Криворожский колледж
Национального авиационного университета
«КРАУСС»

Конспект лекций
по учебной дисциплине
«Цифровые устройства и микропроцессоры»
Часть II

Курсанта _____ группы

_____

Кривой Рог
2006

Пономарев И.А., Панченко Ю.Н. Конспект лекций по учебной дисциплине «Цифровые устройства и микропроцессоры» , часть II: Учебное пособие для учебных заведений гражданской авиации I-III уровня аккредитации. – Кривой Рог: КК НАУ, 2006. – 85 с.

Рецензенты: Бакулин Е.В., Сергеев Н.Г.

Рекомендован цикловой комиссией «Радиоэлектронного оборудования наземных средств обеспечения полетов» КК НАУ «КРАУСС» для проведения лекционных занятий для специальностей 5.090.705 «Техническая эксплуатация радиоэлектронного оборудования воздушных судов», 5.090.706 «Техническая эксплуатация радиоэлектронного оборудования наземных средств обеспечения полетов», 6.090.700 «Радиоэлектронные устройства, системы и комплексы» протоколом №1 от 30 августа 2005 г.

Появление ЭВМ не отмечалось сполохами ядерных
взрывов. Об этом долго не писалось в газетах.
Наступление новой эры проходило в тиши сильно
засекреченных исследовательских лабораторий.
И даже создатели первых ЭВМ не отдавали себе
отчета в значении того, что они делали.
Н. Н. Моисеев, академик.

Это маленькое чудо, и я думаю,
что микропроцессор окажет
огромное влияние на развитие
вычислительной техники.
По существу, мы теперь начинаем
на все смотреть другими глазами.
Д. Слотник,
создатель сети ARPA
и супервычислителя ILLIAC-4.

Введение

Примерно за 25 лет до того, как корпорация Intel выпустила первый в мире функционально завершенный однокристальный микропроцессор (МП) 4004, положив тем самым начало микропроцессорному буму, произошло событие, радикально изменившее наш мир, был создан транзистор (1946 г.).

Транзистор является неотъемлемой частью любой микросхемы, а также МП.

Возможность интеграции нескольких полупроводниковых элементов на одной подложке дала возможность созданию малых интегральных схем. Эти элементы соединялись между собой напыляемыми металлическими полосками (рис. 1).

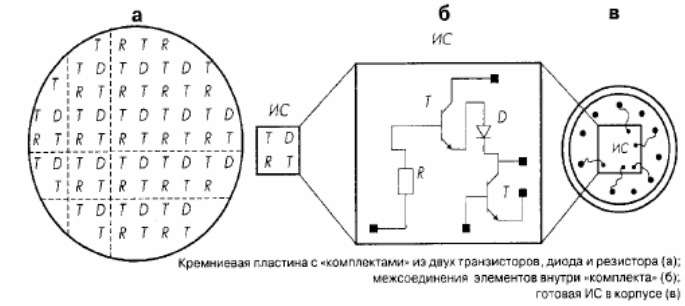


Рис. !

Далее пластина разрезалась на отдельные кристаллы, которые помещались в корпус. В результате получался готовый функциональный узел. Первые микросхемы содержали от единиц до нескольких десятков элементов. Это были малые интегральные схемы. Очень скоро появились приборы средней степени интеграции 100-1000 элементов.

В течение 80-х годов сменилось два поколения микросхем: большие - до 10000 элементов на кристалл и сверхбольшие - до 100 тысяч элементов. В начале 90-х ультра большие микросхемы содержали до 1 млн. транзисторов, а МП Pentium Pro 200 MHz - 5,5 млн. транзисторов.

Технология больших интегральных схем позволила получать весьма крупные арифметико-логические устройства, или компоненты, например микропроцессоры.

История создала все предпосылки для появления МП. Долго ждать не пришлось.

28 лет назад компания Intel начала выпуск семейства интегральных микросхем 4004 (комплект Intel MCS-4), предназначенных для построения программируемых калькуляторов.

Это были первые вычислительные приборы, которые обеспечивали параллельную обработку под микропрограммным управлением. В состав семейства входил 4-разрядный микропроцессор со всеми присущими ему достоинствами: универсальность, гибкость, миниатюрность, микромощность, низкая стоимость. Такой прибор выполнял 18 команд, имел период машинного цикла 10,8 мкс при работе по стандартной двухфазной системе синхроимпульсов.

Адресное пространство ограничивалось 4 КВ ROM, 1 КВ RAM и 128 каналами ввода-вывода (I/O).

Микропроцессор возник не на пустом месте, его появлению предшествовало десятилетие развития микроэлектроники, которая к началу 70-х годов добилась существенных успехов в создании интегральных

микросхем. И все же выход 4004-го стал революционным событием, приведшим к коренной ломке представлений о возможностях микроэлектронной и вычислительной техники. Вычислительная техника вступила в бурную микропроцессорную эру, которая характеризуется гонкой технологий и высокой скоростью совершенствования технического уровня изделий. Преимущества микропроцессоров были мгновенно оценены, и буквально в считанные месяцы к выпуску аналогичных БИС приступили другие компании.

Вопрос о приоритете создания первого в мире микропроцессора уже к 1974 г. полностью прояснился. Вот как писал об этом журнал "Electronics" в 1974 г. :"Intel определенно принадлежит честь использования идеи микропроцессора, и она была первой, кто выпустил их на рынок, хотя большие заслуги принадлежат также другим фирмам и отдельным лицам, которые тем или иным путем содействовали развитию техники больших интегральных схем.

Следует вспомнить компанию Viatron, которая в 1968 г удивила мир, объявив о своем намерении создать систему обработки данных на основе собственного 8-разрядного микропроцессора, управляемого примитивной программой, записанной в ПЗУ. Однако эта компания встретилась с серьезными финансовыми трудностями и спустя два года обанкротилась. Между тем, General E1ectric разработала однокристальный базовый логический блок и вплотную подошла к созданию микропроцессора".

Уже в 1971 г. Intel не скрывала своих достижений. Она существовала всего четыре года и была известна микросхемами памяти, когда в шумной рекламной кампании провозгласила "новую эру в микроэлектронике", выпустив на рынок первый коммерческий микропроцессор.

На сей раз реклама оказалась пророчеством, а не беззастенчивым обманом покупателей, как это иногда случается. Один кристалл 4004 обладал почти такими же вычислительными возможностями, как и ENIAC выпуска 1946 г. с его 18 000 электронных ламп. Микропроцессор был дешевым (около $200) и имел крошечные размеры, что позволяло разработчикам создавать принципиально новые системы. Скептики предсказывали, что рынок однокристального процессора будет такой же крошечный, как и он сам, но действительность превзошла все ожидания.

Идея 4004-го возникла у одного из основателей компании Intel Роберта Нойса (Robert Noyce). Он предложил использовать технологический процесс изготовления кристаллов памяти для создания логических чипов. Но какими именно должны быть эти логические устройства? Этот вопрос решил счастливый случай. В 1969 г. японская компания Busicom заказала Intel разработку комплекта из 12 специализированных чипов для настольных калькуляторов. Вместо этого Intel предложила строить калькуляторы на основе однокристального микропроцессора, который впоследствии приобрел известность как Intel 4004. Конструкторскую разработку выполнили Тэд Хофф (Ted Hoff), Стэн Мэйзор (Stan Mazor) и Федерико Фэджин (Federico Faggin). Они позаимствовали многие решения из архитектуры больших ЭВМ. Однако чтобы разместить процессор на одном кристалле, им пришлось снизить его разрядность до четырех. Это позволило уменьшить количество транзисторов и упаковать процессор в 16-контактный корпус - самый большой в то время (для сравнения, современный Pentium имеет 64-разрядную внешнюю шину и помещается в 296-контактном корпусе). Процессор мог выполнять 45 инструкций, и для программируемого калькулятора компании Busicom к комплекту MCS-4 достаточно было добавить четыре кристалла памяти по 256 байт. Процессор имел 2300 транзисторов, 4-разрядную шину данных и 12-разрядную шину адреса (шины мультиплексировались) и тактовую частоту 108 kHz. Это позволяло ему выполнять 0,06 млн операций в секунду (MIPS).

15 ноября 1971 г. можно считать началом новой эры в электронике. В этот день компания приступила к поставкам первого в мире микропроцессора Intel 4004.

Это был настоящий прорыв, ибо МП Intel-4004 размером менее 3 см был производительнее гигантской машины ENIAC. Правда работал он гораздо медленнее и мог обрабатывать одновременно только 4 бита информации (процессоры больших ЭВМ обрабатывали 16 или 32 бита одновременно), но и стоил первый МП в десятки тысяч раз дешевле.

Микропроцессор (МП) представляет собой функционально завершенное универсальное программно-управляемое устройство цифровой обработки данных, выполненное в виде одной или нескольких микропроцессорных БИС.

Микросистема (МС) - цифровое устройство или система обработки данных, контроля и управления, построенная на базе одного или нескольких МП.

Чип 4004 открыл дорогу более мощным и быстродействующим микропроцессорам.

Как только разработка поступила в серийное производство, Тэд Хофф и Стэн Мэйзор приступили к созданию его 8-разрядной версии - 8008. Это был первый микропроцессор, имевший систему прерываний и мультиплексированную шину адрес-данные, но функционировал он неудовлетворительно. И уже в 1974 г. появился, также 8-разрядный, кристалл 8080. Он содержал 6000 транзисторов, выполнял более 250 команд, причем некоторые из них работали с парами регистров - с 16-разрядными данными. Его адресное пространство 64 KB по тем временам было огромным (современный Pentium Pro может адресоваться к 64 TB виртуальной памяти).

С выходом 8-разрядных чипов Intel открылись широчайшие перспективы для разработки дешевого программируемого оборудования. И события не заставили себя ждать. Вскоре Гари Килдэлл (Gary Kildall) из Digital Research создал для процессора 8080 операционную систему CP/M. Этот программный продукт

значительно упростил процесс разработки и отладки прикладного программного обеспечения и, вместе с чипом 8080, фактически открыл невиданные ранее возможности создания доступных массовому потребителю компьютеров.

Первым персональным компьютером принято считать микроЭВМ Altair 8800 компании MITS, которая поступила в продажу в 1975 г. и уже была оснащена операционной системой CP/M. Машина выпускалась в виде набора "Сделай сам" или как законченное изделие и предназначалась для любителей сложной технической игрушки. В дальнейшем появился расширенный вариант устройства для бухгалтерских и инженерных расчетов.

Многие компании с середины 70-х годов приступили к производству личных, домашних, любительских микроЭВМ, но главный успех достался компании Apple, выпустившей в 1977 г. одноименную ПЭВМ, которая отличалась совершенным программным обеспечением, рассчитанным на самого неподготовленного пользователя. Дружественное программное обеспечение (friendly software) расширило круг потенциальных покупателей, ограниченный ранее любителями электроники и профессионалами.

Микропроцессорная гонка набирала все большее ускорение. Уже упоминавшийся Федерико Фэджин и Масатоши Шима (Masatoshi Shima) основали компанию Zilog, которая молниеносно наладила выпуск чипа Z80 - несколько усовершенствованного аналога 8080, совместимого и с ним, и с операционной системой CP/M. Микропроцессор Z80 имел встроенную программу регенерации динамического ОЗУ и вместе с CP/M составил "идеальную пару" для массового производства дешевых ПК.

Вскоре дебютировала Motorola со своим 6800-м, стартовали Texas Instruments, National Semiconductor и Fairchild, разработав свои собственные микропроцессоры; большинство из них использовалось для встраивания в "интеллектуальные" приборы.

Наконец, на новый и чрезвычайно динамичный сектор компьютерного рынка обратила внимание корпорация IBM - гигант вычислительной индустрии. Выпустив в 1981 г. свой первый IBM PC, она сразу стала законодателем мод для производителей персоналок и поныне занимает лидирующее место на всемирном компьютерном рынке.

Усовершенствованный 16-разрядный чип появился в 1978 г. одновременно в "двух лицах" - 8086 и 8088. Первый чип содержал 29000 транзисторов, более развитую систему команд, имел 16-разрядные внешнюю и внутреннюю шины данных, 20-разрядную шину адреса и работал на частоте 5 MHz (некоторые версии имели частоту до 10 MHz). Его производительность в 10 раз превышала этот показатель своего предшественника (0,33 MIPS, но на 16-разрядной архитектуре).

Строго придерживаясь принципа программной совместимости, Intel сохранила 16-разрядный адресный регистр, присущий чипу 8080, но для расширения адресного пространства до 1 MB добавила сегментный регистр, тем самым создав новую идеологию адресации, которая применяется и сегодня.

Процессор 8086 уже имел примитивный конвейер: блок интерфейса с шиной (Bus Interface Unit) поставлял поток инструкций исполнительному блоку (Execution Unit) из 6-байтовой очереди, так что операции выборки и исполнения осуществлялись одновременно. Это был истинно 16-разрядный процессор, в отличие от следующей модели — 8088 (с математическим сопроцессором 8087), которая вышла спустя год.

Процессор 8088 совпадал со своим предшественником во всем, кроме того, что имел 8-разрядную внешнюю шину данных. Это позволяло строить недорогие системы, используя 8-разрядные наборы чипов серии 8085. Именно на процессор 8088 обратила, наконец, внимание, корпорация IBM — флагман компьютерной индустрии. В октябре 1981 г. Голубой Гигант предложил рынку первую IBM PC, а уже к концу года было продано более 35 тыс. компьютеров.

Эти близнецы обрели мировую известность благодаря вычислительным системам класса Extended Technology - знаменитым PC XT компании IBM с прекрасным DOS-ом и незабываемой оболочной Norton Commander. Архитектура 8086/8088 превратилась в промышленный стандарт для персональных компьютеров.

Достигнутые к началу 70-х гг. успехи в области технологии интегральных микросхем и организации вычислительных устройств привели к появлению нового класса приборов — микропроцессоров. Сегодня микропроцессорная техника — индустриальная отрасль со своей методологией и средствами проектирования.

К настоящему времени накоплен большой практический опыт проектирования микропроцессоров и микропроцессорных систем, область применения которых постоянно расширяется.

1. Классификация микропроцессоров

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), БИС управляющего (УП) и БИС интерфейсного (ИП) процессоров.

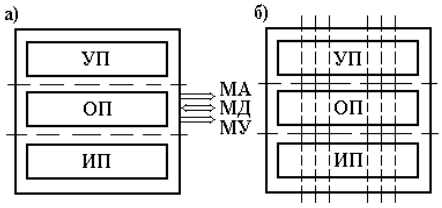


Рис. 1.1 Функциональная структура процессора (а) и ее разбиение для реализации процессора в виде комплекта секционных БИС.

