» и символ принят неправильно. Иначе проверяется, четно ли общее число единиц в информационных битах и бите контроля, и если оно четно, производится запись принятого символа в буфер приемника.

Передний фронт стартового бита сигнализирует о начале поступления передаваемой информации, а момент его появления служит точкой отсчета времени для считывания бит данных. Стоповый бит предоставляет время для записи принятого символа в буфер приемника и обеспечивает возможность выявления ошибки кадра. Наиболее часто ошибки кадра появляются тогда, когда приемник ошибочно синхронизирован с битом 0, который в действительности не является стартовым битом. Если передатчик бездействует (посылает сигнал логической единицы) в течение одного кадра или более, то всегда можно восстановить правильную синхронизацию. Хуже обстоит дело при рассинхронизации генераторов передатчика и приемника, когда временной интервал между сигналами считывания принимаемых битов будет меньше или больше времени передачи бита.

Например, если при считывании битов посылки, показанной на рис. 1.9, *б*, временной интервал между сигналами считывания станет на 6% меньше, чем время передачи бита, то восьмой и девятый сигналы считывания будут выработаны тогда, когда на линии находится бит контроля четности (рис. 1.10). Следовательно, не будет обнаружен стоповый бит и будет зафиксирована ошибка кадра, несмотря на правильность принятой информации. Однако при 18%-й рассинхронизации генераторов, когда вместо кода (01110001) приемник зафиксирует код (11100001), никаких ошибок не будет обнаружено – четность соблюдена и стоповый (девятый по порядку) бит равен 1 (рис. 1.10).

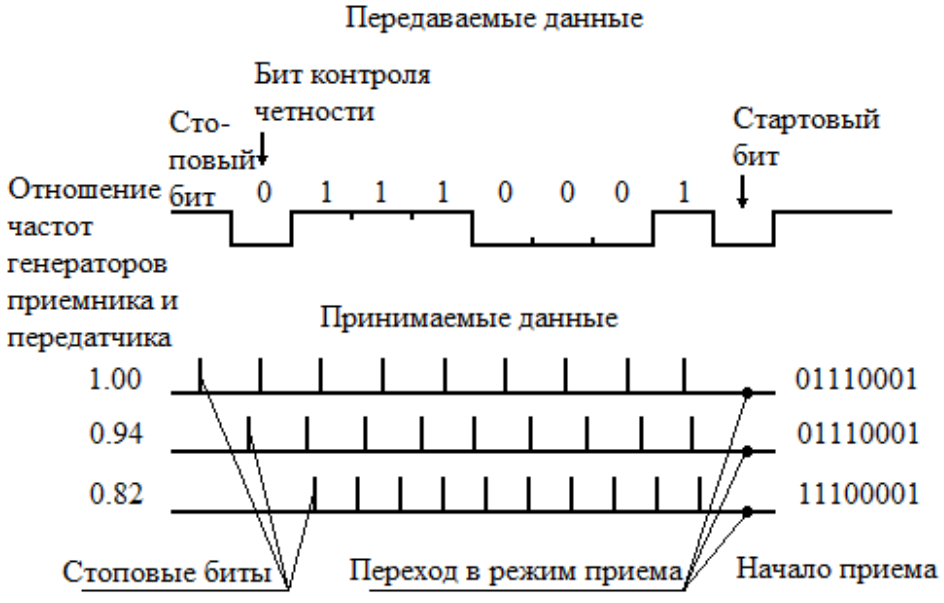


Рис. 1.10. Ошибка из-за рассинхронизации генераторов передатчика и приемника

1.7. Параллельная передача данных

Параллельная передача данных между контроллером и ВУ является по своей организации наиболее простым способом обмена. Для организации параллельной передачи данных помимо шины данных, количество линий в которой равно числу одновременно передаваемых битов данных, используется минимальное количество управляющих сигналов.

В простом контроллере ВУ, обеспечивающем побайтную передачу данных на внешнее устройство (рис. 1.11), в шине связи с ВУ используются всего два управляющих сигнала: «Выходные данные готовы» и «Данные приняты».

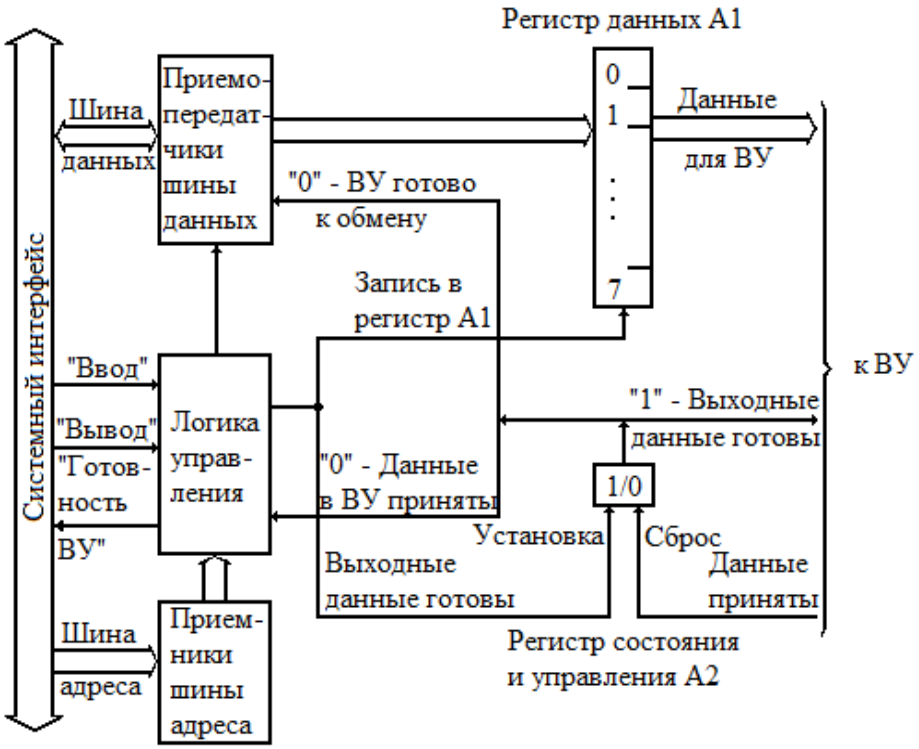


Рис. 1.11. Простой параллельный контроллер вывода

Для формирования управляющего сигнала «Выходные данные готовы» и приема из ВУ управляющего сигнала «Данные приняты» в контроллере используется одноразрядный адресуемый регистр состояния и управления А2 (обычно используются раздельные регистр состояния и регистр управления). Одновременно с записью очередного байта данных с шины данных системного интерфейса в адресуемый регистр данных контроллера (порт вывода А1) в регистр состояния и управле-

ния записывается логическая единица. Тем самым формируется управляющий сигнал «Выходные данные готовы» в шине связи с ВУ.

ВУ, приняв байт данных, управляющим сигналом «Данные приняты» обнуляет регистр состояния контроллера. При этом формируются управляющий сигнал системного интерфейса «Готовность ВУ» и признак готовности ВУ к обмену, передаваемый в процессор по одной из линий шины данных системного интерфейса посредством стандартной операции ввода при реализации программы асинхронного обмена.

Логика управления контроллера обеспечивает селекцию адресов регистров контроллера, прием управляющих сигналов системного интерфейса и формирование на их основе внутренних управляющих сигналов контроллера, формирование управляющего сигнала системного интерфейса «Готовность ВУ». Для сопряжения регистров контроллера с шинами адреса и данных системного интерфейса в контроллере используются соответственно приемники шины адреса и приемопередатчики шины данных.

Рассмотрим на примере, каким образом контроллер ВУ обеспечивает параллельную передачу данных в ВУ под управлением программы асинхронного обмена. Алгоритм асинхронного обмена в данном случае передачи прост.

1. Процессор микроЭВМ проверяет готовность ВУ к приему данных.

2. Если ВУ готово к приему данных (в данном случае это логический 0 в нулевом разряде регистра А2), то данные передаются с шины данных системного интерфейса в регистр данных А1 контроллера и далее в ВУ. Иначе повторяется п. 1.

Пример 1.1. Фрагмент программы передачи байта данных в асинхронном режиме с использованием параллельного контроллера ВУ (рис. 1.11). Для написания программы асинхронной передачи воспользуемся командами процессора 8086.

MOV	DX, A2	Номер порта А2 помещаем в DX
m1:IN	AL, DX	Чтение байта из порта А2
TEST	AL, 1	Проверка нулевого состояния регистра А2
JNS	ml	Переход на метку ml если разряд не нулевой
MOV	AL, 64	Выводимый байт данных помещается в AL
MOV	DX, A1	Номер порта А1 записываем в DX
OUT	DX, AL	Содержимое регистра АХ передаем в порт А1

Команда во второй строке приводит к следующим действиям. При ее выполнении процессор по шине адреса передает в контроллер адрес A2, сопровождая его сигналом «Ввод» (IORD#; здесь и далее в скобках указаны сигналы на шине ISA). Логика управления контроллера, реагируя на эти сигналы, обеспечивает передачу в процессор содержимого регистра состояния A2 по шине данных системного интерфейса.

Команда в третьей строке приводит к следующим действиям. Процессор проверяет значение соответствующего разряда принятых данных. Нуль в этом разряде указывает на неготовность ВУ к приему данных и, следовательно, на необходимость возврата к проверке содержимого A2, т. е. процессор, выполняя три первые команды, ожидает готовности ВУ к приему данных. Единица в этом разряде подтверждает готовность ВУ и, следовательно, возможность передачи байта данных.

В седьмой строке осуществляется пересылка данных из регистра AX процессора в регистр данных контроллера A1. Процессор по шине адреса передает в контроллер адрес A1, а по шине данных – байт данных, сопровождая их сигналом «Вывод» (IOWR#). Логика управления контроллера обеспечивает запись данных с шины данных в регистр данных A1 и устанавливает в ноль бит готовности регистра состояния A2, формируя тем самым управляющий сигнал для ВУ «Выходные данные готовы».

ВУ принимает байт данных и управляющим сигналом «Данные приняты» устанавливает в единицу регистр состояния A2. (Далее контроллер ВУ по этому сигналу может сформировать и передать в процессор сигнал «Готовность ВУ», который в данном случае извещает процессор о приеме данных внешним устройством и разрешает процессору снять сигнал «Вывод» и тем самым завершить цикл вывода данных в команде пересылки, однако в IBM-совместимых персональных компьютерах с шиной ISA сигнал «Готовность ВУ» не формируется, а имеется сигнал IO CH RDY#, позволяющий продлить цикл обмена, если устройство недостаточно быстрое. В данном случае нет необходимости в сигнале «Готовность ВУ», т. к. шина ISA является синхронной и, следовательно, все операции выполняются по тактовым импульсам.)

Блок-схема простого контроллера ВУ, обеспечивающего побайтный прием данных из ВУ, приведена на рис. 1.12. В этом контроллере при взаимодействии с внешним устройством также используются два управляющих сигнала: «Данные от ВУ готовы» и «Данные приняты».

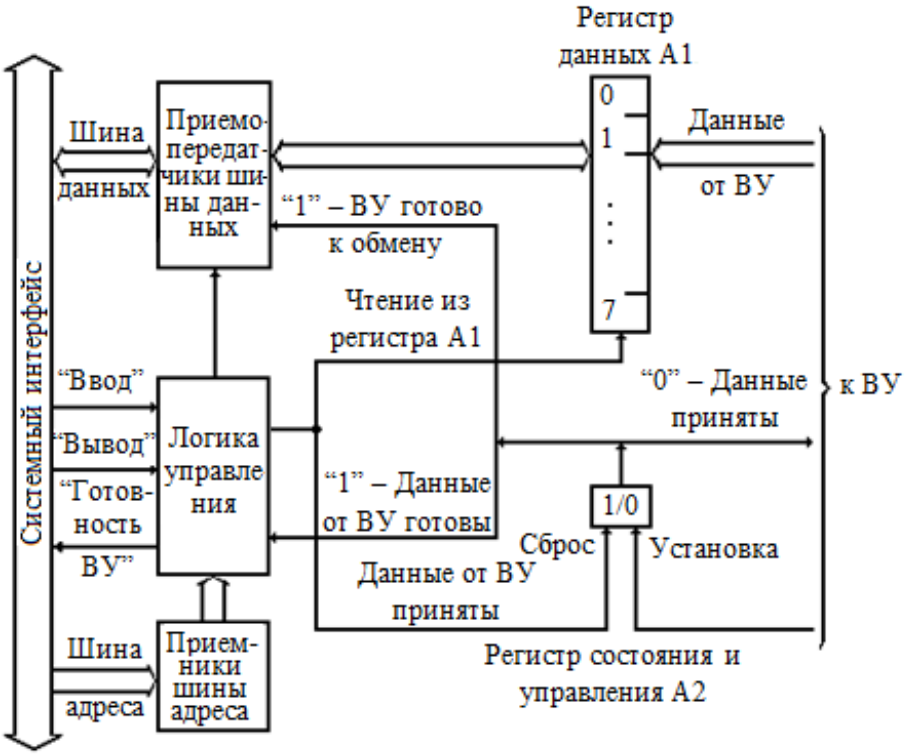


Рис. 1.12. Простой параллельный контроллер ввода

Для формирования управляющего сигнала «Данные приняты» и приема из ВУ управляющего сигнала «Данные от ВУ готовы» используется одноразрядный адресуемый регистр состояния и управления А2.

Внешнее устройство записывает в регистр данных контроллера А1 очередной байт данных и управляющим сигналом «Данные от ВУ готовы» устанавливает в единицу регистр состояния и управления А2.

При этом формируются: управляющий сигнал системного интерфейса «Готовность ВУ»; признак готовности ВУ к обмену, передаваемый в процессор по одной из линий шины данных системного интерфейса посредством операции ввода при реализации программы асинхронного обмена.

Тем самым контроллер извещает процессор о готовности данных в регистре А1. Процессор, выполняя программу асинхронного обмена, читает байт данных из регистра данных контроллера и обнуляет регистр состояния и управления А2. При этом формируется управляющий сигнал «Данные приняты» в шине связи с внешним устройством.

Логика управления контроллера и приемопередатчики шин системного интерфейса выполняют те же функции, что и в контроллере вывода (рис. 1.12).

Рассмотрим работу параллельного интерфейса ввода при реализации программы асинхронного обмена. Алгоритм асинхронного ввода так же прост, как и асинхронного вывода.

1. Процессор проверяет наличие данных в регистре данных контроллера A1.

2. Если данные готовы (логическая 1 в регистре A2), то они передаются из регистра данных A1 на шину данных системного интерфейса и далее в регистр процессора или ячейку памяти микрокомпьютера. Иначе повторяется п. 1.

Пример 1.2. Фрагмент программы приема байта данных в асинхронном режиме с использованием параллельного интерфейса (контроллер ВУ).

MOV	DX, A2	Номер порта A2 помещаем в DX
m1:IN	AL, DX	Чтение байта из порта A2
TEST	AL, 1	Проверка нулевого разряда состояния регистра A2
JZ	ml	Переход на метку ml если разряд не нулевой
MOV	DX, A1	Номер порта A1 записываем в DX
IN	AL, DX	Содержимое регистра A1 передаем в регистр AL

В третьей строке выполняется проверка содержимого регистра A2, т. е. признака наличия данных в регистре данных A1. Команда выполняется точно так же, как и в примере 1.1. Единица в нулевом разряде (содержимое регистра A2) подтверждает, что данные от ВУ записаны в регистр данных контроллера и необходимо переслать их на шину данных. Нуль в знаковом разряде указывает на неготовность данных от ВУ и, следовательно, на необходимость вернуться к проверке готовности.

IN AL, DX – пересылка данных из регистра данных контроллера A1 в регистр процессора AL. Процессор передает в контроллер по шине адреса системного интерфейса адрес A1, сопровождая его сигналом «Ввод». Логика управления контроллера в ответ на сигнал «Ввод» (IORD#) обеспечивает передачу байта данных из регистра данных A1 на шину данных и, в общем случае, но не в IBM-совместимом персональном компьютере с шиной ISA, сопровождает его сигналом «Готовность ВУ», который подтверждает наличие данных от ВУ на шине данных и по которому процессор считывает байт с шины данных и помещает его в указанный регистр. (В IBM-совместимом персональном компьютере с шиной ISA процессор счи-

тывает байт с шины данных по истечении определенного времени после установки сигнала IORD#). Затем логика управления обнуляет регистр состояния и управления A2, формируя тем самым управляющий сигнал для внешнего устройства «Данные приняты». Таким образом завершается цикл ввода данных.

Как видно из рассмотренных примеров, для приема или передачи одного байта данных процессору необходимо выполнить всего несколько команд, время выполнения которых и определяет максимально достижимую скорость обмена данными при параллельной передаче. Таким образом, при параллельной передаче обеспечивается довольно высокая скорость обмена данными, которая ограничивается только быстродействием ВУ.

1.8. Последовательная передача данных

Использование последовательных линий связи для обмена данными с внешними устройствами возлагает на контроллеры ВУ дополнительные по сравнению с контроллерами для параллельного обмена функции. Во-первых, возникает необходимость преобразования формата данных: из параллельного формата, в котором они поступают в контроллер ВУ из системного интерфейса микроЭВМ, в последовательный при передаче в ВУ и из последовательного в параллельный при приеме данных из ВУ. Во-вторых, требуется реализовать соответствующий режиму работы внешнего устройства способ обмена данными: синхронный или асинхронный.

1.9. Синхронный последовательный интерфейс

Простой контроллер для синхронной передачи данных в ВУ по последовательной линии связи (последовательный интерфейс) представлен на рис. 1.13.

Восьмиразрядный адресуемый буферный регистр контроллера A1 служит для временного хранения байта данных до его загрузки в сдвиговый регистр. Запись байта данных в буферный регистр с шины данных системного интерфейса производится так же, как и в параллельном интерфейсе (см. Параллельная передача данных), только при наличии единицы в одноразрядном адресуемом регистре состояния контроллера A2. Единица в регистре состояния указывает на готовность контроллера принять очередной байт в буферный регистр.

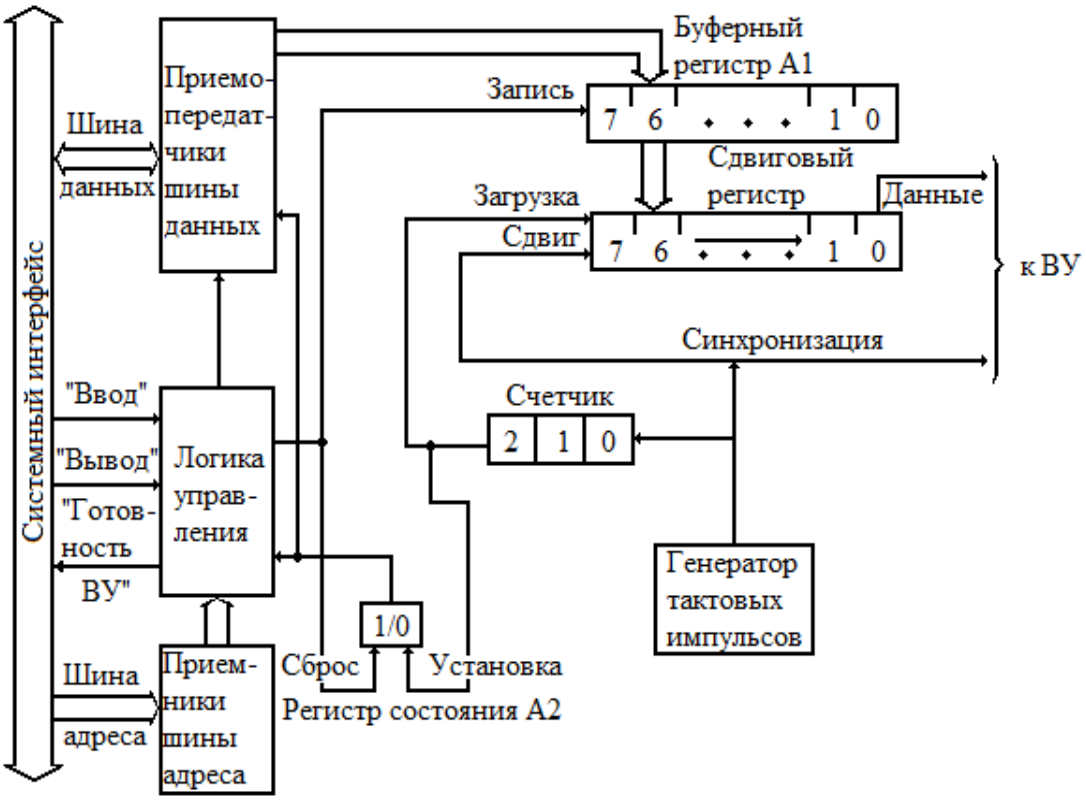


Рис. 1.13. Контроллер последовательной синхронной передачи

Содержимое регистра А2 передается в процессор по одной из линий шины данных системного интерфейса и используется для формирования управляющего сигнала системного интерфейса «Готовность ВУ». При записи очередного байта в буферный регистр А1 обнуляется регистр состояния А2.

Программа записи байта данных в буферный регистр аналогична программе из примера 1.1 за исключением команды перехода: вместо команды JNZ m1 (переход, если не ноль) необходимо использовать команду JZ m1 (переход, если ноль).

Преобразование данных из параллельного формата, в котором они поступили в буферный регистр контроллера из системного интерфейса, в последовательный и передача их на линию связи производятся в сдвиговом регистре с помощью генератора тактовых импульсов и двоичного трехразрядного счетчика импульсов следующим образом.

Последовательная линия связи контроллера с ВУ подключается к выходу младшего разряда сдвигового регистра. По очередному тактовому импульсу содержимое сдвигового регистра сдвигается на один разряд вправо и в линию связи «Данные» выдается значение очередного разряда. Одновременно со сдвигом в ВУ передается по отдель-

ной линии «Синхронизация» тактовый импульс. Таким образом, каждый передаваемый по линии «Данные» бит информации сопровождается синхронизирующим сигналом по линии «Синхронизация», что обеспечивает его однозначное восприятие на приемном конце последовательной линии связи.

Количество переданных в линию тактовых сигналов, а следовательно, и переданных бит информации подсчитывается счетчиком тактовых импульсов. Как только содержимое счетчика становится равным 7, т. е. в линию переданы 8 бит (1 байт) информации, формируется управляющий сигнал «Загрузка», обеспечивающий запись в сдвиговый регистр очередного байта из буферного регистра. Этим же управляющим сигналом устанавливается в «1» регистр состояния. Очередным тактовым импульсом счетчик будет сброшен в «0», и начнется очередной цикл выдачи восьми битов информации из сдвигового регистра в линию связи.

Синхронная последовательная передача отдельных битов данных на линию связи должна производиться без какого-либо перерыва, и следующий байт данных должен быть загружен в буферный регистр из системного интерфейса за время, не превышающее времени передачи восьми битов в последовательную линию связи.

При записи байта данных в буферный регистр обнуляется регистр состояния контроллера. Нуль в этом регистре указывает, что в линию связи передается байт данных из сдвигового регистра, а следующий передаваемый байт данных загружен в сдвиговый регистр.

Контроллер для последовательного синхронного приема данных из ВУ состоит из тех же компонентов, что и контроллер для синхронной последовательной передачи, за исключением генератора тактовых импульсов.

1.10. Асинхронный последовательный интерфейс

Организация асинхронного последовательного обмена данными с внешним устройством осложняется тем, что на передающей и приемной стороне последовательной линии связи используются настроенные на одну частоту, но физически разные генераторы тактовых импульсов и, следовательно, общая синхронизация отсутствует. Рассмотрим на примерах организацию контроллеров последовательных интерфейсов для последовательных асинхронных передачи и приема данных.

Простейший контроллер для асинхронной передачи данных в ВУ по последовательной линии связи представлен на рис. 1.14. Он предназначен для передачи данных в формате с двумя стоповыми битами.