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 всего двух серий данных: входных, передаваемых формой сигнала, и фиксированных опорных и к определению их подобия. 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 преобразования. 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.

2. Архитектура микропроцессора

2.1 Основные характеристики микропроцессора

Микропроцессор характеризуется:

1) тактовой частотой, определяющей максимальное время выполнения переключения элементов в ЭВМ;

2) разрядностью, т.е. максимальным числом одновременно обрабатываемых двоичных разрядов.

Разрядностть МП обозначается m/n/k/ и включает:

m - разрядность внутренних регистров, определяет принадлежность к тому или иному классу процессоров;

n - разрядность шины данных, определяет скорость передачи информации;

k - разрядность шины адреса, определяет размер адресного пространства. Например, МП i8088 характеризуется значениями m/n/k=16/8/20;

3) архитектурой. Понятие архитектуры микропроцессора включает в себя систему команд и способы адресации, возможность совмещения выполнения команд во времени, наличие дополнительных устройств в составе микропроцессора, принципы и режимы его работы. Выделяют понятия микроархитектуры и макроархитектуры.

Микроархитектура микропроцессора - это аппаратная организация и логическая структура микропроцессора, регистры, управляющие схемы, арифметико-логические устройства, запоминающие устройства и связывающие их информационные магистрали.

Макроархитектура - это система команд, типы обрабатываемых данных, режимы адресации и принципы работы микропроцессора.

В общем случае под архитектурой ЭВМ понимается абстрактное представление машины в терминах основных функциональных модулей, языка ЭВМ, структуры данных.

Архитектура типичной небольшой вычислительной системы на основе микроЭВМ показана на рис. 2.1 Такая микроЭВМ содержит все 5 основных блоков цифровой машины: устройство ввода информации, управляющее устройство (УУ), арифметико-логическое устройство (АЛУ) (входящие в состав микропроцессора), запоминающие устройства (ЗУ) и устройство вывода информации.

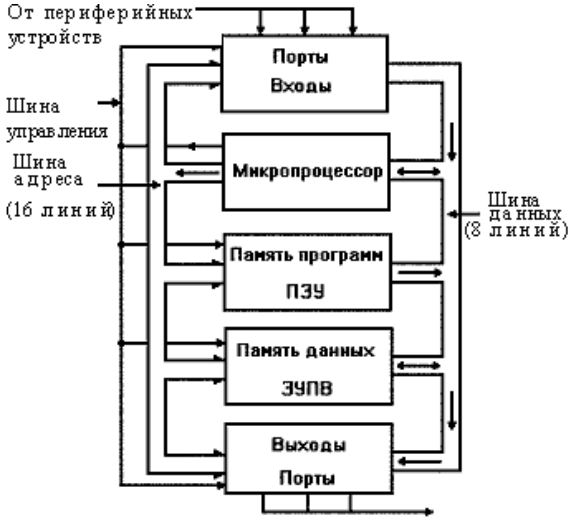


Рис. 2.1. Архитектура типового микропроцессора.

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. 2.2 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.

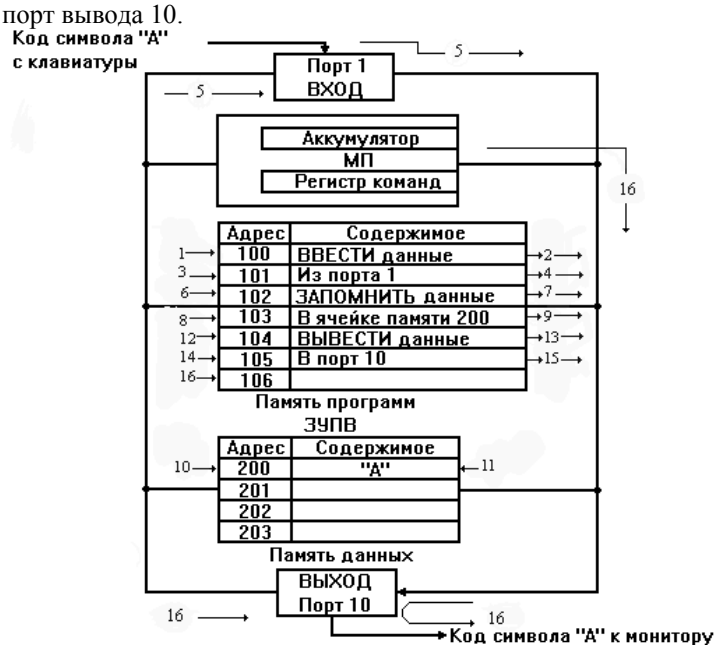


Рис. 2.2. Диаграмма выполнения процедуры ввода-запоминания-вывода.

В данной программе всего три команды, хотя на рис. 2.2 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. Код операции и операнд размещаются в отдельных ячейках памяти программ. На рис. 2.2 КОП хранится в ячейке 100, а код операнда - в ячейке 101 (порт 1); последний указывает откуда нужно взять информацию.

В МП на рис. 2.2 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.

2.2 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. 2.3.

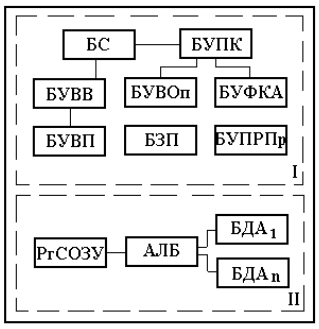


Рис. 2.3. Общая логическая структура микропроцессора: I - управляющая часть, II - операционная часть; БУПК - блок управления последовательностью команд; БУВОп - блок управления выполнением операций; БУФКА - блок управления формированием кодов адресов; БУВП - блок управления виртуальной памятью; БЗП - 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.

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 если не оптимальных, то 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. 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.

2.3 Устройство управления

Коды операции команд программы, воспринимаемые управляющей частью микропроцессора, расшифрованные и преобразованные в ней, дают информацию о том, какие операции надо выполнить, где в памяти расположены данные, куда надо направить результат и где расположена следующая за выполняемой команда.

Управляющее устройство имеет достаточно средств для того, чтобы после восприятия и интерпретации информации, получаемой в команде, обеспечить переключение (срабатывание) всех требуемых функциональных частей машины, а также для того, чтобы подвести к ним данные и воспринять полученные результаты. Именно срабатывание, т. е. изменение состояния двоичных логических элементов на противоположное, позволяет посредством коммутации вентилей выполнять элементарные логические и арифметические действия, а также передавать требуемые операнды в функциональные части микроЭВМ.

Устройство управления в строгой последовательности в рамках тактовых и цикловых временных интервалов работы микропроцессора (такт - минимальный рабочий интервал, в течение которого совершается одно элементарное действие; цикл - интервал времени, в течение которого выполняется одна машинная операция) осуществляет: выборку команды; интерпретацию ее с целью анализа формата, служебных признаков и вычисления адреса операнда (операндов); установление номенклатуры и временной последовательности всех функциональных управляющих сигналов; генерацию управляющих импульсов и передачу их на управляющие шины функциональных частей микроЭВМ и вентили между ними; анализ результата операции и изменение своего состояния так, чтобы определить месторасположение (адрес) следующей команды.

2.4 Особенности программного и микропрограммного управления

В микропроцессорах используют два метода выработки совокупности функциональных управляющих сигналов: программный и микропрограммный.

Выполнение операций в машине сводится к элементарным преобразованиям информации (передача информации между узлами в блоках, сдвиг информации в узлах, логические поразрядные операции, проверка условий и т.д.) в логических элементах, узлах и блоках под воздействием функциональных управляющих сигналов блоков (устройств) управления. Элементарные преобразования, неразложимые на более простые, выполняются в течение одного такта сигналов синхронизации и называются микрооперациями.

В аппаратных (схемных) устройствах управления каждой операции соответствует свой набор логических схем, вырабатывающих определенные функциональные сигналы для выполнения микроопераций в определенные моменты времени. При этом способе построения устройства управления реализация микроопераций достигается за счет однажды соединенных между собой логических схем, поэтому ЭВМ с аппаратным устройством управления называют ЭВМ с жесткой логикой управления. Это понятие относится к фиксации системы команд в структуре связей ЭВМ и означает практическую невозможность каких-либо изменений в системе команд ЭВМ после ее изготовления.

При микропрограммной реализации устройства управления в состав последнего вводится ЗУ, каждый разряд выходного кода которого определяет появление определенного функционального сигнала управления. Поэтому каждой микрооперации ставится в соответствие свой информационный код - микрокоманда. Набор микрокоманд и последовательность их реализации обеспечивают выполнение любой сложной операции. Набор микроопераций называют микропрограммами. Способ управления операциями путем последовательного считывания и интерпретации микрокоманд из ЗУ (наиболее часто в виде микропрограммного ЗУ используют быстродействующие программируемые логические матрицы), а также использования кодов микрокоманд для генерации функциональных управляющих сигналов называют

микропрограммным, а микроЭВМ с таким способом управления - микропрограммными или с хранимой (гибкой) логикой управления.

К микропрограммам предъявляют требования функциональной полноты и минимальности. Первое требование необходимо для обеспечения возможности разработки микропрограмм любых машинных операций, а второе связано с желанием уменьшить объем используемого оборудования. Учет фактора быстродействия ведет к расширению микропрограмм, поскольку усложнение последних позволяет сократить время выполнения команд программы.

Преобразование информации выполняется в универсальном арифметико-логическом блоке микропроцессора. Он обычно строится на основе комбинационных логических схем.

Для ускорения выполнения определенных операций вводятся дополнительно специальные операционные узлы (например, циклические сдвигатели). Кроме того, в состав микропроцессорного комплекта (МПК) БИС вводятся специализированные оперативные блоки арифметических расширителей.

Операционные возможности микропроцессора можно расширить за счет увеличения числа регистров. Если в регистровом буфере закрепление функций регистров отсутствует, то их можно использовать как для хранения данных, так и для хранения адресов. Подобные регистры микропроцессора называются регистрами общего назначения (РОН). По мере развития технологии реально осуществлено изготовление в микропроцессоре 16, 32 и более регистров.

В целом же, принцип микропрограммного управления (ПМУ) включает следующие позиции:
1) любая операция, реализуемая устройством, является последовательностью элементарных действий - микроопераций;
2) для управления порядком следования микроопераций используются логические условия;
3) процесс выполнения операций в устройстве описывается в форме алгоритма, представляемого в терминах микроопераций и логических условий, называемого микропрограммой;
4) микропрограмма используется как форма представления функции устройства, на основе которой определяются структура и порядок функционирования устройства во времени.

ПМУ обеспечивает гибкость микропроцессорной системы и позволяет осуществлять проблемную ориентацию микро- и миниЭВМ.

2.5 Система команд

Система команд (состав команд) — это список операций, которые МП может выполнить. Она включает в себя передачу данных, арифметические и логические операции, команды тестирования данных и ветвлений, операции ввода/вывода (ВВ). В то же время команды используют различные способы адресации.

Проектирование системы команд оказывает влияние на структуру ЭВМ. Оптимальную систему команд иногда определяют как совокупность команд, которая удовлетворяет требованиям проблемно-ориентированных применений таким образом, что избыточность аппаратных и аппаратно-программных средств на реализацию редко используемых команд оказывается минимальной. В различных программах ЭВМ частота появления команд различна; например, по данным фирмы DEC в программах для ЭВМ семейства PDP-11 наиболее часто встречается команда передачи MOV(B), на ее долю приходится приблизительно 32% всех команд в типичных программах. Систему команд следует выбирать таким образом, чтобы затраты на редко используемые команды были минимальными.

При наличии статистических данных можно разработать (выбрать) ЭВМ с эффективной системой команд. Одним из подходов к достижению данной цели является разработка команд длиной в одно слово и кодирование их таким образом, чтобы разряды таких коротких команд использовать оптимально, что позволит сократить время реализации программы и ее длину.

Другим подходом к оптимизации системы команд является использование микроинструкций. В этом случае отдельные биты или группы бит команды используются для кодирования нескольких элементарных операций, которые выполняются в одном командном цикле. Эти элементарные операции не требуют обращения к памяти, а последовательность их реализации определяется аппаратной логикой.

Сокращение времени выполнения программ и емкости памяти достигается за счет увеличения сложности логики управления.

Важной характеристикой команды является ее формат, определяющий структурные элементы команды, каждый из которых интерпретируется определенные образом при ее выполнении. Среди таких элементов (полей) команды выделяют следующие: код операции, определяющий выполняемое действие; адрес ячейки памяти, регистра процессора, внешнего устройства; режим адресации; операнд при использовании непосредственной адресации; код анализируемых признаков для команд условного перехода.

Классификация команд по основным признакам представлена на рис. 2.4. Важнейшим структурным элементом формата любой команды является код операции (КОП), определяющей действие, которое должно быть выполнено. Большое число КОП в процессоре очень важно, так как аппаратная реализация команд экономит память и время. Но при выборе ЭВМ необходимо концентрировать внимание на полноте операций с конкретными типами данных, а не только на числе команд, на доступных режимах адресации. Число бит, отводимое под КОП, является функцией полного набора реализуемых команд.

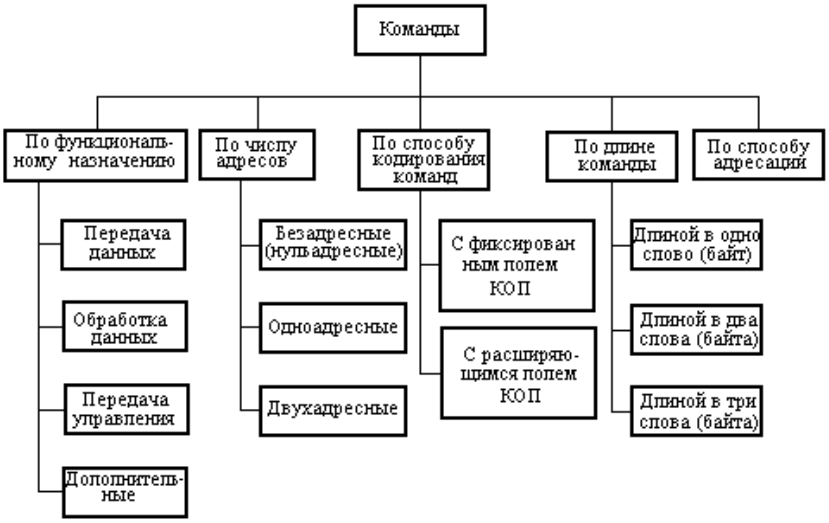


Рис. 2.4. Классификация команд.