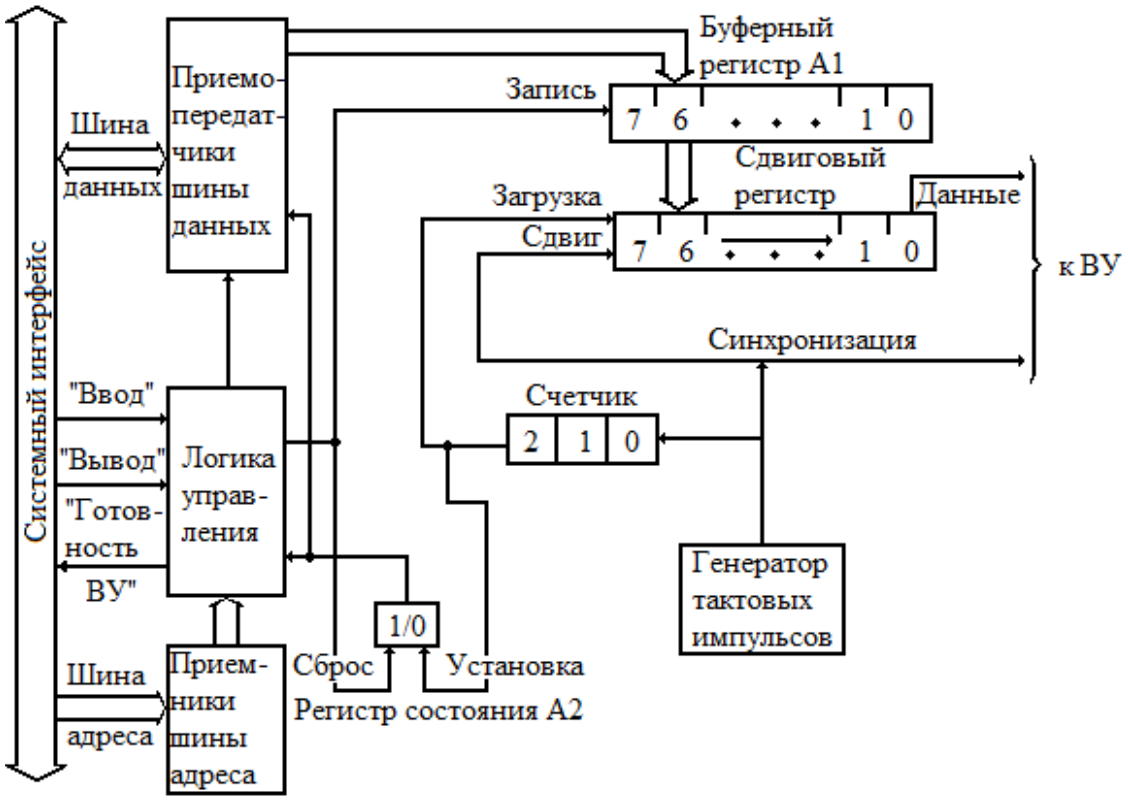


Рис. 1.14. Контроллер последовательной асинхронной передачи

После передачи очередного байта данных в регистр состояния А2 записывается 1. Единичный выходной сигнал регистра А2 информирует процессор о готовности контроллера к приему следующего байта данных и передаче его по линии связи в ВУ. Этот же сигнал запрещает формирование импульсов со схемы выработки импульсов сдвига – делителя частоты сигналов тактового генератора на 16. Счетчик импульсов сдвига (счетчик по mod 10) находится в нулевом состоянии и его единичный выходной сигнал поступает на вентиль И, подготавливая цепь выработки сигнала загрузки сдвигового регистра.

Процесс передачи байта данных начинается с того, что процессор, выполняя команду «Вывод», выставляет этот байт на шине данных. Одновременно процессор формирует управляющий сигнал системного интерфейса «Вывод», по которому производятся запись передаваемого байта в буферный регистр А1, сброс регистра состояния А2 и

формирование на вентиле И сигнала «Загрузка». Передаваемый байт переписывается в разряды 1, ... , 8 сдвигового регистра, в нулевой разряд сдвигового регистра записывается 0 (стартовый бит), а в разряды 9 и 10 – 1 (стоповые биты). Кроме того, снимается сигнал «Сброс» с делителя частоты, он начинает накапливать импульсы генератора тактовой частоты и в момент приема шестнадцатого тактового импульса вырабатывает импульс сдвига.

На выходной линии контроллера «Данные» поддерживается состояние 0 (значение стартового бита) до тех пор, пока не будет выработан первый импульс сдвига. Импульс сдвига изменит состояние счетчика импульсов сдвига и перепишет в нулевой разряд сдвигового регистра первый информационный бит передаваемого байта данных. Состояние, соответствующее значению этого бита, будет поддерживаться на линии «Данные» до следующего импульса сдвига.

Аналогично будут переданы остальные информационные биты, первый стоповый бит и, наконец, второй стоповый бит, при передаче которого счетчик импульсов сдвига снова установится в нулевое состояние. Это приведет к записи 1 в регистр состояния А2. Единичный сигнал с выхода регистра А2 запретит формирование импульсов сдвига, а также информирует процессор о готовности к приему нового байта данных. После завершения передачи очередного кадра (стартового бита, информационного байта и двух стоповых бит) контроллер поддерживает в линии связи уровень логической единицы (значение второго стопового бита).

Уровень логической единицы поступает по линии «Данные» в контроллер для асинхронного приема данных (рис. 1.15). Этот уровень создает условия для выработки сигнала, запрещающего работу делителя частоты генератора тактовых импульсов. Действительно, после приема предыдущего байта данных счетчик импульсов сдвига (счетчик по mod 9) находится в нулевом состоянии и на вентиль И поступают два единичных сигнала: со счетчика сдвигов и из линии «Данные». На выходе вентиля И вырабатывается сигнал сброса делителя частоты сигналов тактового генератора, запрещающий формирование импульсов сдвига.

В момент смены стопового бита на стартовый бит (момент начала передачи нового кадра) на линии «Данные» появится уровень логического нуля и тем самым будет снят сигнал сброса с делителя частоты. Состояние 4-разрядного двоичного счетчика (делителя частоты) начнет изменяться. Когда на счетчике накопится значение 8, он вы-

даст сигнал, поступающий на входы сдвигового регистра и счетчика импульсов сдвига.

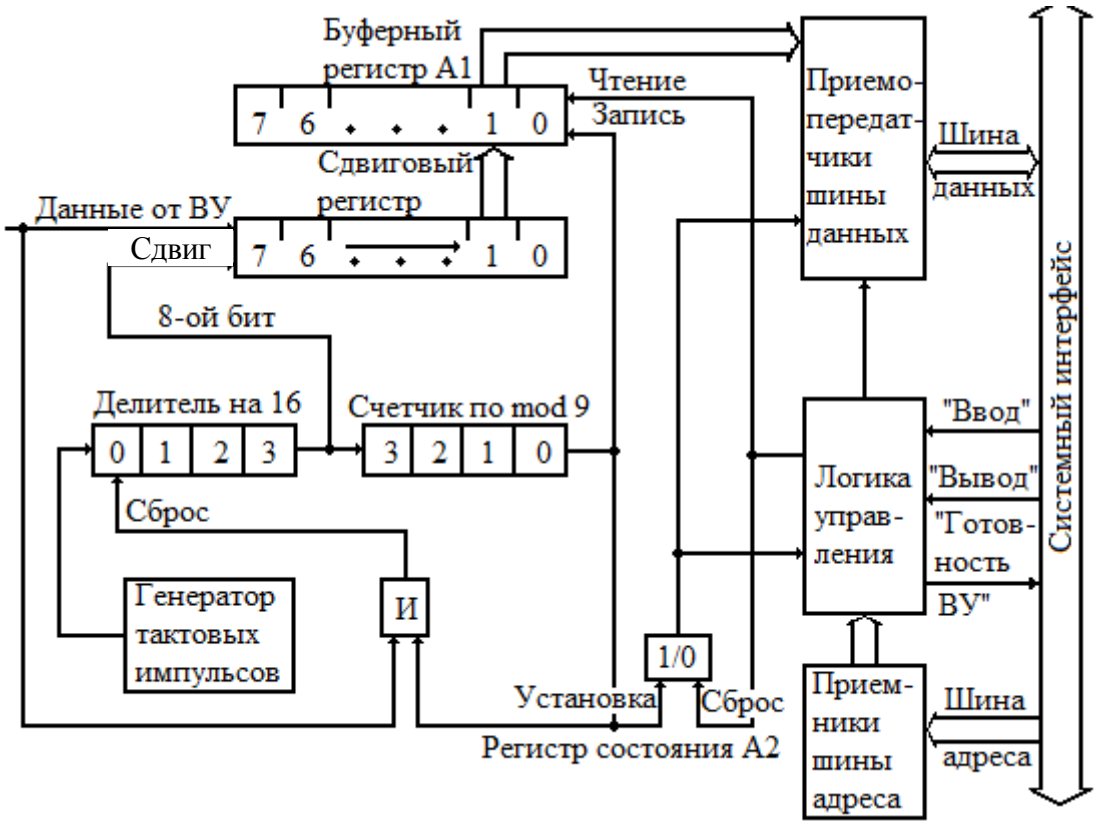


Рис. 1.15. Контроллер последовательного асинхронного приема

Так как частота сигналов генератора тактовых импульсов приемника должна совпадать с частотой генератора тактовых импульсов передатчика, то сдвиг (считывание) бита произойдет примерно на середине временного интервала, отведенного на передачу бита данных, т. е. времени, необходимого для выработки шестнадцати тактовых импульсов. Это делается для уменьшения вероятности ошибки из-за возможного различия частот генераторов передатчика и приемника, искажения формы передаваемых сигналов (переходные процессы) и т. п. Следующий сдвиг произойдет после прохождения шестнадцати тактовых импульсов, т. е. на середине временного интервала передачи первого информационного бита.

При приеме в сдвиговый регистр девятого бита кадра (восьмого информационного бита) из него «выдвинется» стартовый бит и, следовательно, в сдвиговом регистре будет размещен весь принятый байт информации. В этот момент счетчик импульсов сдвига придет в нулевое состояние и на его выходе будет выработан единичный сигнал, по

которому содержимое сдвигового регистра перепишется в буферный регистр, в регистр состояния А2 запишется 1 и он будет информировать процессор об окончании приема очередного байта, вентиль И подготовится к выработке сигнала «Сброс» (этот сигнал сформируется после прихода первого стопового бита).

Получив сигнал готовности (1 в регистре А2), процессор выполнит команду «Ввод» (см. пример 1.2 Параллельной передачи данных). При этом вырабатывается управляющий сигнал системного интерфейса «Ввод», по которому производятся пересылка принятого байта данных из буферного регистра в процессор (сигнал «Чтение») и сброс регистра состояния А2.

Отметим, что для простоты изложения в контроллере на рис. 1.15 не показаны схемы контроля стоповых бит принимаемого кадра. Не показаны также схемы контроля четности или нечетности (паритета) передаваемой информации (обычно в передаваемом байте восьмому биту придается значение 0 или 1, так чтобы в этом байте было четное количество единиц). В реальных контроллерах имеются такие схемы, и если контроллер не принимает из линии связи нужного количества стоповых бит или вырабатывается сигнал ошибки паритета в схеме контроля четности, то принятые в текущем кадре биты данных игнорируются и контроллер ожидает поступления нового стартового бита.

Обмен данными с ВУ по последовательным линиям связи широко используется в микроЭВМ, особенно в тех случаях, когда не требуется высокой скорости обмена. Вместе с тем применение в них последовательных линий связи с ВУ обусловлено двумя важными причинами. Во-первых, последовательные линии связи просты по своей организации: два провода при симплексной и полудуплексной передаче и максимум четыре – при дуплексной. Во-вторых, в микроЭВМ используются внешние устройства, обмен с которыми необходимо вести в последовательном коде.

В современных микроЭВМ применяют, как правило, универсальные контроллеры для последовательного ВВ, обеспечивающие как синхронный, так и асинхронный режим обмена данными с ВУ.

1.11. Способы обмена информацией в микропроцессорной системе

В ЭВМ применяются три режима ввода/вывода: программно-управляемый ВВ (называемый также программным или неформированным ВВ), ВВ по прерываниям (форсированный ВВ) и прямой доступ к

памяти. Первый из них характеризуется тем, что инициирование и управление ВВ осуществляется программой, выполняемой процессором, а внешние устройства играют сравнительно пассивную роль и сигнализируют только о своем состоянии, в частности, о готовности к операциям ввода/вывода.

Во втором режиме ВВ инициируется не процессором, а внешним устройством, генерирующим специальный сигнал прерывания. Реагируя на этот сигнал готовности устройства к передаче данных, процессор передает управление подпрограмме обслуживания устройства, вызвавшего прерывание. Действия, выполняемые этой подпрограммой, определяются пользователем, а непосредственными операциями ВВ управляет процессор. Наконец, в режиме прямого доступа к памяти, который используется, когда пропускной способности процессора недостаточно, действия процессора приостанавливаются, он отключается от системной шины и не участвует в передачах данных между основной памятью и быстродействующим ВУ. Заметим, что во всех вышеуказанных случаях основные действия, выполняемые на системной магистрали ЭВМ, подчиняются двум основным принципам.

В процессе взаимодействия любых двух устройств ЭВМ одно из них обязательно выполняет активную, управляющую роль и является задатчиком, второе оказывается управляемым, исполнителем. Чаще всего задатчиком является процессор.

Другим важным принципом, заложенным в структуру интерфейса, является принцип квитирования (запроса – ответа): каждый управляющий сигнал, посланный задатчиком, подтверждается сигналом исполнителя. При отсутствии ответного сигнала исполнителя в течение заданного интервала времени формируется так называемый тайм-аут, задатчик фиксирует ошибку обмена и прекращает данную операцию.

1.12. Программно-управляемый ввод/вывод

Данный режим характеризуется тем, что все действия по вводу/выводу реализуются командами прикладной программы. Наиболее простыми эти действия оказываются для «всегда готовых» внешних устройств, например индикатора на светодиодах. При необходимости ВВ в соответствующем месте программы используются команды IN или OUT. Такая передача данных называется синхронным или безусловным ВВ.

Однако для большинства ВУ до выполнения операций ВВ надо убедиться в их готовности к обмену, т. е. ВВ является асинхронным. Общее состояние устройства характеризуется флагом готовности READY, называемым также флагом готовности/занятости (READY/BUSY). Иногда состояния готовности и занятости идентифицируются отдельными флагами READY и BUSY, входящими в слово состояния устройства.

Процессор проверяет флаг готовности с помощью одной или нескольких команд. Если флаг установлен, то инициируются собственно ввод или вывод одного или нескольких слов данных. Когда же флаг сброшен, процессор выполняет цикл из 2-3 команд с повторной проверкой флага READY до тех пор, пока устройство не будет готово к операциям ВВ (рис. 1.16). Данный цикл называется циклом ожидания готовности ВУ и реализуется в различных процессорах по-разному.

Основной недостаток программного ВВ связан с непроизводительными потерями времени процессора в циклах ожидания. К достоинствам следует отнести простоту его реализации, не требующей дополнительных аппаратных средств.

Рис. 1.16. Цикл программного ожидания готовности внешнего устройства

1.13. Способы обмена информацией в микропроцессорной системе

1.13.1. Организация прерываний в микроЭВМ

Одной из разновидностей программно-управляемого обмена данными с ВУ в микроЭВМ является обмен с прерыванием программы, отличающийся от асинхронного программно-управляемого обмена

тем, что переход к выполнению команд, физически реализующих обмен данными, осуществляется с помощью специальных аппаратных средств. Команды обмена данными в этом случае выделяют в отдельный программный модуль – подпрограмму обработки прерывания. Задачей аппаратных средств обработки прерывания в процессоре микроЭВМ как раз и является приостановка выполнения одной программы (ее еще называют основной программой) и передача управления подпрограмме обработки прерывания. Действия, выполняемые при этом процессором, как правило, те же, что и при обращении к подпрограмме. Только при обращении к подпрограмме они инициируются командой, а при обработке прерывания – управляющим сигналом от ВУ, который называют «Запрос на прерывание» или «Требование прерывания».

Эта важная особенность обмена с прерыванием программы позволяет организовать обмен данными с ВУ в произвольные моменты времени, не зависящие от программы, выполняемой в микроЭВМ. Таким образом, появляется возможность обмена данными с ВУ в реальном масштабе времени, определяемом внешней по отношению к микроЭВМ средой. Обмен с прерыванием программы существенным образом экономит время процессора, затрачиваемое на обмен. Это происходит за счет того, что исчезает необходимость в организации программных циклов ожидания готовности ВУ (см. примеры 1.1 и 1.2, Параллельная передача данных), на выполнение которых тратится значительное время, особенно при обмене с медленными ВУ.

Прерывание программы по требованию ВУ не должно оказывать на прерванную программу никакого влияния кроме увеличения времени ее выполнения за счет приостановки на время выполнения подпрограммы обработки прерывания. Поскольку для выполнения подпрограммы обработки прерывания используются различные регистры процессора (счетчик команд, регистр состояния и т. д.), то информацию, содержащуюся в них в момент прерывания, необходимо сохранить для последующего возврата в прерванную программу.

Обычно задача сохранения содержимого счетчика команд и регистра состояния процессора возлагается на аппаратные средства обработки прерывания. Сохранение содержимого других регистров процессора, используемых в подпрограмме обработки прерывания, производится непосредственно в подпрограмме. Отсюда следует достаточно очевидный факт: чем больший объем информации о прерванной программе сохраняется программным путем, тем больше время реакции микроЭВМ на сигнал прерывания, и наоборот. Предпочтительными с

точки зрения повышения производительности микроЭВМ (сокращения времени выполнения подпрограмм обработки, а, следовательно, и основной программы) являются уменьшение числа команд, обеспечивающих сохранение информации о прерванной программе, и реализация этих функций аппаратными средствами.

Формирование сигналов прерываний – запросов ВУ на обслуживание происходит в контроллерах соответствующих ВУ. В простейших случаях в качестве сигнала прерывания может использоваться сигнал «Готовность ВУ», поступающий из контроллера ВУ в системный интерфейс микроЭВМ. Однако такое простое решение обладает существенным недостатком – процессор не имеет возможности управлять прерываниями, т. е. разрешать или запрещать их для отдельных ВУ. В результате организация обмена данными в режиме прерывания с несколькими ВУ существенно усложняется.

Для решения этой проблемы регистр состояния и управления контроллера ВУ (рис. 1.17) дополняют еще одним разрядом – «Разрешение прерывания». Запись 1 или 0 в разряд «Разрешение прерывания» производится программным путем по одной из линий шины данных системного интерфейса. Управляющий сигнал системного интерфейса «Запрос на прерывание» формируется с помощью схемы совпадения только при наличии единиц в разрядах «Готовность ВУ» и «Разрешение прерывания» регистра состояния и управления контроллера.

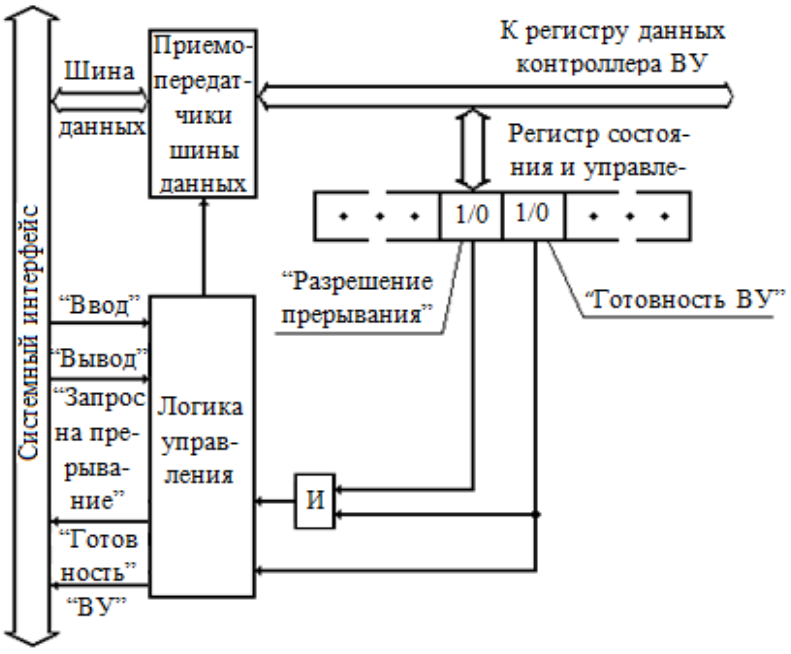


Рис. 1.17. Фрагмент блок-схемы контроллера ВУ с разрядом «Разрешение прерывания» в регистре состояния и управления

Аналогичным путем решается проблема управления прерываниями в микроЭВМ, в целом. Для этого в регистре состояния процессора выделяется разряд, содержимое которого определяет, разрешены или запрещены прерывания от внешних устройств. Значение этого разряда может устанавливаться программным путем.

В микроЭВМ обычно используется одноуровневая система прерываний, т. е. сигналы «Запрос на прерывание» от всех ВУ поступают на один вход процессора. Поэтому возникает проблема идентификации ВУ, запросившего обслуживание, и реализации заданной очередности (приоритета) обслуживания ВУ при одновременном поступлении нескольких сигналов прерывания. Существуют два основных способа идентификации ВУ, запросивших обслуживания:

– программный опрос регистров состояния (разряд «Готовность ВУ») контроллеров всех ВУ;

– использование векторов прерывания.

Организация прерываний с программным опросом готовности предполагает наличие в памяти микроЭВМ единой подпрограммы обслуживания прерываний от всех внешних устройств. Структура такой подпрограммы приведена на рис. 1.18.

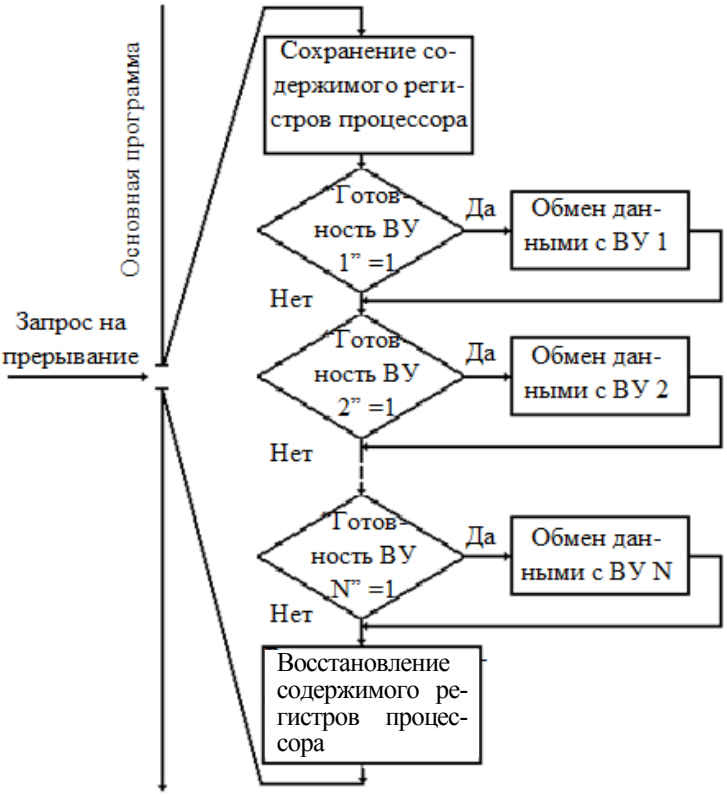


Рис. 1.18. Структура единой программы обработки прерываний и ее связь с основной программой

Обслуживание ВУ с помощью единой подпрограммы обработки прерываний производится следующим образом. В конце последнего машинного цикла выполнения очередной команды основной программы процессор проверяет наличие требования прерывания от ВУ. Если сигнал прерывания есть и в процессоре прерывание разрешено, то процессор переключается на выполнение подпрограммы обработки прерываний.