При использовании фиксированного числа бит под КОП для кодирования всех m команд необходимо в поле КОП выделить n двоичных разрядов, так чтобы $2^n = m$. Однако, учитывая ограниченную длину слова мини- и микроЭВМ, различное функциональное назначение команд, источники и приемники результатов операций, а также то, что не все команды содержат адресную часть для обращения к памяти и периферийным устройствам, в малых ЭВМ для кодирования команд широко используется принцип кодирования с переменным числом бит под поле КОП для различных групп команд.

В некоторых командах необходим только один операнд и они называются однооперандными (или одноадресными) командами в отличие от двухоперандных (или двухадресных), в которых требуются два операнда. При наличии двух операндов командой обычно изменяется только один из них. Так как информация берется только из одной ячейки, эту ячейку называются источником; ячейка, содержимое которой изменяется, называется приемником.

Ниже приведен формат двухадресной (двухоперандной) команды процессоров СМ.

а	15 11	10 6	5 0	б	15 11	10 0
	КОП	Источник	Приемник		КОП	Приемник

Формат команд процессоров СМ:
а) двухадресная команда;
б) одноадресная команда.

Примеры кодирования двухадресных команд в процессорах СМ

КОП	Мнемоника команды	Комментарий
0001 0010 0110 1110	MOV CMP ADD SUB	Передачаданных Сравнение Сложение Вычитание
0000 1000	- -	Кодирование группы одноадресных команд

Четырехбитный КОП (биты 15-12) кодирует ряд двухоперандных операций, приведенных в таблице 1. Биты (11-6) и (5-0) для команд данного типа определяют адреса источника и приемника данных. Как видно из таблицы, комбинации 0000 и 1000 поля КОП определяют группы одноадресных команд (рис 1,б). КОП 1 (биты 15-12), соответствующий кодам 0000 и 1000, определяет группу одноадресных команд, а КОП 2 (биты 11-6) кодирует конкретную операцию команд данной группы. Таким образом, команды, использующие один операнд, кодируются 10-битным КОП (биты 15-6).

Наиболее гибкая команда требует до четырех операндов. Например, команда сложения может указывать адреса слагаемых, адрес результата и адрес следующей команды. Если для задания адреса требуется 16 бит, то четырехоперандная команда займет 8 байт памяти, не учитывая код операции. Следовательно, получится медленнодействующая ЭВМ с огромной памятью. Поэтому в большинстве микроЭВМ любой команде требуется не более двух операндов. Это достигается следующими приемами:

1. Адрес следующей команды указывается только в командах переходов; в остальных случаях очередная команда выбирается из ячеек памяти, следующих за выполненной командой.
2. Использование ячейки, в которой находится один из операндов, для запоминания результата (например, сумма запоминается в ячейки первого операнда).

Локализацию и обращение к операндам обеспечивают режимы адресации. При введении нескольких режимов адресации необходимо отвести в команде биты, указывающие режимы адресации для каждого операнда. Если предусмотрено восемь режимов адресации, то для задания каждого из них нужно три бита.

Почти во всех форматах команд первые биты отводятся для кода операции, но далее форматы команд разных ЭВМ сильно отличаются друг от друга. Остальные биты должны определять операнды или их адреса, и поэтому они используются для комбинации режимов, адресов регистров, адресов памяти, относительных адресов и непосредственных операндов. Обычно длина команды варьируется от 1 до 3 и даже 6 байт.

По форматам команд можно судить о возможностях ЭВМ.

2.6 Режимы адресации

Для взаимодействия с различными модулями в ЭВМ должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров МП и регистров устройств ввода/вывода. Поэтому каждой из запоминающих ячеек присваивается адрес, т.е. однозначная комбинация бит. Количество бит определяет число идентифицируемых ячеек. Обычно ЭВМ имеет различные адресные пространства памяти и регистров МП, а иногда - отдельные адресные пространства регистров устройств ввода/вывода и внутренней памяти. Кроме того, память хранит как данные, так и команды. Поэтому для ЭВМ разработано множество способов обращения к памяти, называемых режимами адресации.

Режим адресации памяти - это процедура или схема преобразования адресной информации об операнде в его исполнительный адрес.

Все способы адресации памяти можно разделить на:

1) прямой, когда исполнительный адрес берется непосредственно из команды или вычисляется с использованием значения, указанного в команде, и содержимого какого-либо регистра (прямая адресация, регистровая, базовая, индексная и т.д.);

2) косвенный, который предполагает, что в команде содержится значение косвенного адреса, т.е. адреса ячейки памяти, в которой находится окончательный исполнительный адрес (косвенная адресация).

В каждой микроЭВМ реализованы только некоторые режимы адресации, использование которых, как правило, определяется архитектурой МП.

2.7 Типы архитектур

Существует несколько подходов к классификации микропроцессоров по типу архитектуры. Так, выделяют МП с CISC (Complete Instruction Set Computer) архитектурой, характеризуемой полным набором команд, и RISC (Reduce Instruction Set Computer) 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:

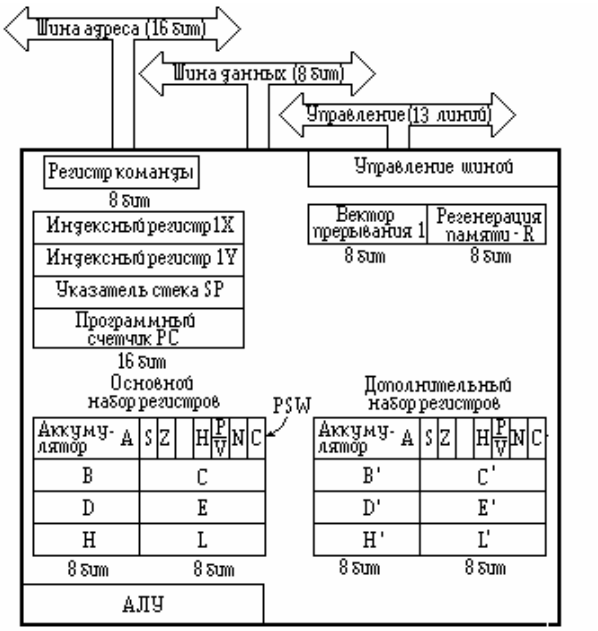
- с фиксированной разрядностью и списком команд (однокристальные);
- с наращиваемой разрядностью (секционные) и микропрограммным управлением.

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.

1. 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. Процессор Z80 - детище фирмы Zilog помимо расширенной системы команд, одного номинала питания и способности исполнять программы, написанные для i8080, имел архитектурные "изюминки".



Рисю 2.5. Микропроцессор Z80 фирмы Zilog.

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 Sinclair ZX, Sinclair Spectrum, Tandy TRS80.

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

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.

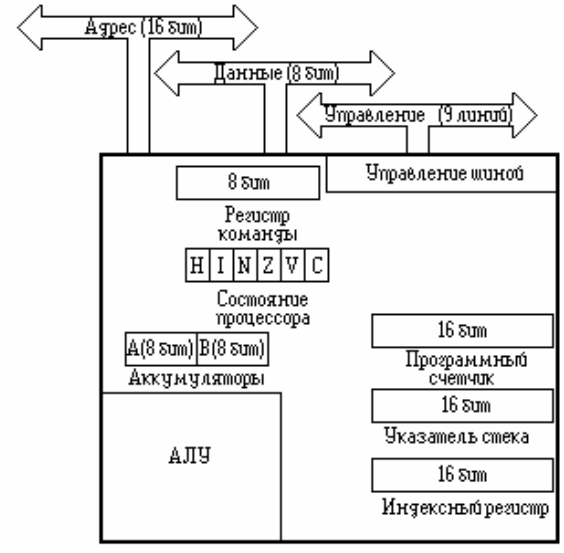


Рис 2.6. Микропроцессор MC6800 фирмы Motorola.

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.

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

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.

3. 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.

3.1 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. 3.1). 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.

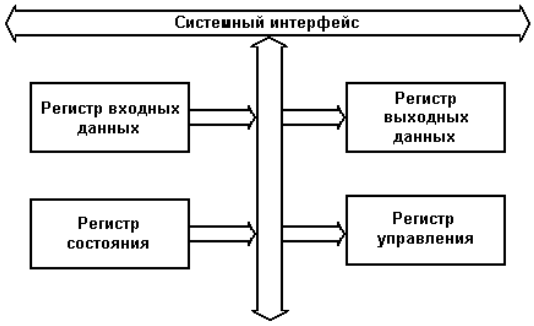


Рис. 3.1. Программная модель внешнего устройства

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. 3.2. 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.

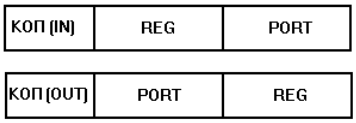


Рис. 3.2. Команды ввода/вывода (общий формат)

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.

3.2 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. 3.3,а). Бит четности (или нечетности) может отсутствовать. Когда передатчик бездействует (данные не посылаются на линию), на линии сохраняется уровень сигнала, соответствующий логической 1.

Рис. 3.3. Формат асинхронной последовательной передачи данных

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. 3.3,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. 8, 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. 3.3 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. 3.4). 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 (см. рис. 3.4).

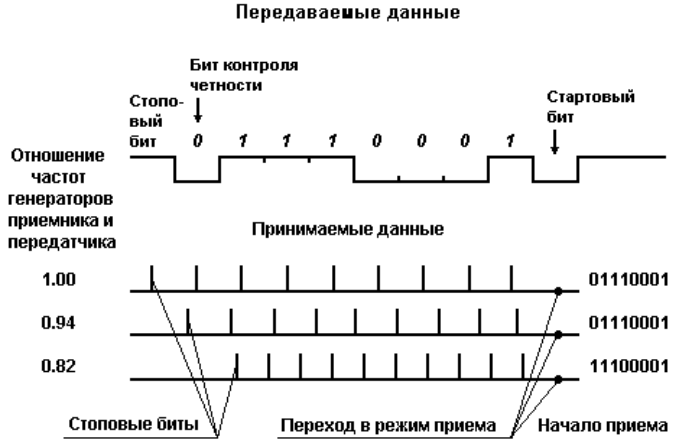


Рис. 3.4. Ошибка из-за рассинхронизации генераторов передатчика и приемника

3.3 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. 3.5), в шине связи с ВУ используются всего два управляющих сигнала: "Выходные данные готовы" и "Данные приняты".

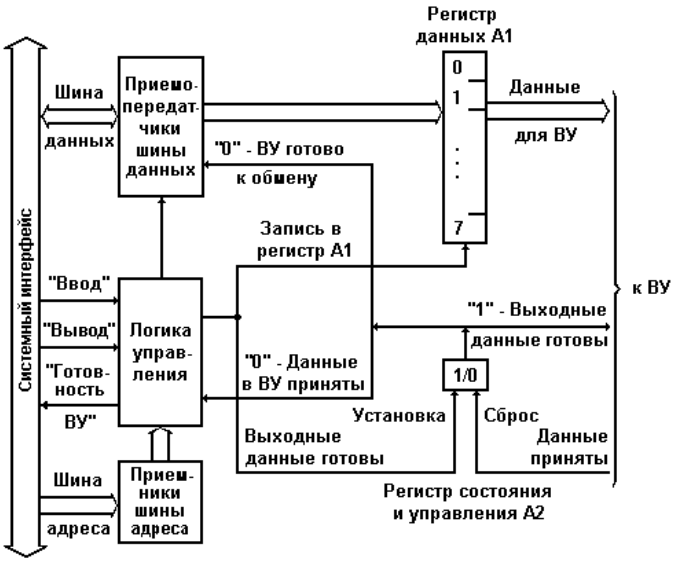


Рис. 3.5. Простой параллельный контроллер вывода.

Для формирования управляющего сигнала "Выходные данные готовы" и приема из ВУ управляющего сигнала "Данные приняты" в контроллере используется одноразрядный адресуемый регистр состояния и управления A2 (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 A1) в регистр состояния и управления 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 A2), то данные передаются с шины данных системного интерфейса в регистр данных A1 контроллера и далее в ВУ. Иначе повторяется п. 1.

Пример 2.1. Фрагмент программы передачи байта данных в асинхронном режиме с использованием параллельного контроллера ВУ (рис. 3.5). Для написания программы асинхронной передачи воспользуемся командами процессора 8086.

MOV	DX, A2	номер порта A2 помещаем в DX
m1:IN	AL, DX	чтение байта из порта A2
TEST	AL, 1	проверка нулевого состояния регистра A2
JNS	ml	переход на метку ml если разряд не нулевой
MOV	AL, 64	выводимый байт данных помещается в AL
MOV	DX, A1	номер порта A1 записываем в DX
OUT	DX, AL	содержимое регистра AX передаем в порт A1

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 А2, 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 АХ процессора в регистр данных контроллера А1. Процессор по шине адреса передает в контроллер адрес А1, а по шине данных - байт данных, сопровождая их сигналом "Вывод" (IOWR#). Логика управления контроллера обеспечивает запись данных с шины данных в регистр данных А1 и устанавливает в ноль бит готовности регистра состояния А2, формируя тем самым управляющий сигнал для ВУ "Выходные данные готовы". ВУ принимает байт данных и управляющим сигналом "Данные приняты" устанавливает в единицу регистр состояния А2. (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. 3.6. В этом контроллере при взаимодействии с внешним устройством также используются два управляющих сигнала: "Данные от ВУ готовы" и "Данные приняты".

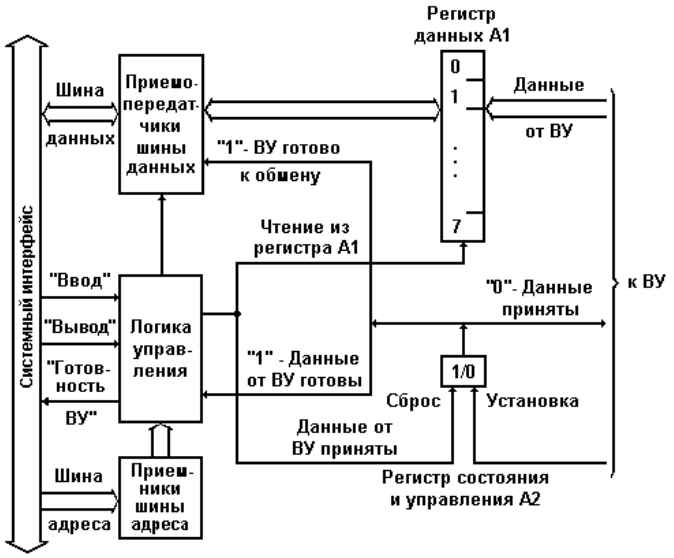


Рис. 3.6. Простой параллельный контроллер ввода

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 (см. рис. 3.5),

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 А1.
2. Если данные готовы (логическая 1 в регистре А2), то они передаются из регистра данных А1 на шину

данных системного интерфейса и далее в регистр процессора или ячейку памяти микрокомпьютера. Иначе повторяется п. 1.