После сохранения содержимого регистров процессора, используемых в подпрограмме, начинается последовательный опрос регистров состояния контроллеров всех ВУ, работающих в режиме прерывания. Как только подпрограмма обнаружит готовое к обмену ВУ, сразу выполняются действия по его обслуживанию. Завершается подпрограмма обработки прерывания после опроса готовности всех ВУ и восстановления содержимого регистров процессора.

Приоритет ВУ в микроЭВМ с программным опросом готовности внешнего устройства однозначно определяется порядком их опроса в подпрограмме обработки прерываний. Чем раньше в подпрограмме опрашивается готовность ВУ, тем меньше время реакции на его запрос и выше приоритет. Необходимость проверки готовности всех внешних устройств существенно увеличивает время обслуживания тех ВУ, которые опрашиваются последними. Это является основным недостатком рассматриваемого способа организации прерываний. Поэтому обслуживание прерываний с опросом готовности ВУ используется только в тех случаях, когда отсутствуют жесткие требования на время обработки сигналов прерывания внешних устройств.

Организация системы прерываний в микроЭВМ с использованием векторов прерываний позволяет устранить указанный недостаток. При такой организации системы прерываний ВУ, запросившее обслуживания, само идентифицирует себя с помощью вектора прерывания – адреса ячейки основной памяти микроЭВМ, в которой хранится либо первая команда подпрограммы обслуживания прерывания данного ВУ, либо адрес начала такой подпрограммы. Таким образом, процессор, получив вектор прерывания, сразу переключается на выполнение требуемой подпрограммы обработки прерывания. В микроЭВМ с векторной системой прерывания каждое ВУ должно иметь собственную подпрограмму обработки прерывания.

Различают векторные системы с интерфейсным и внеинтерфейсным вектором. В первом случае вектор прерывания (или его адрес) формирует контроллер ВУ, запросившего обслуживания, во

втором – контроллер прерываний, общий для всех устройств, работающих в режиме прерываний (IBM-совместимые персональные компьютеры).

Рассмотрим организацию векторной системы с интерфейсным вектором. Вектор прерывания (или его адрес) выдается контроллером не одновременно с запросом на прерывание, а только по разрешению процессора, как это реализовано в схеме на рис. 1.19. Это делается для того, чтобы исключить одновременную выдачу векторов прерывания от нескольких ВУ. В ответ на сигнал контроллера ВУ «Запрос на прерывание» процессор формирует управляющий сигнал «Предоставление прерывания (вх.)», который разрешает контроллеру ВУ, запросившему обслуживание, выдачу вектора прерывания в шину адреса системного интерфейса. Для этого в контроллере используются регистр вектора прерывания и схема совпадения И3. Регистр вектора прерывания обычно реализуется с помощью перемычек или переключателей, что позволяет пользователю устанавливать для конкретных ВУ требуемые значения векторов прерывания.

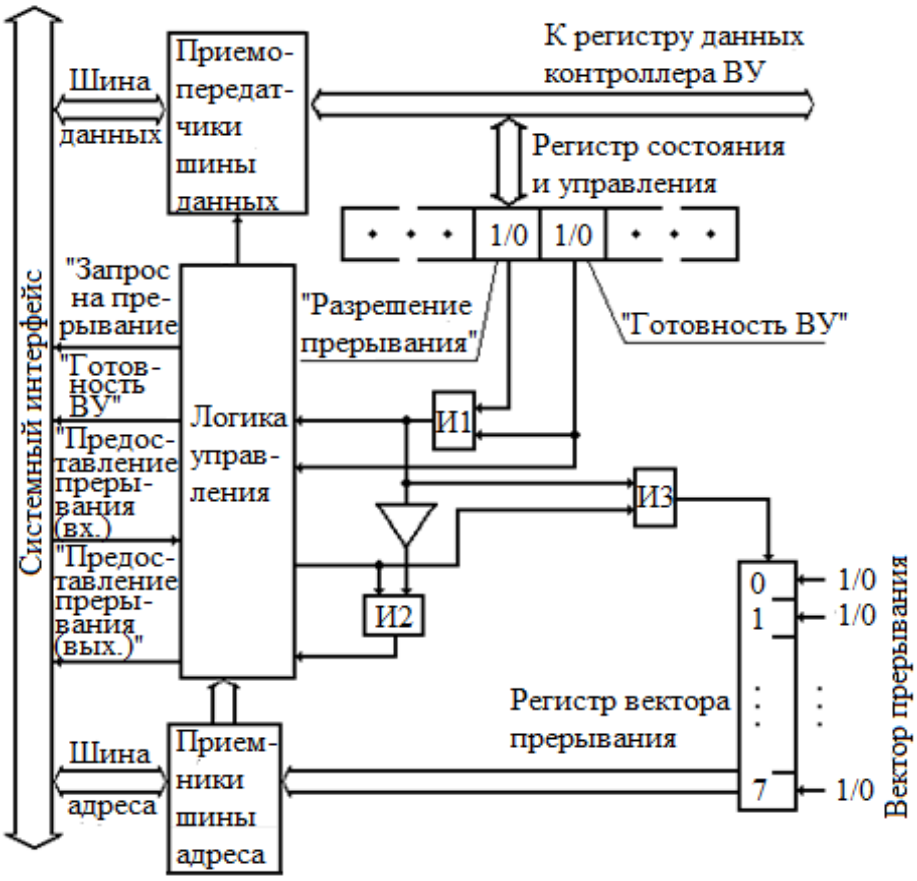


Рис. 1.19. Формирование векторов прерывания в контроллере ВУ

Управляющий сигнал «Предоставление прерывания (вых.)» формируется в контроллере ВУ с помощью схемы совпадения И2. Этот сигнал используется для организации последовательного аппаратного опроса готовности ВУ и реализации тем самым требуемых приоритетов ВУ. Процессор при поступлении в него по общей линии системного интерфейса «Запрос на прерывание» сигнала прерывания формирует управляющий сигнал «Предоставление прерывания (вх.)», который поступает сначала в контроллер ВУ с наивысшим приоритетом (рис. 1.20). Если это устройство не требовало обслуживания, то его контроллер пропускает сигнал «Предоставление прерывания» на следующий контроллер, иначе дальнейшее распространение сигнала прекращается и контроллер выдает вектор прерывания на адресно-информационную шину.

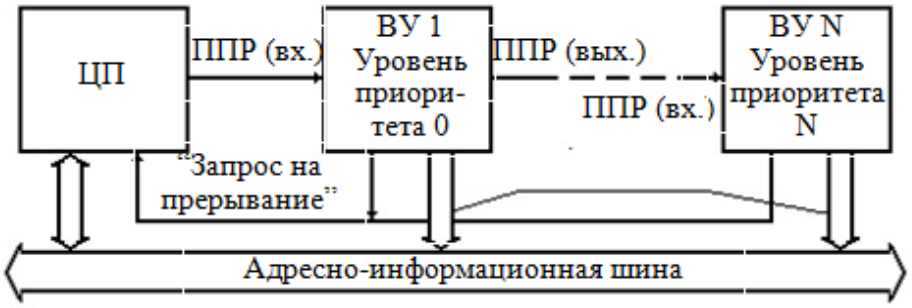


Рис. 1.20. Реализация приоритетов ВУ в микроЭВМ с векторной системой прерываний с интерфейсным вектором:
ППР (вх.) – «Предоставление прерывания (входной)»;
ППР (вых.) – «Предоставление прерывания (выходной)»

Аппаратный опрос готовности ВУ производится гораздо быстрее, нежели программный. Но если обслуживания запросили одновременно два или более ВУ, обслуживание менее приоритетных ВУ будет отложено на время обслуживания более приоритетных, как и в системе прерывания с программным опросом.

Рассмотренная векторная система прерываний практически полностью соответствует системе прерываний, реализованной в микроЭВМ «Электроника-60». Восьмиразрядный вектор прерывания в «Электронике-60» указывает одну из ячеек памяти с адресами от 0 до (376)8, в которой размещается адрес начала подпрограммы обработки прерывания. В следующей за указанной вектором прерывания ячейке памяти хранится новое содержимое регистра состояния процессора, загружаемое в него при переключении на подпрограмму обработки прерывания. Один из бит нового содержимого регистра состояния

процессора запрещает или разрешает прерывания от других ВУ, что позволяет ВУ с более высоким приоритетом прерывать подпрограммы обслуживания ВУ с меньшим приоритетом и наоборот.

Векторная система с внеинтерфейсным вектором прерывания используется в IBM-совместимых персональных компьютерах. В этих компьютерах контроллеры внешних устройств не имеют регистров для хранения векторов прерывания, а для идентификации устройств, запросивших обслуживания, используется общий для всех ВУ контроллер прерываний. Ниже приведен пример контроллера прерываний INTEL 8259A.

БИС программируемого контроллера прерываний (ПКП) представляет собой устройство, реализующее до восьми уровней запросов на прерывания с возможностью программного маскирования и изменения порядка обслуживания прерываний. За счет каскадного включения БИС ПКП число уровней прерывания может быть расширено до 64 (в архитектуре персонального компьютера IBM PC AT – 16).

Структурная схема ПКП приведена на рис. 1.21.

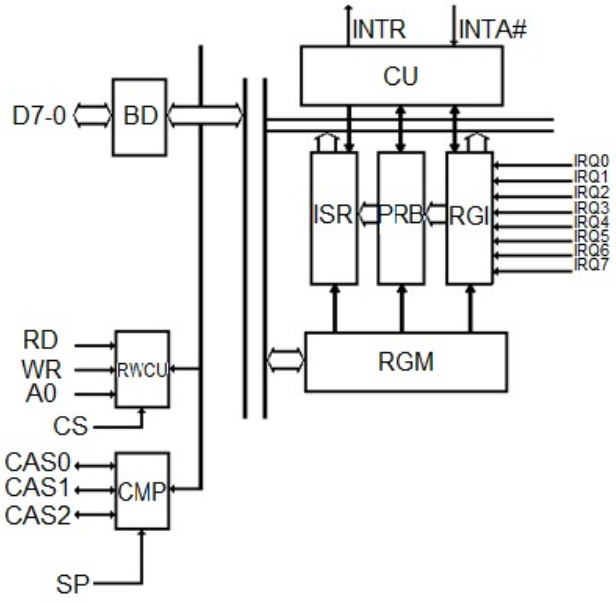


Рис. 1.21. Контроллер прерываний Intel 8259A

В состав БИС входят:

– RGI – регистр запретов прерываний; хранит все уровни, на которые поступают запросы IRQx;

– PRB – схема принятия решений по приоритетам; схема идентифицирует приоритет запросов и выбирает запрос с наивысшим приоритетом;

– ISR – регистр обслуживаемых прерываний; сохраняет уровни запросов прерываний, находящиеся на обслуживании ПКП;

– RGM – регистр маскирования прерываний; обеспечивает запрещение одной или нескольких линий запросов прерывания;

– BD – буфер данных; предназначен для сопряжения ПКП с системной шиной данных;

– RWCU – блок управления записью/чтением; принимает управляющие сигналы от микропроцессора и задает режим функционирования ПКП;

– CMP – схема каскадного буфера-компаратора; используется для включения в систему нескольких ПКП;

– CU – схема управления; вырабатывает сигналы прерывания и формирует трехбайтовую команду CALL для выдачи на шину данных.

Установка ПКП в исходное состояние и «настройка» его на определенный режим обслуживания прерываний происходит с помощью двух типов команд: команд инициализации (ICW) и команд управления операциями (OCW).

Программируемый контроллер прерываний (ПКП) имеет 16 входов запросов прерываний (IRQ 0 – IRQ 15). Контроллер состоит из двух каскадно включенных контроллеров – выход INTR (запрос на прерывание) второго контроллера подключен ко входу IRQ 2 первого контроллера. В качестве примера отметим, что к линии IRQ 0 подключен системный таймер, к линии IRQ 1 – клавиатура, к линии IRQ 8 – часы реального времени и т. д.

Упрощенная схема взаимодействия контроллера прерываний с процессором и контроллером шины имеет следующий вид (рис. 1.22).

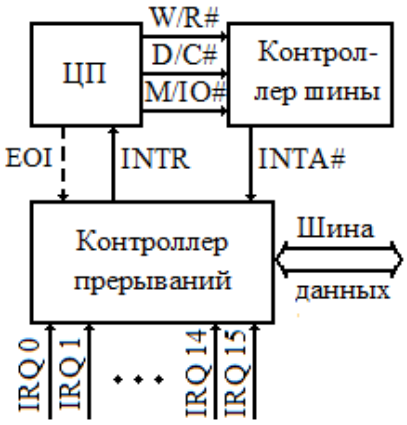


Рис. 1.22. Упрощенная схема взаимодействия контроллера прерываний с процессором и контроллером шины в IBM-совместимых персональных компьютерах класса AT

Эта схема функционирует следующим образом. Пусть в некоторый момент времени контроллер клавиатуры с помощью единичного сигнала по линии IRQ 1 известил контроллер прерываний о своей готовности к обмену. В ответ на запрос контроллер прерываний генерирует сигнал INTR (запрос на прерывание) и посылает его на соответствующий вход процессора. Процессор, если маскируемые прерывания разрешены (т. е. установлен флаг прерываний IF в регистре флагов процессора), посылает на контроллер шины сигналы R# – чтение, C# – управление и IO# – ввод/вывод, определяющие тип цикла шины. Контроллер шины, в свою очередь, генерирует два сигнала подтверждения прерывания INTA# и направляет их на контроллер прерываний. По второму импульсу контроллер прерываний выставляет на шину данных восьмибитный номер вектора прерывания, соответствующий данной линии IRQ.

В режиме реального адреса («реальном» режиме) векторы прерываний хранятся в таблице векторов прерываний, которая находится в первом килобайте оперативной памяти. Под каждый вектор отведено 4 байта (2 байта под адрес сегмента и 2 байта под смещение), т. е. в таблице может содержаться 256 векторов. Адрес вектора в таблице – номер вектора * 4.

Далее процессор считывает номер вектора прерывания. Сохраняет в стеке содержимое регистра флагов, сбрасывает флаг прерываний IF и помещает в стек адрес возврата в прерванную программу (регистры CS и IP). После этого процессор извлекает из таблицы векторов прерываний адрес подпрограммы обработки прерываний для данного устройства и приступает к ее выполнению.

Процедура обработки аппаратного прерывания должна завершаться командой конца прерывания EOI (EndofInterruption), посылаемой контроллеру прерываний. Для этого необходимо записать байт 20h в порт 20h (для первого контроллера) и в порт A0h (для второго).

В IBM PC/XT/AT используется режим прерываний с фиксированными приоритетами. Высшим приоритетом обладает запрос по линии IRQ 0, низшим – IRQ 7. Так как второй контроллер подключен к линии IRQ 2 первого контроллера, то приоритеты линий IRQ в порядке убывания приоритета располагаются следующим образом: IRQ 0, IRQ 1, IRQ 8 – IRQ 15, IRQ 3 – IRQ 7. Если запрос на обслуживание посылают одновременно два устройства с разными приоритетами, то контроллер обслуживает запрос с большим приоритетом, а запрос с меньшим приоритетом блокирует. Блокировка сохраняется до получения команды EOI.

1.13.2. Организация прямого доступа к памяти

Одним из способов обмена данными с ВУ является обмен в режиме прямого доступа к памяти (ПДП). В этом режиме обмен данными между ВУ и основной памятью микроЭВМ происходит без участия процессора. Обменом в режиме ПДП управляет не программа, выполняемая процессором, а электронные схемы, внешние по отношению к процессору. Обычно схемы, управляющие обменом в режиме ПДП, размещаются или в специальном контроллере, который называется контроллером прямого доступа к памяти, или в контроллере самого ВУ.

Обмен данными в режиме ПДП позволяет использовать в микроЭВМ быстродействующие внешние запоминающие устройства, такие, например, как накопители на жестких магнитных дисках, поскольку ПДП может обеспечить время обмена одним байтом данных между памятью и ВЗУ, равное циклу обращения к памяти.

Для реализации режима прямого доступа к памяти необходимо обеспечить непосредственную связь контроллера ПДП и памяти микроЭВМ. Для этой цели можно было бы использовать специально выделенные шины адреса и данных, связывающие контроллер ПДП с основной памятью. Но такое решение нельзя признать оптимальным, так как это приведет к значительному усложнению микроЭВМ в целом, особенно при подключении нескольких ВЗУ. В целях сокращения количества линий в шинах микроЭВМ контроллер ПДП подключается к памяти посредством шин адреса и данных системного интерфейса. При этом возникает проблема совместного использования шин системного интерфейса процессором и контроллером ПДП. Можно выделить два основных способа ее решения: реализация обмена в режиме ПДП с «захватом цикла» и в режиме ПДП с блокировкой процессора.

Существуют две разновидности прямого доступа к памяти с «захватом цикла». Наиболее простой способ организации ПДП состоит в том, что для обмена используются те машинные циклы процессора, в которых он не обменивается данными с памятью. В такие циклы контроллер ПДП может обмениваться данными с памятью, не мешая работе процессора. Однако возникает необходимость выделения таких циклов, чтобы не произошло временного перекрытия обмена ПДП с операциями обмена, инициируемыми процессором. В некоторых процессорах формируется специальный управляющий сигнал, указывающий циклы, в которых процессор не обращается к систем-

ному интерфейсу. При использовании других процессоров для выделения таких циклов необходимо применение в контроллерах ПДП специальных селектирующих схем, что усложняет их конструкцию. Применение рассмотренного способа организации ПДП не снижает производительности микроЭВМ, но при этом обмен в режиме ПДП возможен только в случайные моменты времени одиночными байтами или словами.

Более распространенным является ПДП с «захватом цикла» и принудительным отключением процессора от шин системного интерфейса. Для реализации такого режима ПДП системный интерфейс микроЭВМ дополняется двумя линиями для передачи управляющих сигналов «Требование прямого доступа к памяти» (ТПДП) и «Предоставление прямого доступа к памяти» (ППДП).

Управляющий сигнал ТПДП формируется контроллером прямого доступа к памяти. Процессор, получив этот сигнал, приостанавливает выполнение очередной команды, не дожидаясь ее завершения, выдает на системный интерфейс управляющий сигнал ППДП и отключается от шин системного интерфейса. С этого момента все шины системного интерфейса управляются контроллером ПДП. Контроллер ПДП, используя шины системного интерфейса, осуществляет обмен одним байтом или словом данных с памятью микроЭВМ и затем, сняв сигнал ТПДП, возвращает управление системным интерфейсом процессору. Как только контроллер ПДП будет готов к обмену следующим байтом, он вновь «захватывает» цикл процессора и т. д. В промежутках между сигналами ТПДП процессор продолжает выполнять команды программы. Тем самым выполнение программы замедляется, но в меньшей степени, чем при обмене в режиме прерываний.

Применение в микроЭВМ обмена данными с ВУ в режиме ПДП всегда требует предварительной подготовки, а именно: для каждого ВУ необходимо выделить область памяти, используемую при обмене, и указать ее размер, т. е. количество записываемых в память или читаемых из памяти байт (слов) информации. Следовательно, контроллер ПДП должен обязательно иметь в своем составе регистр адреса и счетчик байт (слов). Перед началом обмена с ВУ в режиме ПДП процессор должен выполнить программу загрузки. Эта программа обеспечивает запись в указанные регистры контроллера ПДП начального адреса выделенной ВУ памяти и ее размера в байтах или словах в зависимости от того, какими порциями информации ведется обмен. Сказанное не относится к начальной загрузке программ в память в режиме ПДП. В этом случае содержимое регистра адреса и счетчика

байт слов устанавливается переключателями или перемычками непосредственно на плате контроллера.

Блок-схема простого контроллера ПДП, обеспечивающего ввод данных в память микроЭВМ по инициативе ВУ в режиме ПДП «Захват цикла», приведена на рис. 1.23.

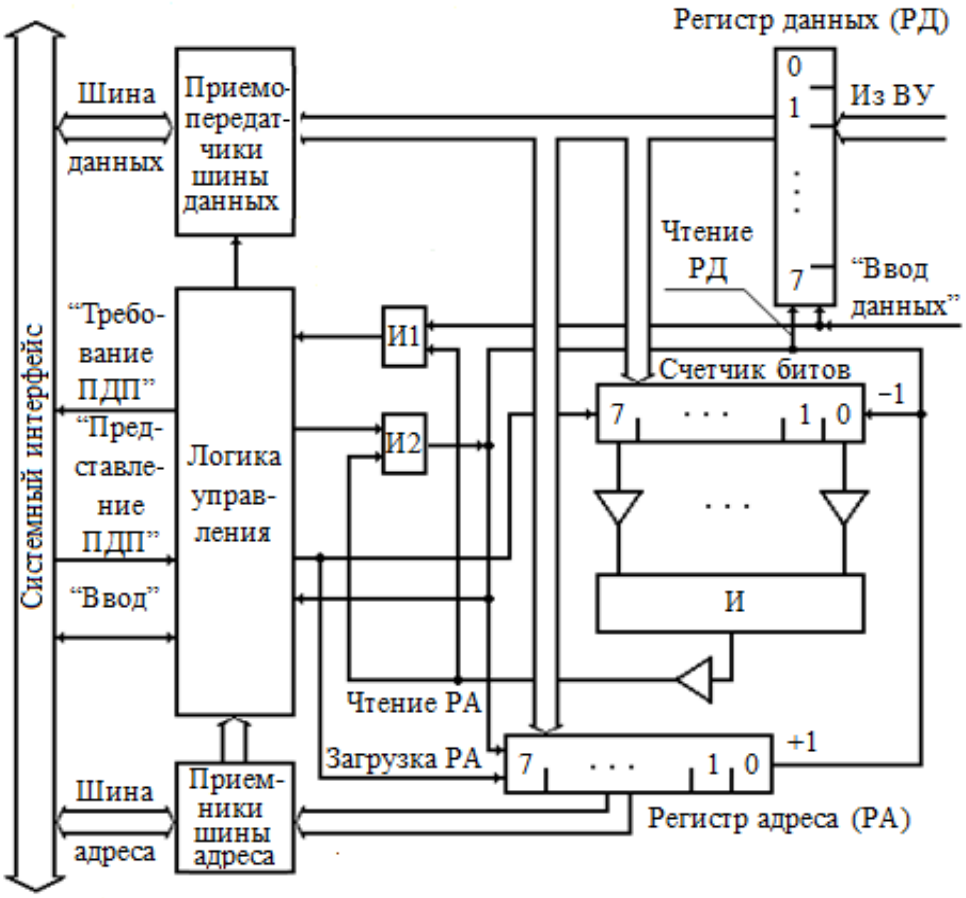


Рис. 1.23. Контроллер ПДП для ввода данных из ВУ в режиме «Захват цикла» и отключением процессора от шин системного интерфейса

Перед началом очередного сеанса ввода данных из ВУ процессор загружает в регистры его контроллера следующую информацию: в счетчик байт – количество принимаемых байт данных, а в регистр адреса – начальный адрес области памяти для вводимых данных. Тем самым контроллер подготавливается к выполнению операции ввода данных из ВУ в память микроЭВМ в режиме ПДП.

Байты данных из ВУ поступают в регистр данных контроллера в постоянном темпе. При этом каждый байт сопровождается управляющим сигналом из ВУ «Ввод данных», который обеспечивает запись байта данных в регистр данных контроллера. По этому же сигналу и

при ненулевом состоянии счетчика байт контроллер формирует сигнал ТПДП. По ответному сигналу процессора ППДП контроллер выставляет на шины адреса и данных системного интерфейса содержимое своих регистров адреса и данных соответственно. Формируя управляющий сигнал «Вывод», контроллер ПДП обеспечивает запись байта данных из своего регистра данных в память микроЭВМ. Сигнал ППДП используется в контроллере и для модификации счетчика байт и регистра адреса.

По каждому сигналу ППДП из содержимого счетчика байт вычитается единица, и как только содержимое счетчика станет равно нулю, контроллер прекратит формирование сигналов «Требование прямого доступа к памяти».

На примере простого контроллера ПДП мы рассмотрели только процесс подготовки контроллера и непосредственно передачу данных в режиме ПДП. На практике любой сеанс обмена данными с ВУ в режиме ПДП всегда инициируется программой, выполняемой процессором, и включает два следующих этапа.