Пример 2.2. Фрагмент программы приема байта данных в асинхронном режиме с использованием параллельного интерфейса (контроллер ВУ, рис. 3.6):

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 2.1. 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 считывает 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.

3.4 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.

3.4.1 Синхронный последовательный интерфейс

Простой контроллер для синхронной передачи данных в ВУ по последовательной линии связи (последовательный интерфейс) представлен на рис. 3.7.

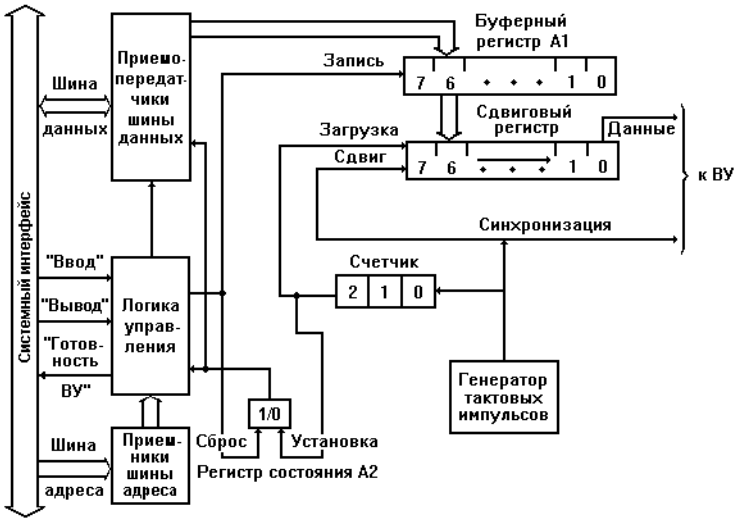


Рис. 3.7. Контроллер последовательной синхронной передачи

Восьмиразрядный адресуемый буферный регистр контроллера А1 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 и рис. 3.5), только при наличии единицы в одноразрядном адресуемом регистре состояния контроллера А2. Единица в регистре состояния указывает на готовность контроллера принять очередной байт в буферный регистр. 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 2.1 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 отдельной 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.

3.4.2 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. 3.8. Он предназначен для передачи данных в формате с двумя стоповыми битами.

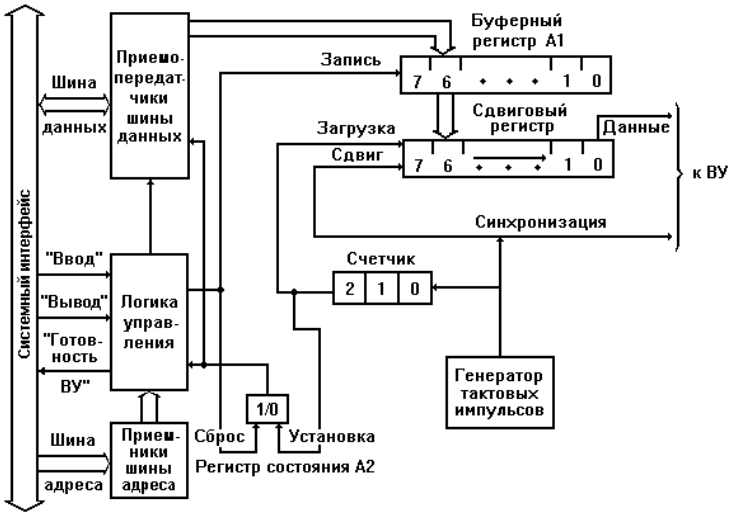


Рис. 3.8. Контроллер последовательной асинхронной передачи

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).

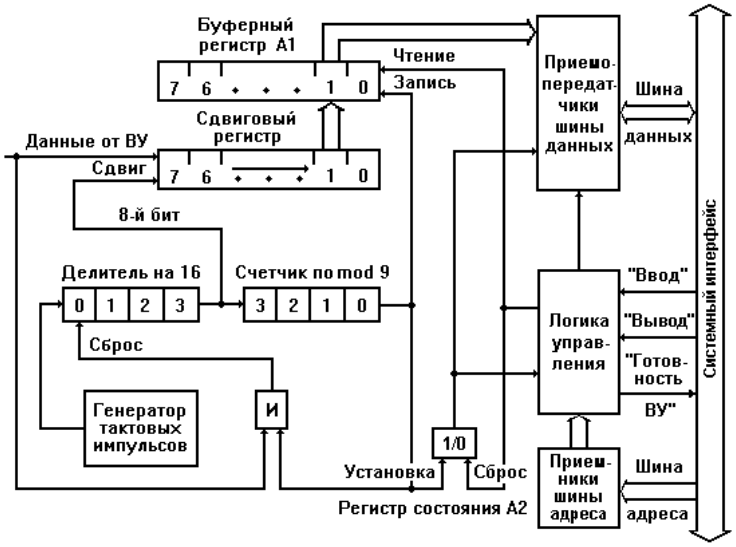


Рис. 3.9. Контроллер последовательного асинхронного приема

Уровень логической единицы поступает по линии "Данные" в контроллер для асинхронного приема данных (рис. 3.9). 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 выдаст сигнал, поступающий на входы сдвигового регистра и счетчика импульсов сдвига. 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 2.2 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. 14 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.

3.5 Способы обмена информацией в микропроцессорной системе

В ЭВМ применяются три режима ввода/вывода: программно-управляемый ВВ (называемый также программным или нефорсированным 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.

1. 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.
2. 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.

3.5.1 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. 3.10). Данный цикл называется циклом ожидания готовности ВУ и реализуется в различных процессорах по-разному.

Рис. 3.10. Цикл программного ожидания готовности внешнего устройства

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.

3.5.2 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 2.1 и 2.2, 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.

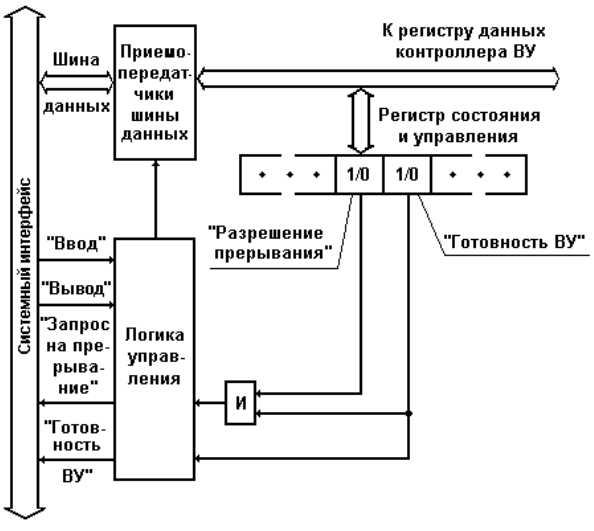


Рис. 3.11. Фрагмент блок-схемы контроллера ВУ с разрядом "Разрешение прерывания" в регистре состояния и управления

Для решения этой проблемы регистр состояния и управления контроллера ВУ (рис. 3.11) 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.

Аналогичным путем решается проблемам 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. 3.12.

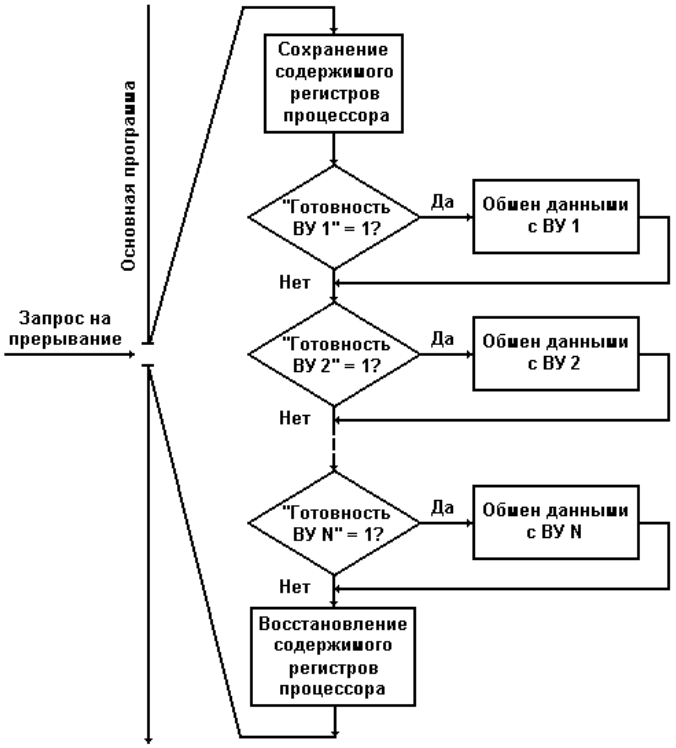


Рис. 3.12. Структура единой программы обработки прерываний и ее связь с основной программой

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 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 выдается контроллером не одновременно с запросом на прерывание, а только по разрешению процессора, как это реализовано в схеме на рис. 3.13. 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.

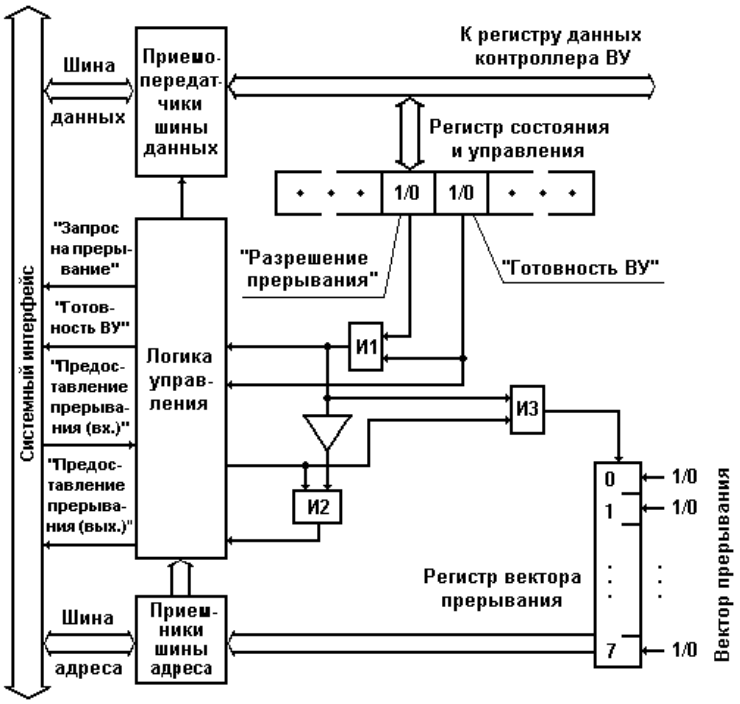


Рис. 3.13. Формирование векторов прерывания в контроллере ВУ

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. 3.14). 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 адресноинформационную шину.

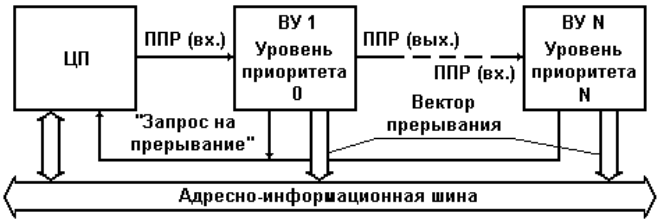


Рис. 3.14. Реализация приоритетов ВУ в микроЭВМ с векторной системой прерываний с интерфейсным вектором (ППР (вх.) - "Предоставление прерывания (входной)"; "ППР (вых.) - Предоставление прерывания (выходной)")

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 рисунке 3.15.

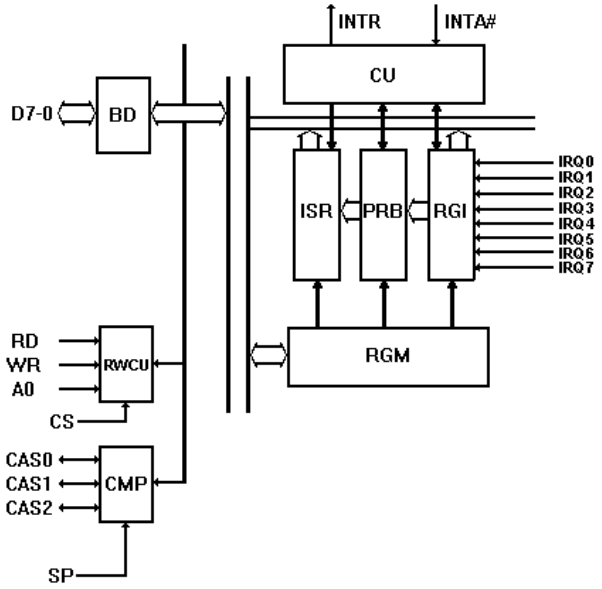


Рис. 3.15. Контроллер прерываний Intel 8259A

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.

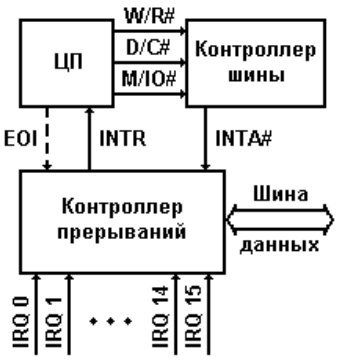


Рис. 3.16. Упрощенная схема взаимодействия контроллера прерываний с процессором и контроллером шины в IBM-совместимых персональных компьютерах класса AT

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 (End of Interruption), 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.

3.5.3 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 в специальном контроллере, который называется контроллером прямого доступа к памяти.

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 системному 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. 3.17.

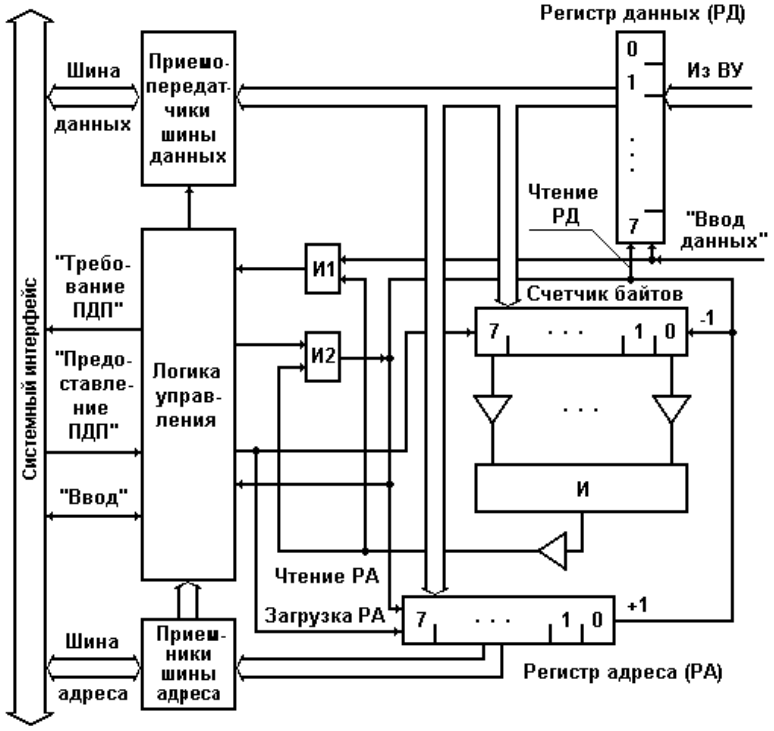


Рис. 3.17. Контроллер ПДП для ввода данных из ВУ в режиме "Захват цикла" и отключением процессора от шин системного интерфейса

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 доступ 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 случаях, 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" (рис. 18 Организация прерываний в микроЭВМ) используются сигналы "Требование прямого доступа" и "Предоставление прямого доступа", соответственно.