1. На этапе подготовки ВУ к очередному сеансу обмена процессор в режиме программно-управляемого обмена опрашивает состояние ВУ (проверяет его готовность к обмену) и посылает в ВУ команды, обеспечивающие подготовку ВУ к обмену. Такая подготовка может сводиться, например, к перемещению головок на требуемую дорожку в накопителе на жестком диске. Затем выполняется загрузка регистров контроллера ПДП. На этом подготовка к обмену в режиме ПДП завершается и процессор переключается на выполнение другой программы.

2. Обмен данными в режиме ПДП начинается после завершения подготовительных операций в ВУ по инициативе либо ВУ, как это было рассмотрено выше, либо процессора. В этом случае контроллер ПДП необходимо дополнить регистром состояния и управления, содержимое которого будет определять режим работы контроллера ПДП. Один из разрядов этого регистра будет инициировать обмен данными с ВУ. Загрузка информации в регистр состояния и управления контроллера ПДП производится программным путем.

Наиболее распространенным является обмен в режиме прямого доступа к памяти с блокировкой процессора. Он отличается от ПДП с «захватом цикла» тем, что управление системным интерфейсом передается контроллеру ПДП не на время обмена одним байтом, а на время обмена блоком данных. Такой режим ПДП используется в тех слу-

чаях, когда время обмена одним байтом с ВУ сопоставимо с циклом системной шины.

В микроЭВМ можно использовать несколько ВУ, работающих в режиме ПДП. Предоставление таким ВУ шин системного интерфейса для обмена данными производится на приоритетной основе. Приоритеты ВУ реализуются так же, как и при обмене данными в режиме прерывания, но вместо управляющих сигналов «Требование прерывания» и «Предоставление прерывания» используются сигналы «Требование прямого доступа» и «Предоставление прямого доступа», соответственно.

ГЛАВА 2. ПАРАЛЛЕЛЬНЫЕ КОМПЬЮТЕРНЫЕ СИСТЕМЫ

2.1. Классификация параллельных ВС

2.1.1. Потоки команд и потоки данных

Общепринятую удачную классификацию ВС предложил в 1970 г. Г. Флин (США). Основным определяющим архитектурным параметром он выбрал взаимодействие *потока команд* и *потока данных* (операндов и результатов).

В ЭВМ классической архитектуры ведется последовательная обработка команд и данных. Команды поступают одна за другой (за исключением точек ветвления программы), и для них из ОЗУ или регистров так же последовательно поступают операнды. Одной команде (операции) соответствует один необходимый ей набор операндов (как правило, два для бинарных операций). Этот тип архитектуры – «***один поток команд – один поток данных», ОКОД (SISD – «SingleInstruction, SingleData»)*** условно изображен на рис. 2.1.

Рис. 2.1. ВС типа ОКОД (SISD)

Тип ***ОКМД – «один поток команд – много потоков данных» (SIMD – «SingleInstruction – MultipleData»)*** охватывает ВС, в которых одной командой обрабатывается набор данных, множество данных, вектор, и вырабатывается множество результатов. Это векторные и матричные системы, в которых по одной команде выполняется одна и та же операция над всеми элементами массива – вектора или матрицы, распределенными между *процессорными (обрабатывающими) элементами ПЭ или процессорами.* Принцип обработки показан на рис. 2.2.

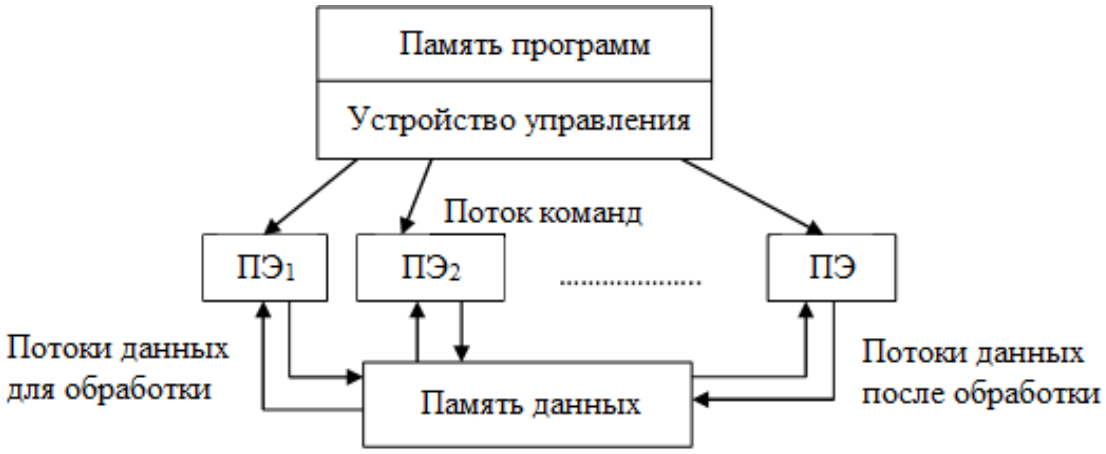


Рис. 2.2. ВС типа ОКМД (SIMD)

Отечественные векторные ВС – ПС-2000, ПС-2100. Допускают организацию матричной обработки. Классический пример матричной архитектуры – ILLIAC-1V (США).

К типу ***МКОД – «много потоков команд – один поток данных» (MISD – «MultipleInstruction – SingleData»)*** принято относить векторный конвейер (обычно в составе ВС, чтобы подчеркнуть основной используемый принцип вычислений), например, в составе ВС *Crey*-1, «Электроника ССБИС». На векторном конвейере производится последовательная обработка одного потока данных многими обрабатывающими устройствами (ступенями, станциями) конвейера.

К такому же типу относится ВС, реализующая *макроконвейер* (ВС «Украина»). В ней задача, решаемая циклически, «разрезается» на последовательные этапы, закрепляемые за отдельными процессорами. Запускается конвейер многократного выполнения цикла, составляющего задачу. Принцип обработки показан на рис. 2.3.

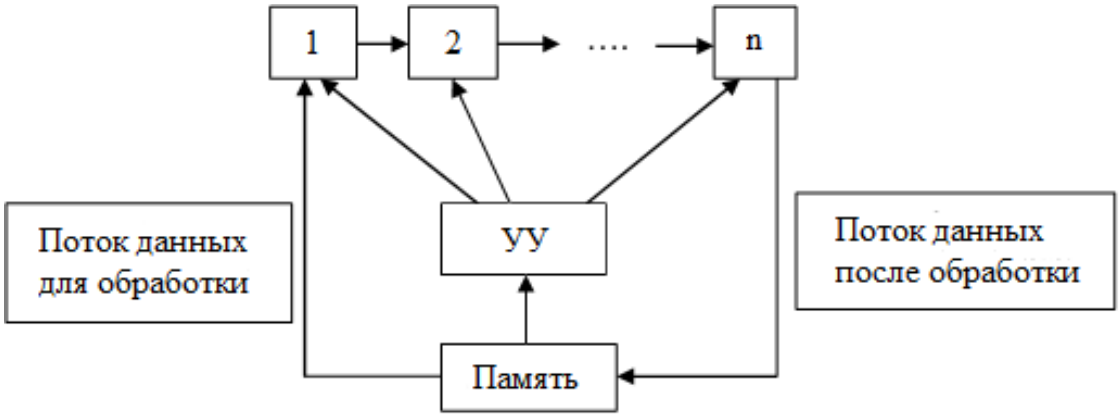


Рис. 2.3. ВС типа МКОД (MISD)

Тип ***МКМД – «много потоков команд – много потоков данных» (MIMD – «MultipleInstruction – MultipleData»)*** соответствует более полному и независимому распараллеливанию. К этому типу относятся, например, все многопроцессорные вычислительные комплексы (МВК) семейства «Эльбрус».

2.1.2. «Фон-Неймановские» и «не-Фон-Неймановские» архитектуры

Немного истории. Первую ЭВМ создал в 1939 г. в США профессор Джон Атанасов, болгарин, со своим аспирантом К. Берри. Две малые ЭВМ, созданные ими в период 1937–1942 гг., были прототипами большой ЭВМ ABC для решения систем линейных уравнений, которая в 1942 г. доводилась по устройствам ввода-вывода и должна была войти в строй в 1943 г., но призыв Атанасова в армию в 1942 г. воспрепятствовал этому. Проект электронной ЭВМ Эниак (ElectronicsNumericalIntegratorandComputer) был сделан в 1942 г. Д. Моучли и Д. Эккертом и осуществлен в 1945 г. в Муровской электротехнической лаборатории Пенсильванского университета. В 1946 г. Эниак был публично продемонстрирован в работе. В нем впервые были применены триггеры. Рождение Эниак считают началом компьютерной эры, посвящая ему научные симпозиумы и другие торжественные мероприятия. Международный симпозиум, посвященный 50-летию первой ЭВМ, был проведен и в Москве в июне 1996 г.

Однако еще в начале 40-х гг. XX века Атанасов поделился с Моучли информацией о принципах, заложенных в ЭВМ ABC. Хотя Моучли впоследствии утверждал, что он не воспользовался этой информацией в патенте на Эниак, суд не согласился с этим. Вернувшись из армии после войны, Атанасов узнал, что более мощная ЭВМ Эниак уже создана, и потерял интерес к этой теме, не поинтересовавшись, насколько Эниак похож на его ЭВМ ABC.

Известный английский математик Алан Тьюринг был не только теоретиком по информации и теории алгоритмов, автором теоретического автомата «машины Тьюринга», но и талантливым инженером, создавшим в начале 1940-х гг. первую работающую специализированную ЭВМ.

Эта ЭВМ под названием «Колосс» была сконструирована и сделана им совместно с Х. А. Ньюменом в Блетчи (Англия) и начала работать в 1943 г. Сообщения о ней своевременно не публиковались,

т. к. она использовалась для расшифровки секретных германских кодов во время войны.

Основные архитектурно-функциональные принципы построения ЦВМ были разработаны и опубликованы в 1946 г. венгерским математиком и физиком Джоном фон Нейманом и его коллегами Г. Голдстайном и А. Берксом в ставшем классическим отчете «Предварительное обсуждение логического конструирования электронного вычислительного устройства». Основополагающими принципами ЭВМ на основании этого отчета являются: 1) принцип программного управления выполнением программы и 2) принцип хранимой в памяти программы. Они легли в основу понятия ***фон-Неймановской архитектуры***, широко использующей *счетчик команд*.

Вернемся к настоящему. Счетчик команд отражает «узкое горло», которое ограничивает поток команд, поступающих на исполнение, их последовательным анализом.

Альтернативной архитектурой является ***«не-фон-Неймановская» архитектура***, допускающая одновременный анализ более одной команды. Поиски ее обусловлены необходимостью распараллеливания выполнения программы между несколькими исполнительными устройствами – процессорами. Счетчик команд при этом не нужен. Порядок выполнения команд определяется наличием исходной информации для выполнения каждой из них. Если несколько команд готовы к выполнению, то принципиально возможно их назначение для выполнения таким же количеством свободных процессоров. Говорят, что такие ВС управляются *потоком данных (dataflow)*.

В командах проверки условия возможно альтернативное задание адреса результата (ИЛИ – ИЛИ). Адреса результатов являются адресами ПК, т. е. результаты выполнения одних команд в качестве операндов могут поступать в текст других команд. Команда не готова к выполнению, если в ее тексте отсутствует хотя бы один операнд. Ячейка, обладающая полным набором операндов, переходит в возбужденное состояние и передает в селекторную сеть информационный пакет (токен), содержащий код операции и необходимую числовую и связную информацию. Он поступает по сети на одно из исполнительных устройств. Там же операция выполняется, и в распределительную сеть выдается результирующий пакет, содержащий результат вычислений и адреса назначения в ПК (возможно, за счет выбора альтернативы, т. е. такой выбор – тоже результат). По этим адресам в ПК результат и поступает, создавая возможность активизации новых ячеек. После выдачи токена в селекторную сеть операнды в тексте команды

уничтожаются, что обеспечивает повторное выполнение команды в цикле, если это необходимо.

Общая схема потоковых ВС представлена на рис. 2.4.