4. Память микропроцессорной системы

4.1 Микросхемы памяти в составе микропроцессорной системы

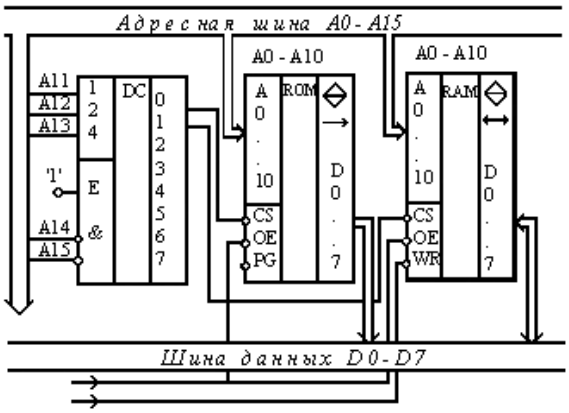


Рис. 4.1. Микросхемы ОЗУ (К573РУ9) и ППЗУ (К573РФ5) в составе микропроцессорной системы.

На рис. 4.1 представлено взаимодействие К573РФ2(5) и К573РУ9, имеющих одинаковую организацию 2Кх8, с системной магистралью. Байт данных с шины данных (линии D0-D7) считывается (или записывается) по адресу, выставленному на шине адреса (линии A0-A10). Естественно, число адресуемых ячеек составляет 2^{11}=800h=2048. Микросхема-дешифратор К555ИД7 посредством сигнала CS# (выбор кристалла) позволяет выбрать положение ИМС ЗУ в адресном пространстве. Для данного случая это адреса 0000h-07FFh для ПЗУ(ROM) и 0800h-0FFFh для ОЗУ(RAM). Низкий уровень сигналов управления MEMW# и MEMR# активизирует процесс записи и чтения, соответственно. Напомним, что запись информации в данную ИМС ППЗУ возможен только вне микропроцессорной системы в специальном программаторе после УФ стирания путем подачи достаточно высокого напряжения на вход PG.

Аналогично можно проследить и взаимодействие программируемых ИМС параллельного интерфейса и программируемого таймера, служащих для взаимодействия МП с внешними устройствами. На рис. 4.12 они обозначены как PI и T. Регистры этих микросхем также доступны пользователю для чтения/записи, как и ячейки основной памяти. Однако их активизируют другие сигналы управления IOR# и IOW# (запись в порты ввода/вывода).

Микросхема КР580ВВ55А (аналог Intel 8255A) позволяет переключать шину данных компьютера на один из трех 8-битных портов (регистров) A, B или C. Направление передачи данных и режим работы (0-2) определяются программным способом. Чаще других используется режим 0: простой ввод-вывод. Состояние адресных линий A0 и A1 позволяет выбрать для обмена информацией регистры A, B, C или регистр управления. Режим работы параллельного интерфейса КР580ВВ55А определяется байтом, записанным в регистр управления. При работе на вывод порты A, B, C действуют как регистры, т.е. сохраняют информацию до следующей записи; при работе на ввод информация теряется. Порт C, в отличие от портов A и B, разбит на полубайты и может программироваться раздельно, т.е. мы имеем группы A и B.

В дополнение к основным режимам работы микросхема КР580ВВ55А обеспечивает возможность программной независимой установки/сброса любого бита (порт C), в этом случае старший бит в регистре управления должен быть 0.

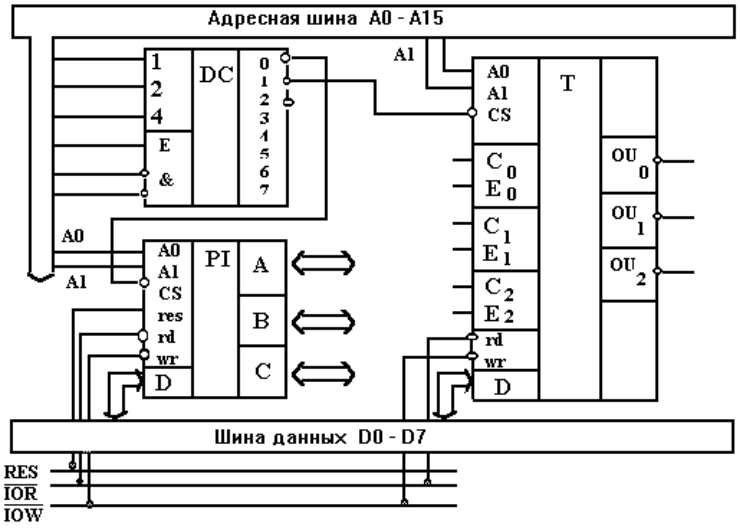


Рис. 4.2. Пример взаимодействия программируемых ИМС параллельного порта (PI) КР580ВВ55А и таймера (Т) КР580ВИ53 с системной магистралью компьютера; DC - дешифратор К555ИД7.

Интегральная схема КР580ВИ53 создана по n-МОП технологии, Uпит=+5 В, U1>2.4 В, U0<0.45 В, P=1 Вт, fтакт <2 МГц, 24 вывода, максимальное значение счета: 216 - в двоичном коде и 104 - в двоично-десятичном. Программируемый таймер КР580ВИ53 реализован в виде трех независимых 16-разрядных вычитающих счетчиков (каналов) с общей схемой управления. Каждый канал может работать в шести режимах. Программирование режимов работы каналов осуществляется индивидуально и в произвольном порядке путем ввода управляющих слов в регистры режимов каналов, а в счетчики - некоторого числа байт. На рис. 4.2 можно видеть, что таймер обменивается информацией с 8-битной шиной данных микропроцессорной системы через вход D, а также он связан с адресной шиной двумя линиями A0-A1, обеспечивающими выбор одного из четырех регистров (3 канала и управляющее слово). Сигналы шины управления IOR# и IOW# (чтение/запись из/во внешнее устройство) определяют направление потока информации от процессора к таймеру и наоборот. C0, C1, C2 - тактовые входы, сигналы на E0, E1, E2 - разрешают или запрещают работу соответствующего канала, OU0, OU1, OU2 - выходы.

4.2 Буферная память

В вычислительных системах используются подсистемы с различным быстродействием, и, в частности, с различной скоростью передачи данных (рис. 4.3). Обычно обмен данными между такими подсистемами реализуется с использованием прерываний или канала прямого доступа к памяти. В первую очередь подсистема 1 формирует запрос на обслуживание по мере готовности данных к обмену. Однако обслуживание прерываний связано с непроизводительными потерями времени и при пакетном обмене производительность подсистемы 2 заметно уменьшается. При обмене данными с использованием канала прямого доступа к памяти подсистема 1 передает данные в память подсистемы 2. Данный способ обмена достаточно эффективен с точки зрения быстродействия, но для его реализации необходим довольно сложный контроллер прямого доступа к памяти.

Рис. 4.3. Применение буферной памяти.

Наиболее эффективно обмен данными между подсистемами с различным быстродействием реализуется при наличии между ними специальной буферной памяти. Данные от подсистемы 1 временно запоминаются в буферной памяти до готовности подсистемы 2 принять их. Емкость буферной памяти должна быть достаточной для хранения тех блоков данных, которые подсистема 1 формирует между считываниями их подсистемой 2. Отличительной особенностью буферной памяти является запись данных с быстродействием и под управлением подсистемы 1, а считывание - с быстродействием и под управлением подсистемы 2 ("эластичная память"). В общем случае память должна выполнять операции записи и считывания совершенно независимо и даже одновременно, что устраняет необходимость синхронизации

подсистем. Буферная память должна сохранять порядок поступления данных от подсистемы 1, т.е. работать по принципу "первое записанное слово считывается первым" (First Input First Output - FIFO). Таким образом, под буферной памятью типа FIFO понимается ЗУПВ, которое автоматически следит за порядком поступления данных и выдает их в том же порядке, допуская выполнение независимых и одновременных операций записи и считывания. На рис. 4.4 приведена структурная схема буферной памяти типа FIFO емкостью 64x4.

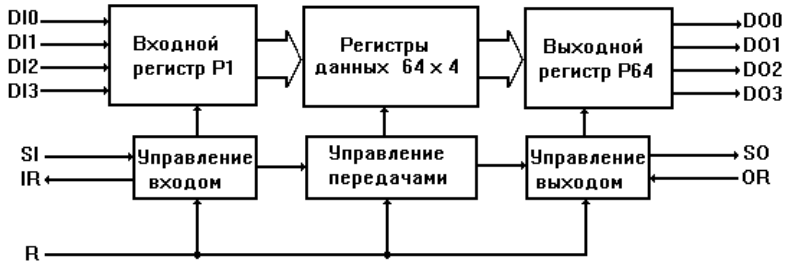


Рис. 4.4. Структурная схема буфера 64x4.

На кристалле размещены 64 4-битных регистра с независимыми цепями сдвига, организованных в 4 последовательных 64-битных регистра данных, 64-битный управляющий регистр, а также схема управления. Входные данные поступают на линии DI0-DI3, а вывод данных осуществляется через контакты DO0-DO3. Ввод (запись) данных производится управляющим сигналом SI (shift in), а вывод (считывание) - сигналом вывода SO (shift out). Ввод данных осуществляется только при наличии сигнала готовности ввода IR (input ready), а вывод - при наличии сигнала готовности вывода OR (output ready). Управляющий сигнал R (reset) производит сброс содержимого буфера.

При вводе 4-битного слова под действием сигнала SI оно автоматически передвигается в ближайший к выходу свободный регистр. Состояние регистра данных отображается в соответствующем ему управляющем триггере, совокупность триггеров образует 64-битный управляющий регистр. Если регистр содержит данные, то управляющий триггер находится в состоянии 1, а если регистр не содержит данных, то триггер находится в состоянии 0. Как только управляющий бит соседнего справа регистра изменяется на 0, слово данных автоматически сдвигается к выходу. Перед началом работы в буфер подается сигнал сброса R и все управляющие триггеры переводятся в состояние 0 (все регистры буфера свободны). На выводе IR формируется логическая 1, т.е. буфер готов воспринимать входные данные. При действии сигнала ввода SI входное слово загружается в регистр P1, а управляющий триггер этого регистра устанавливается в состояние 1: на входе IR формируется логический 0. Связи между регистрами организованы таким образом, что поступившее в P1 слово "спонтанно" копируется во всех регистрах данных FIFO и появляется на выходных линиях DO0-DO3. Теперь все 64 регистра буфера содержат одинаковые слова, управляющий триггер последнего регистра P64 находится в состоянии 1, а остальные управляющие триггеры сброшены при передаче данных в соседние справа регистры. Состояние управляющего триггера P64 выведено на линию готовности выхода OR; OR принимает значение 1, когда в триггер записывается 1. Процесс ввода может продолжаться до полного заполнения буфера; в этом случае все управляющие триггеры находятся в состоянии 1 и на линии IR сохраняется логический 0.

При подаче сигнала SO производится восприятие слова с линий DO0-DO3, управляющий триггер P64 переводится в состояние 1, на линии OR появляется логическая 1, а управляющий триггер P64 сбрасывается в 0. Затем этот процесс повторяется для остальных регистров и нуль в управляющем регистре перемещается ко входу по мере сдвига данных вправо.

В некоторых кристаллах буфера FIFO имеется дополнительная выходная линия флажка заполнения наполовину. На ней формируется сигнал 1, если число слов составляет более половины емкости буфера.

Рассмотренный принцип организации FIFO допускает выполнение записи и считывания данных независимо и одновременно. Скорость ввода определяется временным интервалом, необходимым для передачи данных из P1, а выводить данные можно с такой же скоростью. Единственным ограничением является время распространения данных через FIFO, равное времени передачи входного слова на выход незаполненного буфера FIFO. Оно равняется произведению времени внутреннего сдвига и числа регистра данных. В буферах FIFO, выполненных по МОП-технологии и имеющих емкость 64 слова, время распространения составляет примерно 30 мкс, а в биполярных FIFO такой же емкости - примерно 2 мкс.

Буферы можно наращивать как по числу слов, так и по их длине.

4.3 Стековая память

Стековой называют память, доступ к которой организован по принципу: "последним записан - первым считан" (Last Input First Output - LIFO). Использование принципа доступа к памяти на основе механизма LIFO началось с больших ЭВМ. Применение стековой памяти оказалось очень эффективным при

построении компилирующих и интерпретирующих программ, при вычислении арифметических выражений с использованием польской инверсной записи. В малых ЭВМ она стала широко использоваться в связи с удобствами реализации процедур вызова подпрограмм и при обработке прерываний.

Принцип работы стековой памяти состоит в следующем (см. рис. 4.15). Когда слово А помещается в стек, оно располагается в первой свободной ячейке памяти. Следующее записываемое слово перемещает предыдущее на одну ячейку вверх и занимает его место и т.д. Запись 8-го кода, после Н, приводит к переполнению стека и потере кода А. Считывание слов из стека осуществляется в обратном порядке, начиная с кода Н, который был записан последним. Заметим, что выборка, например, кода Е невозможна до выборки кода F, что определяется механизмом обращения при записи и чтении типа LIFO. Для фиксации переполнения стека желательно формировать признак переполнения.

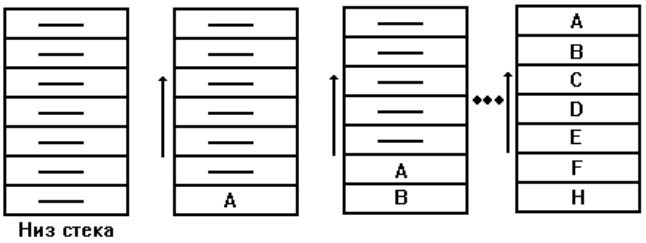


Рис. 4.15. Принцип работы стековой памяти.

Перемещение данных при записи и считывании информации в стековой памяти подобно тому, как это имеет место в сдвигающих регистрах. С точки зрения реализации механизма доступа к стековой памяти выделяют аппаратный и аппаратно-программный (внешний) стеки.

Аппаратный стек представляет собой совокупность регистров, связи между которыми организованы таким образом, что при записи и считывании данных содержимое стека автоматически сдвигается. Обычно емкость аппаратного стека ограничена диапазоном от нескольких регистров до нескольких десятков регистров, поэтому в большинстве МП такой стек используется для хранения содержимого программного счетчика и его называют стеком команд. Основное достоинство аппаратного стека - высокое быстродействие, а недостаток - ограниченная емкость.

Наиболее распространенным в настоящее время и, возможно, лучшим вариантом организации стека в ЭВМ является использование области памяти. Для адресации стека используется указатель стека, который предварительно загружается в регистр и определяет адрес последней занятой ячейки. Помимо команд CALL и RET, по которым записывается в стек и восстанавливается содержимое программного счетчика, имеются команды PUSH и POP, которые используются для временного запоминания в стеке содержимого регистров и их восстановления, соответственно. В некоторых МП содержимое основных регистров запоминается в стеке автоматически при прерывании программ. Содержимое регистра указателя стека при записи уменьшается, а при считывании увеличивается на 1 при выполнении команд PUSH и POP, соответственно.

5. Микропроцессор Intel 8086(88)

5.1. Поставляемая разработчиком информация

Какими бы ни были рассматриваемые микропроцессоры, касающаяся их информация содержит много общего. Типовая документация содержит информацию о структуре ИС, схеме выводов ИС и назначении каждого из них. Схематизируется архитектура МП, описываются его основные свойства. Одновременно даются временные диаграммы и состав команд МП. Документация содержит также схемы различных систем, использующих рассматриваемый микропроцессор.

Обычно микропроцессор помещается в корпус интегральной схемы с 40 двусторонними выводами (корпус с двухрядной упаковкой выводов DIP — *dual-in-line package).* На рис. 5.1 приведены два типа микропроцессоров - в пластмассовом корпусе (рис. 5.1, а) и в керамическом (рис. 5.1,6) с 40 выводами. Микропроцессор в керами ческом DIP-корпусе используется при высоких температурах. Микропроцес-соры могут поставляться также с 28, -42, 50 и 64 выводами.

На рис. 5.1, *в* и *г* приведены два способа определения положения вывода *1.* Заметим вырез и желобок по всей длине, являющиеся отметками на пластмассовом корпусе (рис. 5.1, *в).* Непосредственно после этой отметки в[1] направлении, обратном ходу часовой стрелки, находится вывод / ИС. На рис. 5.1, г отметкой, позволяющей определить вывод *1* ИС, является маленькая точка слева. Затем выводы нумеруются в направлении, обратном ходу часовой стрелки при виде на ИС сверху.

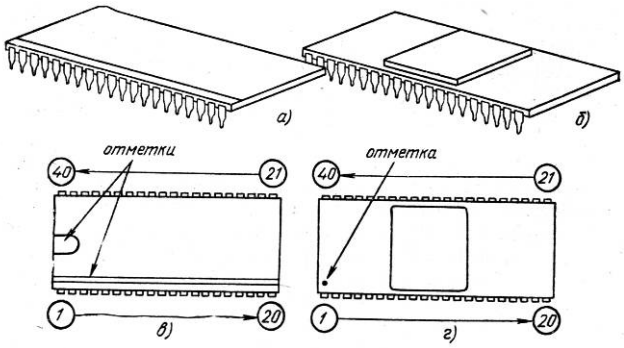


Рис. 5.1. Микропроцессоры в корпусах:

Схема выводов (рис. 5.2) приводится в документации. Разработчики представляют все сведения о названиях и назначениях каждого из выводов микропроцессора. Схема на рис. 5.2 соответствует микропроцессору Intel 8080. Отметим, что выводы *2, 11, 20, 28* являются выводами питания.

Выводы *15, 22 (Ф1, Ф2)* являются входами внешнего двухфазного генератора тактовых импульсов — часов. Выводы *3—10* (Intel 8080) двунаправленные (это значит, что они являются то входами, то выходами). Эти выводы данных *(D_0—D_7)* являются восемью подсоединениями на шину данных системы. Адресная 16-разрядная шина системы будет связана выходами $Л_0$—$Л_{15}$. Шесть других выходов *(SYNC, DBIN, WAIT, WR, HID A, INTE)* несут сигналы управления и синхронизации всем прочим элементам системы. Наконец, четыре входа *(READY, HOLD, INT, RESET)* являются входами управления, которые воспринимают информацию, поступающую из системы. На рис. 3.2 приведена вся информация по каждому выводу микропроцессора Intel 8080.

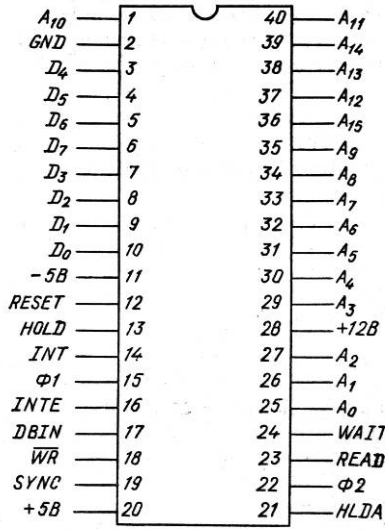


Рис. 5.2. Схема выводов МП Intel 8080

Выводы	Назначение	Вход или выход
GND, +5 В, −5 В, +12 В	Питание	Входы
Ф1, Ф2	Тактовые импульсы	Входы
D_0—D_7	Линии данных	Двунаправленные
A_0—A_{15}	Линии адресов	Выходы
SYNC	Синхронизация	Выход
DBIN	Строб входных данных	Выход
WAIT	Ожидание	Выход
$\overline{WR}$	Строб записи	Выход
HLDA	Подтверждение захвата	Выход
INTE	Разрешение прерывания	Выход
READY	Готовность ввода данных	Вход
HOLD	Захват	Вход
INT	Требование прерывания	Вход
RESET	Сброс	Вход

Типовая документация содержит также структурную схему микропроцессора. На рис. 5.3, *а* представлена функциональная схема МП Intel 8080, которая содержит внутренние регистры — аккумулятор, пары регистров *B* и *C, D* и *E, H* и L,*указатель стека SP(Stack Pointer* (англ.) — указатель стека*), регистр состояния (индикатор),* несколько *регистров временного хранения данных.* Эта схема содержит также *регистр команд, дешифратор команд,* а также *устройство управления и синхронизации.* Наконец, она содержит также АЛУ, его *объединенный индикатор* и блок *десятичного корректора.* Все восемь линий данных, так же как и 16-разрядные адресные выходы, снабжены буферами. Микропроцессор Intel 8080 содержит также несколько внутренних линий управления, цепей данных и шины.

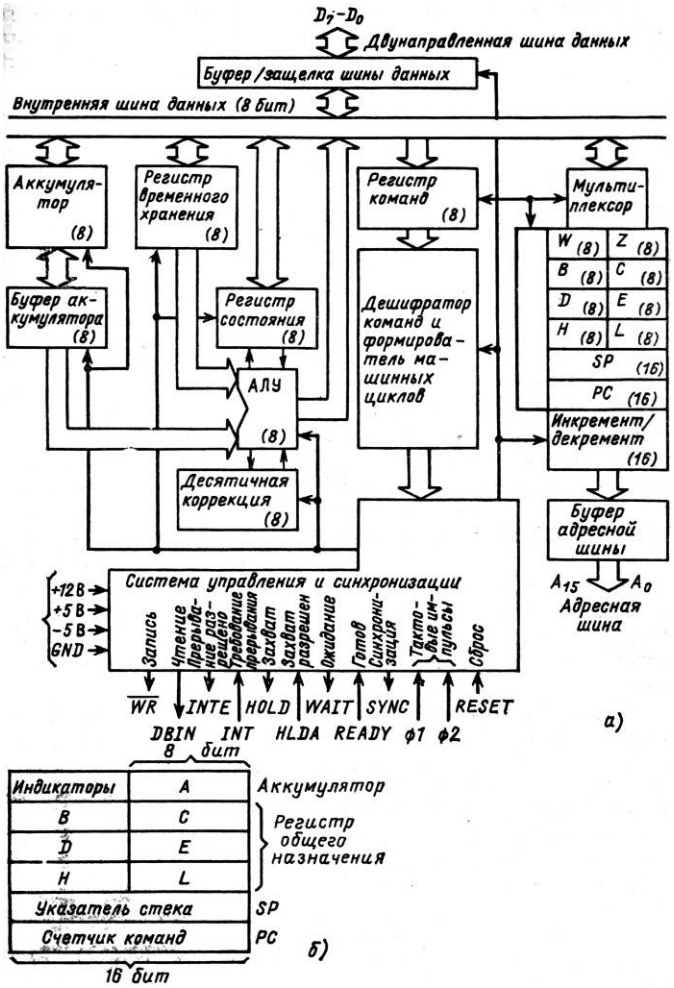


Рис. 5.3 Функциональная схема МП Intel 8080 и программно-доступные регистры

На рис. 5.3, *б* представлены используемые программистом регистры МП Intel 8080. Отметим, что основным является регистр *A* или аккумулятор. Регистры *B* и *C, D* и *E, H* и *L* являются универсальными. Указатель стека, счетчик команд и индикатор состояния являются специальными регистрами. Пара регистров *HL* может быть использована также в качестве адресного регистра.

Документация содержит разработанные временные диаграммы, которые показывают соотношения между входами тактовых импульсов и другими внешними сигналами (синхронизации, записи, адресных выходов, ВВ данных и т. д.) и внутренними операциями. Разработчик дает также указания о способе, по которому микропроцессор используется в случае минимальной системы Такая система, основанная на МП Intel 8080, могла бы содержать микропроцессор, генератор тактовых импульсов, устройство управления системой, ПЗУ, ОЗУ и интерфейс портов ВВ. Документация содержит подробную информацию о системе команд.

5.2. Схема и назначение выводов

Микропроцессор заключен в DIP-корпус с 40 двухрядными выводами. Эта ИС питается напряжением +5 В по выводам 1 и *2*.Выводы X_1 и X_2 вверху справа предназначены для подсоединения кристалла управления частотой ГТИ МП. Для наиболее распространенных устройств характерно наличие ГТИ на кристалле МП, тогда как для более старых устройств был необходим внешний ГТИ. Выход *CLK* (вывод *38)* предназначен для выдачи сигналов ГТИ в систему. Частота сигнала на выводе *38 (CLK),* очевидно, подчинена частоте внутреннего ГТИ.

Адресная шина системы будет подсоединена к выводам ИС A_o—A_{x_b} (рис. 3.4). Эти 16 адресных линий (может быть и другое количество) могут обеспечить доступ к 65 536 (2^{16}) ячейкам памяти или/и портам ВВ.

Поток данных и команд от микропроцессора и в него обеспечивается выводами D_o—D_7 на ИС рис. 5.4. Эти выводы *(21—28)* двунаправленные, т. е. являются то выходами, то входами. Кроме того, обычно они могут переводиться в третье состояние (высокого сопротивления).

Вывод *30* является выходом управления записью. Сигнал L-уровня на выходе *WR* указывает, что данные, имеющиеся на шине данных, должны быть записаны в область памяти или выбранное УВВ. Выход управления считыванием *RD* (вывод *31)* активизируется L-сигналом, который указывает, что избранные места в памяти или УВВ должны быть считаны.

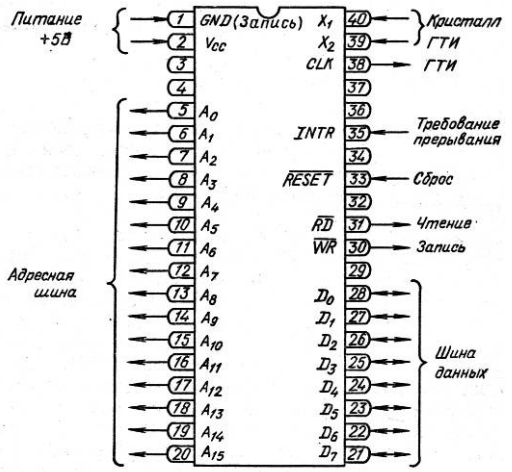


Рис. 5.4. Схема выводов типового микропроцессора

Результатом активизации входа сброса является остановка работы МП по текущей программе и переход к подпрограмме сброса. Сигнал L-уровня на входе *RESET* МП сбрасывает счетчик команд до заранее предопределенного адреса, например 0000H. Другие внутренние регистры МП могут быть также сброшены или их содержимое изменяется в течение операции сброса. Когда вход *RESET* переходит в состояние HIGH, МП начинает выполнение команд с нового адреса памяти, т. е. с адреса 0000H в данном случае (или с другого заранее предопределенного адреса памяти).

Этот адрес соответствует началу подпрограммы новой инициализации системы, содержащейся обычно в ПЗУ.

Большинство микропроцессоров находятся в фазе с ГТИ, следовательно, они являются синхронными. Вход *RESET* МП асинхронный и может вмешаться и приостановить наполовину выполненную команду.

Вход требования прерывания помещен на вывод *35*. Вход *WTR* отвечает на Н-сигнал внешнего устройства. Предположим, что устройство интерфейса ввода на рис. 3.5 загружено 8 бит параллельных данных, готовых для передачи в МП: процесс может быть продолжен в порядке, показанном на рис. 5.5.

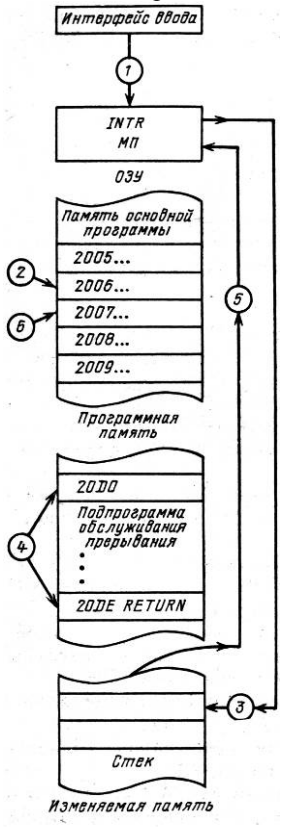


Рис. 5.5. Этапы отработки требования прерывания в типовом МП

1. Интерфейс ввода выдает сигнал требования прерывания в направлении МП.
2. Микропроцессор завершает выполнение текущей команды, находящейся в памяти по адресу 2006H.