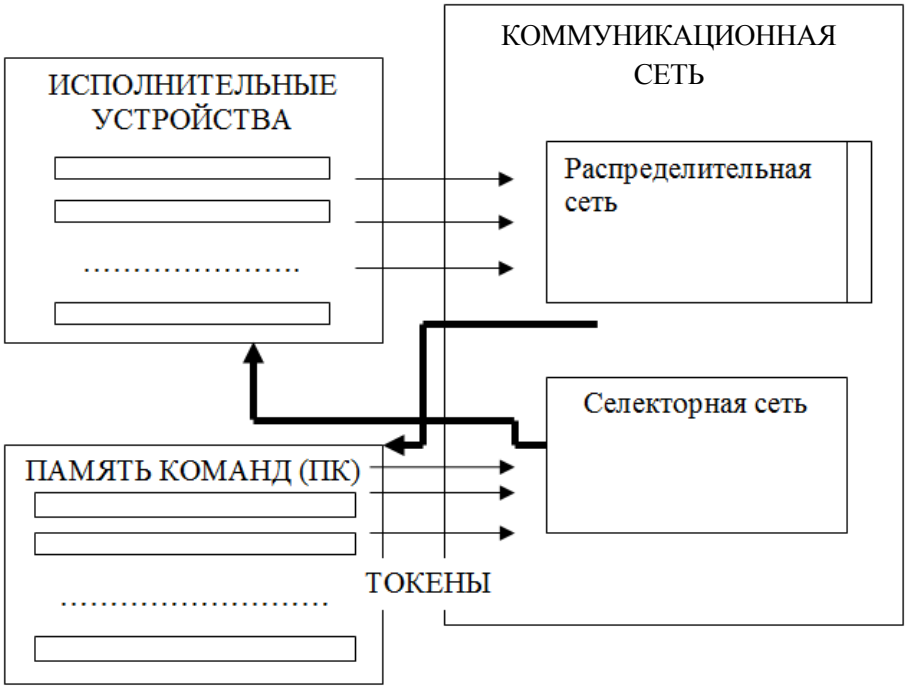


Рис. 2.4. «Идеальная» потоковая ВС

Программа или ее часть (сегмент) размещается в памяти команд ПК, состоящей из ячеек команд. Команды имеют слудующую структуру.

{код операции, операнд 1, ..., операнд L,
адрес результата 1, ..., адрес результата M}

Селекторная и распределительная сети образуют ***коммуникационную сеть*** ВС.

Ожидаемая сверхвысокая производительность такой системы может быть достигнута за счет одновременной и независимой активизации большого числа готовых команд, проблематичном допущении о бесконфликтной передаче пакетов по сетям и параллельной работы многих исполнительных устройств.

Существует ряд трудностей, в силу которых «не-фон-Неймановские» архитектуры не обрели технического воплощения для массового применения в «классическом», отраженном выше, исполнении. Однако многие устройства используют данный принцип, но чаще всего взаимодействие процессоров при совместном решении общей задачи и их синхронизация при использовании общих данных

основаны на анализе готовности данных для их обработки. Это дает основание многим конструкторам заявлять, что в своих моделях они реализовали принцип *dataflow*.

2.1.3. Системы с общей и распределенной памятью

Системы с общей (разделяемой) оперативной памятью образуют современный класс ВС – многопроцессорных супер-ЭВМ. Одинаковый доступ всех процессоров к программам и данным представляет широкие возможности организации параллельного вычислительного процесса *(параллельных вычислений)*. Отсутствуют потери реальной производительности на межпроцессорный (между задачами, процессами и т. д.) обмен данными (рис. 2.5, *а*).

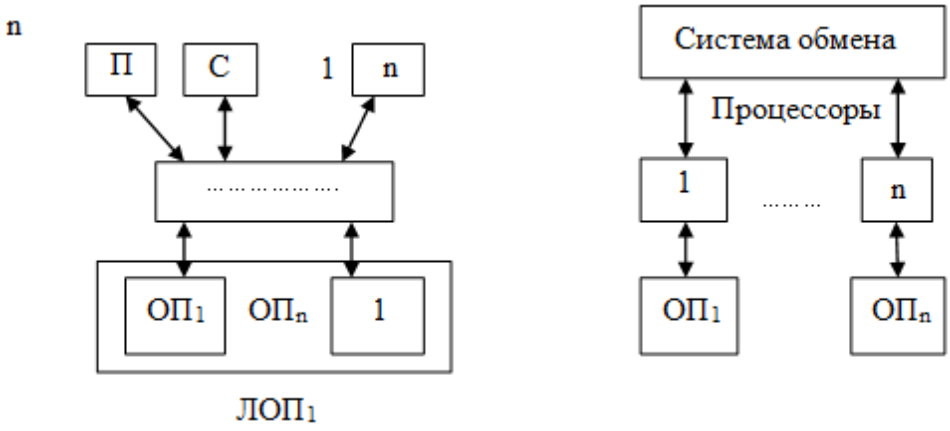


Рис. 2.5. ВС с общей (*а*) и распределенной (*б*) памятью

Системы с распределенной памятью образуют вычислительные комплексы (ВК) – коллективы ЭВМ с межмашинным обменом для совместного решения задач (рис. 2.5, *б*). В ВК объединяются вычислительные средства систем управления, решающие специальные наборы задач, взаимосвязанных по данным. Принято говорить, что такие ВК выполняют *распределенные вычисления*, а сами ВК называют *распределенными ВК*.

Другое, противоположное воплощение принципа МИМД – *масспроцессорные* или высокопараллельные архитектуры, объединяющие сотни – тысячи – десятки тысяч процессоров.

В современных супер-ЭВМ наметилась тенденция объединения двух принципов: общей (распределяемой) и распределенной (локаль-

ной) оперативной памяти (ЛОП). Такая структура используется в проекте МВК «Эльбрус-3» и «Эльбрус-3М» (рис. 2.6).

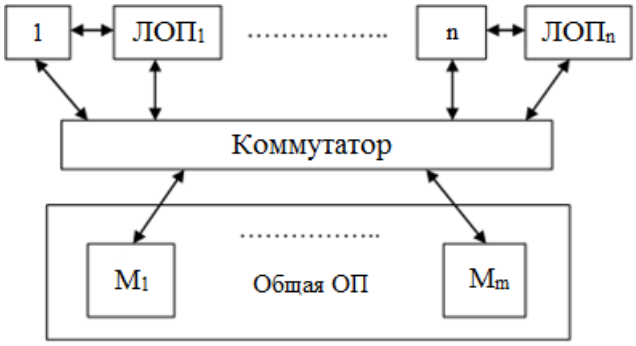


Рис. 2.6. Схема ВС с модулями локальной памяти

2.2. Способы межмодульного соединения (комплексирования)

Различают два противоположных способа комплексирования: *с общей шиной (шинная архитектура) и с перекрестной (матричной) коммутацией* модулей ВС (процессоров, модулей памяти, периферии). На рис. 2.7 представлена система с общей шиной. Шина состоит из линий, по которым передаются информационные и управляющие сигналы.

Шина используется в режиме разделения времени, при котором лишь один модуль в данный момент работает на передачу. Принимать принципиально могут все модули, хотя преимущественно информация при выдаче в нее адресуется. Применяется в микро- и мини-ЭВМ при сравнительно небольшом числе модулей. Практически производится разделение шины на управляющую, адресную и шину данных.

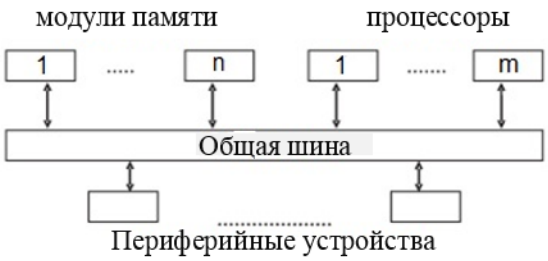


Рис. 2.7. Схема ВС с общей шиной

В высокопроизводительных ВС для возможности одновременного обмена многими парами абонентов используется перекрестная или матричная коммутация.

Матричный коммутатор можно представить (прямоугольной) сеткой шин. К одному концу каждой подсоединен источник-потребитель информации (рис. 2.8). Точки пересечения – *узлы* этой сетки – представляют собой *управляющие ключи*, которые соединяют или разъединяют соответствующие шины, устанавливая или прекращая связь между модулями. Реализуется связь «каждый с каждым». Одновременно могут связываться многие (до $n/2$) пары модулей. На рис. 2.8, *а* – перекрестная связь между процессорами в ВС с распределенной памятью, на рис. 2.8, *б* – между n процессорами и m модулями ОП.

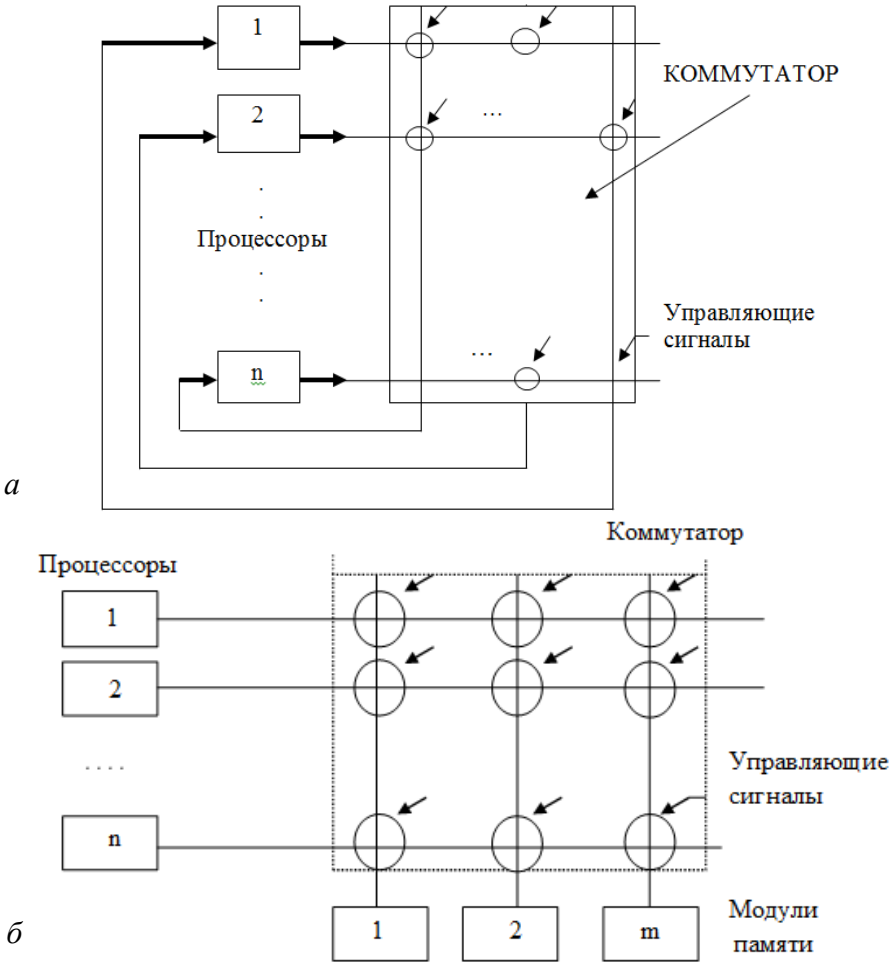


Рис. 2.8. Матричные коммутаторы:
а – перекрестная коммутация процессоров,
б – коммутация процессоров и модулей памяти

Кроме рассмотренных, широкое распространение в параллельных системах получили другие разновидности коммутационных сетей – сеть Бенеша, омега-сеть и др.

Учебное издание

Кобайло Александр Серафимович
Жиляк Надежда Александровна

АРИФМЕТИКО-ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН И АРХИТЕКТУРА КОМПЬЮТЕРОВ

Учебно-методическое пособие

Редактор *Ю. Д. Нежикова*
Компьютерная верстка *К. В. Лактысева*
Корректор *Ю. Д. Нежикова*

Издатель и полиграфическое исполнение:
УО «Белорусский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий
№ 1/227 от 20.03.2014.
Ул. Свердлова, 13а, 220006, г. Минск.