3. Поскольку управление должно обеспечить последующее обращение к команде по адресу 2007H, содержимое счетчика команд (именно 2007H) и содержимое большинства регистров МП помещается в специальную зону ОЗУ, называемую *стеком*. Это содержимое будет позже извлечено в определенном порядке в регистры МП и в счетчик команд.

4. МП разветвляется в предопределенный адрес памяти и начинает выполнение подпрограммы обслуживания прерывания (в нашем примере 20DOH). Микропроцессор выполняет тогда команды подпрограммы, которые всегда в нашем примере обеспечивают выполнение операций ввода. По адресу 20DEH МП находит конец этой подпрограммы обслуживания и получает приказ вернуться в основную программу.

5. Перед возвращением в основную программу данные регистров и счетчик команд, помещенные в стек, возвращаются в МП.

6. Теперь счетчик команд отсылает МП в память по адресу 2007H, т. е. в основную программу, и нормальное выполнение ее продолжается.

Прерывание является очень полезным способом, позволяющим периферии вмешаться и заставить МП выполнять требуемую операцию почти сразу. Многие микропроцессоры обладают одним или несколькими прерываниями. Входы прерывания могут быть названы также сбросами, новым запуском, маскируемыми прерываниями или сетками.

5.3. Архитектура микропроцессора Intel 8080

Практически все микропроцессоры содержат по меньшей мере следующие элементы: АЛУ; несколько регистров; счетчик команд; систему декодирования команд; секцию управления и синхронизации; буферы и защелки; внутренние шины цепей управления; несколько входов и выходов управления.

Кроме того, кристалл микропроцессора может также содержать функциональные устройства: ПЗУ; ОЗУ; ряд портов ВВ; внутренние цепи ГТИ — часов; программируемый таймер; систему выбора приоритета прерываний; логику интерфейса последовательно-параллельных взаимодействий при ВВ; логическое управление прямым доступом к памяти.

Микропроцессор обладает восемью двунаправленными связями с шиной данных, по которым они выводятся на внутреннюю шину. Слева от МП показаны 16 выходов на адресную шину с буферами/защелками на внутренней адресной шине. Выходы управления показаны внизу слева; это линии записи, чтения и ГТИ. Внизу справа от МП принимаются два сигнала по линиям сброса и требования прерывания. У нашего МП есть внутренняя цепь ПИ, и ему нужен только один внешний кристалл (или в некоторых случаях — одна емкость) для запуска МП. Наконец, этот микропроцессор запитан от единственного источника напряжением +5 В.

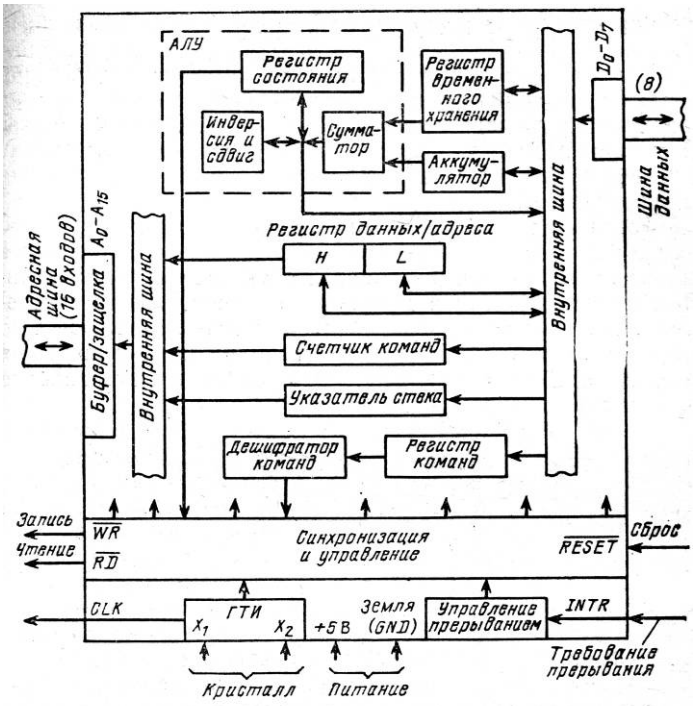


Рис. 5.6. Функциональная схема (архитектура) типового МП

Регистр команд: это устройство является 8-разрядным регистром и содержит первый байт команды (ее КОП).

Дешифратор команд: это устройство интерпретирует (декодирует) содержимое регистра команд, определяет микропрограмму для выполнения нужной из всего множества команд и последовательно вводит в действие секцию управления.

Арифметико-логическое устройство **(АЛУ):** это устройство выполняет операции арифметические, логические и сдвига, в результате которых устанавливает регистр состояния (индикаторы). Результаты помещаются в аккумулятор, связанный с внутренней шиной. Часто внутренние регистры и аккумулятор рассматриваются как часть АЛУ. Условия индикатора передаются в устройство управления и синхронизации.

Аккумулятор: это устройство является универсальным 8-разрядным регистром, где концентрируется большинство результатов выполнения команд — арифметических, логических, загрузки, запоминания результата, ввода/вывода.

Счетчик команд: это устройство является разновидностью 16-разрядной памяти, которое постоянно указывает на следующую выполняемую команду. Оно всегда содержит 16-разрядный адрес. Счетчик может быть инкрементировап или сброшен устройством управления или изменен командой передачи данных.

Устройство управления и синхронизации: это устройство получает сигналы дешифратора команд для определения природы выполняемой команды. Оно получает также информацию от регистра состояния в случае условного перехода. Сигналы управления и синхронизации передаются во все устройства системы для координации выполнения команд, и, наконец, оно вырабатывает сигналы управления внешними устройствами (ОЗУ, ПЗУ, УВВ и т. д.).

Регистр состояния: элементарный микропроцессор на рис. 5.6 содержит в своем регистре состояния индикаторы только нуля и переноса.

Новые дополнительные устройства этого микропроцессора содержат внутренний ГТИ, систему управления прерываниями, указатель стека и универсальный регистр данных/адреса.

Устройство управления прерываниями принимает сигнал прерывания с внешнего устройства через вход *1NTR.* Оно управляет по этому сигналу МП в соответствии с ранее рассмотренными этапами (см. § 3.2). Таким образом, МП ветвится в подпрограмму обслуживания прерываний, которая отвечает на требование прерывания, и по окончании ее МП возвращается в основную программу.

Указатель стека подобен счетчику команд в том смысле, что в нем содержится адрес, который он инкрементирует или декрементирует, он может быть также загружен новым адресом. Емкость указателя стека составляет 16 бит, т. е. он может посылать адрес по 16 линиям.

Регистр данных/адреса составляется из двух 8-разрядных регистров, которые могут быть использованы вместе или раздельно; они обозначены H и L соответственно старшему (HIGH) и младшему (LOW) байтам. Когда эти два регистра используются вместе, мы обращаемся к паре HL. Регистры H и L являются универсальными подобно аккумулятору в том смысле, что они могут быть инкрементированы, декрементированы, загружены данными и служить источником данных. Пара HL может служить также адресным регистром и хранить адрес назначения в ходе размещения данных в памяти или адресом источника в ходе загрузки аккумулятора. Некоторые микропроцессоры обладают специальным регистром — счетчиком данных, который указывает на ячейку памяти (он используется подобно паре регистров HL нашего МП).

5.4. Использование регистра адреса/данных

Использование пары регистров *HL* в качестве указателя адреса является интересным свойством типового МП. Обычно рассматривают его использование в качестве указателя адреса, когда она временно берет на себя роль основного счетчика команд, указывая адрес ячейки памяти или УВВ. Многие широко распространенные МП (например, Intel 8080/8085, Z80) содержат регистры такого типа. Регистры адреса/данных в рассматриваемом типовом МП называются также парой HL-регистров, регистром адреса, счетчиком данных или указателем адреса. Рассмотрим простую задачу сложения содержимого трех последовательных ячеек памяти с размещением суммы в следующей ячейке памяти (выполнение ее показано на рис. 3.9). Программа загружена в ячейки памяти 2000H— 200AH, а три слагаемых (OCH+OAH+07H) — в ячейки памяти 2100H—2102H. Программа содержит 6 команд, записанных справа на рис. 3.9. Не следует забывать, что текущая сумма будет всегда помещаться в аккумулятор содержащий вначале первое слагаемое (OCH).

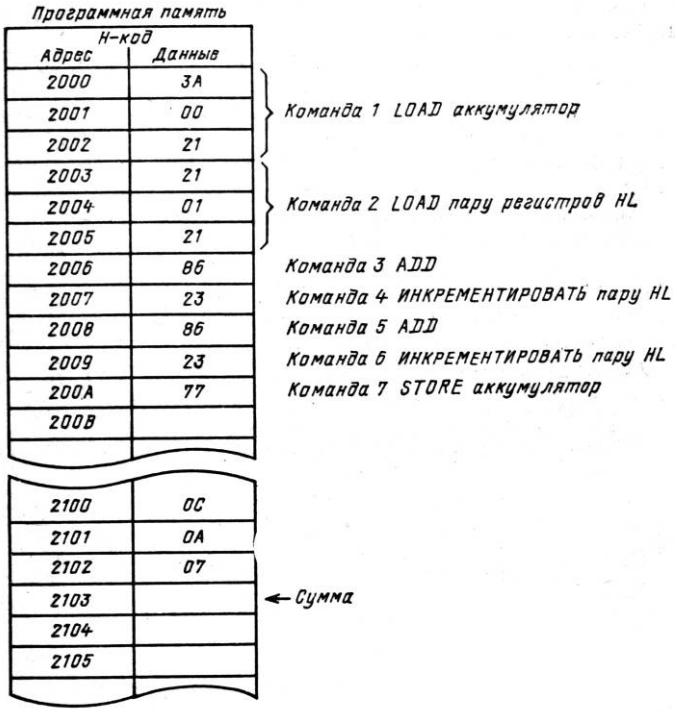


Рис. 5.7 Воображаемая память и команды в примере сложения

Команда 1 имеет КОП 3АН (рис.5.7) и приказывает МП ЗАГРУЗИТЬ (LOAD) в аккумулятор содержимое ячейки памяти 2100Н. Выполнение этой команды прямой загрузки аккумулятора показано на рис. 3.10, *а.* После выполнения команды аккумулятор будет содержать первое слагаемое (ОСН).

Команда *2* приказывает МП загрузить (LOAD) 2101Н в пару регистров *HL,* емкость которых 16 бит. Это число представляет собой адрес памяти данных. Заметим что содержимое первой ячейки памяти (2004Н) загружается в младший байт *L,* следующей за ней — в старший байт *Н* пары регистров *HL.*.

Команда *3* приказывает МП выполнить операцию сложить (ADD) содержимое аккумулятора с содержимым ячейки памяти, адрес которой содержится в паре регистров *HL.* Пара регистров *HL* указывает на ячейку памяти 2101Н и АЛУ складывает свое содержимое ($0000\ 1010_2$) с содержимым аккумулятора ($0000\ 1100_2$), что дает сумму ($0001\ 0110_2$), помещаемую в аккумулятор.

Команда *4* приказывает МП инкрементировать (увеличить на 1) содержимое пары регистров *HL.* Заметим, что изменился только младший байт папы регистров *HL.*

Команда 5 снова приказывает МП сложить (ADD) содержимое аккумулятора и ячейки памяти с адресом 2102Н на которую указывает пара регистров *HL.*

Оба содержимых складываются, что дает сумму ($0001\ 1101_2$), помещаемую в аккумулятор.

По команде *6* содержимое пары регистров *HL* снова инкрементируется.

Команда *7* приказывает МП поместить (STORE) содержимое аккумулятора, т.е. окончательную сумму ($0001\ 1101_2$) в ячейку памяти, на которую указывает пара регистров *HL*, т.е. по адресу 2103Н.

Команды *3, 5,* 7, взаимодействующие с парой регистров *HL* как с указателем адреса, используют косвенно-регистровый способ адресации.

5.5. Использование указателя стека

Типовой микропроцессор содержит указатель стека— специализированный 16-разрядный регистр-счетчик, содержимым которого всегда является адрес. Этот адрес принадлежит особой группе ячеек памяти данных, которая называется *стеком.* В некоторых МП стек может быть составлен из группы физически локализованных на кристалле МП ячеек памяти. Когда микропроцессором выполнялась подпрограмма обслуживания прерывания, текущие данные во всех регистрах МП должны были временно сохраняться. Эта сохранность обеспечена стеком. А когда подпрограмма полностью выполнена, содержимое счетчика команд должно быть сохранено таким образом, чтобы МП мог возвратиться в соответствующее место в программной памяти. Зона временной памяти является стеком.

Стек типового микропроцессора будет содержаться в ОЗУ, и его положение определяется программистом. Указатель стека загружается старшим адресом, представляющим собой вершину стека. В этом случае указатель стека содержит адрес на единицу старше первой ячейки памяти стека.

Данные можно записать в стек, используя команды PUSH (поместить) или CALL (вызвать). Они могут быть считаны из стека по командам POP (извлечь) или RETURN (возврат). Стек функционирует как память с последовательным доступом по типу: данные, поступившие последними, извлекаются первыми (тип LIFO от *Last In— First Out* — последний входит — первый выходит, или FILO от *First In— Last Out* — первый входит — последний выходит).

Извлечение данных из стека и их восстановление в регистре адреса/данных является действием, обратным операции загрузки в стек (PUSH). Команды PUSH и POP используются всегда совместно, однако между ними располагаются другие команды, которые меняют данные, содержащиеся в регистрах МП.

Типовой МП загружает в стек и извлекает содержимое пары регистров. Некоторые МП осуществляют эти операции только с однобайтовыми регистрами, единственной командой. В других МП указатель стека может указывать на пустую ячейку памяти по ближайшему, большему по отношению к вершине стека адресу вместо указания на вершину стека, как в предыдущем случае. При этих условиях имеется возможность сохранить в памяти то, что программист загрузил в начале в указатель стека для определения его адреса.

6. ПРОГРАММИРОВАНИЕ МИКРОПРОЦЕССОРА

6.1. Машинный код и ассемблер

На своем рабочем уровне микропроцессор реагирует на список операций, называемый машинной программой. Программа в машинном коде становится проще для восприятия, когда она представлена в шестнадцатеричном коде (Н-коде). Однако, хотя двоичные данные приведены в шестнадцатеричном коде, эта часть программы всегда рассматривается как заданная на машинном языке и оказывается трудной для понимания.

Возникает вопрос: как перейти от этой формы человеческого языка, иногда длинной и сложной, к машинному языку? Ответ состоит в использовании языка простого программирования — от самого высокого уровня до машинного. *Ассемблер* использует слова и фразы, преобразуя их в машинный код микропроцессора.

Обычно фраза или заданная величина на ассемблере будет соответствовать выражению длиной от одного до трех байт машинного языка. Суть и процедура ассемблирования - команда программы представляется ***мнемоникой*** *(мнемокодом)* – сокращенной символьной записью команды. Программа на языке ассемблер, записанная человеком, могла бы быть представлена в виде табл. 6.1.

Таблица 6.1. *Программа на языке ассемблер*

Метка	Мнемо-ника	Операнд	Комментарий
	MVI	*A,* E4H	Загрузить в аккумулятор данные, следующие непосредственно: B4H
	CMA		Инвертировать содержимое аккумулятора
	STA	21 00H	Разместить содержимое аккумулятора
	HLT		Остановить МП

Обычным является деление объявлений на машинном языке на четыре поля: метка; мнемоника; операнд и комментарий. *Поле метки* используется не всегда и в этой программе остается пустым. *Поле мнемоники* содержит точную мнемонику, установленную разработчиком, которая обычно указывает программе ассемблера операцию для выполнения. *Поле операнда* содержит информацию о регистрах, данных и адресах, объединенных соответствующей операцией. Используя информацию только полей мнемоники и операнда, ассемблер может выдать соответствующий код на машинном языке. Можно также назначить ячейки памяти списку команд. Поле комментариев не учитывается ассемблером. Это поле очень важно, поскольку позволяет понять события в программе.

Как только программа составлена, она представляется затем в виде табл. 6.2. Таким образом, задача ассемблирования (или составление программы на ассемблере) состоит из этапов: 1) перевод мнемоники и операндов на машинный язык; 2) назначение последовательно ячейки памяти каждому КОП и операнду. Переход от версии табл. 6.1 к ассемблированной версии табл. 6.2 может быть выполнен либо вручную, либо на машине при помощи специальной программы интерпретатора ассемблера.

Таблица 6.2. *Программа в машинных кодах и на языке ассемблер*

Адрес, Н-код	Содержи-мое, Н-код	Метка	Мнемони-кА	Операнд	Комментарий
2000	3E		MVI	*A,* B4H	Загрузить аккумулятор данны-ми, следующими епосредствен-но за КОП, B4H
2001	B4				
2002	2F		CMA		Инвертировать содержимое ак-кумулятора
2003	32		STA	2100H	Поместить содержимое аккуму-

2004	00				лятора в ячейку памяти 2100H
2005	21				
2006	76		HLT		Остановить МП

Программа, состоящая из символических команд (фрагмент показан в табл. 6.1), иногда называется *исходной программой,* а переведенная однажды на машинный язык — уже *объектной программой* (содержащей объектные коды команд и данные).

Программирование на языке ассемблер является способом «очеловечивания» действий микропроцессора. Языки высокого уровня (Бейсик, Паскаль и т. д.) при их использовании делают программирование более удобным. Например, одна команда на Бейсике или Паскале может соответствовать 20 или 30 машинным командам.

6.2. Простой состав команд

Введем состав команд, относящихся к типовому микропроцессору, приведенному на рис. 5.6. Мнемоника и КОП, которые будут использованы, представляют собой подсостав команд микропроцессоров Intel 8080/8085. На рис. 6.1 в сокращенном варианте приведены регистры типового МП, предоставляемые программисту. Вверху справа мы видим универсальный 8-разрядный регистр-аккумулятор *A,* а слева показан 8-разрядный регистр состояния. В составе этого регистра индикатор переноса *CY* находится в позиции 7, а индикатор нуля Z — в позиции 0. Позиции бит от первого до шестого регистра состояния в типовом МП не используются, но в выпускаемых МП индикаторов больше, чем здесь. Во второй строке на рис. 6.1 расположены регистры H и L. Это универсальные регистры адреса/данных. Они могут использоваться раздельно или в форме пары регистров (мы говорим тогда о паре регистров *HL).* В последнем случае они используются как указатель адреса.

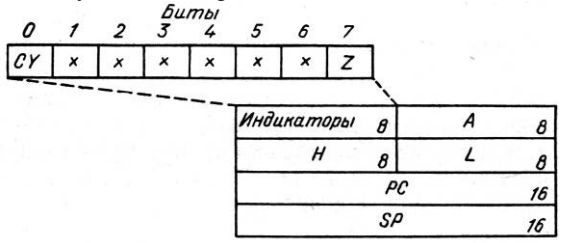


Рис. 6.1 Программно-доступные регистры типового МП

Снизу на рис. 6.1 находятся два специальных 16-разрядных регистра. Счетчик команд *PC (Program Counter* - программный счетчик) указывает МП следующую для выполнения команду. Указатель стека *SP (Stack Pointer)* содержит адрес вершины стека. Сам стек находится в ОЗУ.

Состав команд такого типового микропроцессора разделен на семь категорий: арифметические; логические; передачи данных; ветвления; вызова подпрограмм; возврата из подпрограмм; прочие.

Типовой микропроцессор в состоянии выполнить только 67 различных команд из 239, которые входят в состав МП Intel 8085.

6.3. Состав команд арифметических действий

Первыми рассмотрим команды арифметических действий. Они сведены в табл. 6.3 и содержат команды сложить, вычесть, инкрементировать, декрементировать, сравнить. Заметим по данным табл. 4.3, что существуют четыре команды сложения. Аккумулятор (регистр *A)* содержит одно из слагаемых. Каждая команда точно оговаривает различные источники другого слагаемого.

Рассмотрим первую из команд сложения в табл. 6.3. Команда сложить непосредственно является двухбайтовой. Ее формат КОП (в этом случае C6H) содержится в первом байте команды, непосредственно за ним — во втором байте — находятся данные для сложения с содержимым аккумулятора. Команда ADI выполняется по схеме, приведенной на рис. 6.2, *а.* Данные, находящиеся в памяти непосредственно за КОП, складываются с содержимым аккумулятора ($0000\ 1111_2$**).** Сумма ($0001\ 1111_2$) помещается в аккумулятор.

Табл. 6.3 Состав команд арифметических операций типового микропроцессора

Операция	Адресация	Мнемоника	КОП	Байты	Формат команды	Символика	Индикаторы
Сложить *A* с данными	Непосредственная	ADI	C6	2	КОП данные	(A) ← (A) + (байт 2)	*Z, CY*
Сложить *L* с *A*	Регистровая	ADD *L*	85	1	КОП	(A) ← (A) + (L)	*Z, CY*
Сложить *H* с *A*	»	ADD *H*	84	1	КОП	(A) ← (A) + (H)	*Z, CY*
Сложить LOC (HL) с *A*	Косвенная регистровая	ADD *M*	86	1	КОП	(A) ← (A) + ((H) (L))	*Z, CY*
Вычесть данные из *A*	Непосредственная	SUI	6	2	КОП данные	(A) ← (A) — (байт 2)	*Z, CY*
Вычесть *L* из *A*	Регистровая	SUB *L*	95	1	КОП	(A) ← (A) — (L)	*Z, CY*
Вычесть *H* из *A*	Регистровая	SUB *H*	94	1	КОП	(A) ← (A) — (H)	*Z, CY*
Вычесть LOC (HL) из *A*	Косвенная регистровая	SUB *M*	96	1	КОП	(A) ← (A) — ((H) (L))	*Z, CY*
Инкремент *A*	Регистровая	INR *A*	3C	1	КОП	(A) ← (A) + 1	*Z*
Инкремент *HL*	»	INX *H*	23	1	КОП	(HL) ← (HL) + 1	
Декремент *A*	»	DCR *A*	3D	1	КОП	(A) ← (A) — 1	*Z*
Декремент *HL*	»	DCX *H*	2B	1	КОП	(HL) ← (HL) — 1	
Сравнить *A* с данными	Непосредственная	CPI	FE	2	КОП данные	(A) ← (байт 2)	*Z* =1, если (A)=(байт 2); *CY* =1, если (A)<(байт 2)
Сравнить *A* с *L*	Регистровая	CMP *L*	BD	1	КОП	(A) — (L))	*Z*=1, если (A)=(*L*); *CY*= =1, если (A)<(*L*)
Сравнить *A* с *H*	»	CMP *H*		1	КОП	(A) — (H)	*Z* = 1, если (A) = (H); *CY* = 1, если (A) < (H)
Сравнить *A* с LOC (HL)	Косвенная регистровая	CMP *M*		1	КОП	(A) — ((H) (L))	*Z*=1, если (A)=((H)(L)); *CY*=1, если (A)<((H)(L))

() — содержимое; (()) — косвенная регистровая адресация.

Второй является команда сложить содержимое регистров *L* и *A* (мнемоника ADD *L),* на рис. 6.2, *б* показано ее выполнение. Содержимое аккумулятора ($0000\ 1000_2$) складывается с содержимым регистра *L* ($0000\ 0001_2$), получающаяся в результате выполнения команды ADD *L* сумма ($0000\ 1001_2$) помещается в аккумулятор.

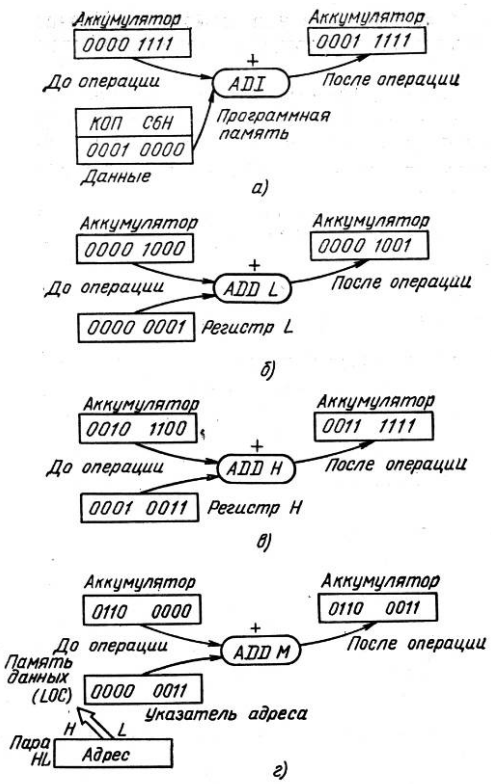


Рис. 6.2 Выполнение МП операций сложения

Третья команда в табл. 6.3 представляет собой однобайтовую команду сложить содержимое регистров H и A (мнемоника ADD H). Это другая команда сложения содержимого одного регистра с содержимым другого (как и предшествующая), и она выполняется в последовательности, приведенной на рис. 6.2, *в.* Содержимое регистра *A* ($0010\ 1100_2$) сложено с содержимым регистра H ($0001\ 0011_2$), сумма (ООП 1111_2) помещена в аккумулятор.

Четвертая строка таблицы представляет собой однобайтовую команду сложить с косвенным регистром (мнемоника ADD *M).* Адрес данного слагаемого числа задан в более сложной форме с использованием способа косвенной регистровой адресации. Рисунок 6.2, *г* является примером выполнения

команды ADD *M*. Пара *HL* указывает 16-разрядный адрес памяти, т.е. LOC (*Location* - место расположения). Содержимое LOC ($0000\ 0011_2$) сложено с содержимым аккумулятора ($0110\ 0000_2$), сумма ($0110\ 0011_2$) помещена в аккумулятор. Команды косвенного сложения используют в качестве указателя адреса 16-разрядный регистр (обычно пару *HL).*

В силу внутренних особенностей АЛУ не обладает возможностями вычитания, оно осуществляет сложение, представляя вычитаемое в форме дополнительного кода и затем складывая его. Вычитание двоичных чисел осуществляют сложением первого числа со вторым, представленным в форме дополнительного кода, пренебрегая переполнением. Микропроцессор использует переполнение для установления индикатора переноса. Вычитая, МП инвертирует переполнение, и результат становится содержимым индикатора переноса CY. Переполнение 1 инвертируется и сбрасывает индикатор переноса в 0. Когда в ходе вычитания индикатор переноса сбрасывается, это значит, что переноса не было и что первое число больше второго.

Таким образом, арифметические операции типового микропроцессора могут выполняться по большому числу команд из всего состава. Многие МП снабжены несколькими дополнительными арифметическими командами и индикаторами в их регистре состояния. Использование индикатора при ветвлениях и переходах по результатам операций будет впоследствии рассмотрено.

6.4. Состав команд логических операций

Логические команды составляют еще одну группу команд типового микропроцессора. Они сведены в табл. 6.4 и содержат команды И, ИЛИ, ИЛИ ИСКЛЮЧАЮЩЕЕ, НЕ (инверсия) и сдвига. Здесь именно аккумулятор составляет ядро большинства операций. Как и при арифметических командах, способ адресации и здесь влияет на способ и место нахождения других данных в системе. Логические операции выполняются МП побитно.

Табл. 6.4 Состав команд логических операций типового микропроцессора

Операция	Адресация	Мнемоника	КОП	Байты	Формат команды	Символика	Установка индикаторов
A И данные	Непосредственная	ANI	E6	2	КОП данные	(A) ← (A)·(байт 2)	Z *CY* — Сброс
A И *L*	Регистровая	ANA *L*	A5	1	КОП	(A) ← (A)·(L)	Z *CY* — Сброс
A И *H*	»	ANA *H*	A4	1	КОП	(A) ← (A)·(H)	Z *CY* — Сброс
A И LOC (HL)	Косвенная регистровая	ANA *M*	A6	1	КОП	(A) ← (A)·(H)	Z *CY* — Сброс
A ИЛИ данные	Непосредственная	ORI	F6	2	КОП данные	(A) ← (A) + (байт 2)	Z *CY* — Сброс
A ИЛИ *L*	Регистровая	ORA	B5	1	КОП	(A) ← (A) + (L)	Z *CY* — Сброс
A ИЛИ *H*	»	ORA *H*	B4	1	КОП	(A) ← (A) + (H)	Z *CY* — Сброс
A ИЛИ LOC (HL)	Косвенная регистрация	ORA *M*	B6	1	КОП	(A) ← (A) + ((H)(L))	Z *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ данные	Непосредственная	XRI	EE	2	КОП данные	(A) ← (A) ⊕ (байт 2)	Z *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ *A*	Регистровая	XRA *A*	AF	1	КОП	(A) ← (A) ⊕ (A) Сброс A	Z = 1 *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ *L*	»	XRA *L*	AD	1	КОП	(A) ← (A) ⊕ (L)	Z *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ *H*	»	XRA *H*	AC	1	КОП	(A) ← (A) ⊕ (H)	Z *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ LOC (HL)	Косвенная регистровая	XRA *M*	AE	1	КОП	(A) ← (A) + ((H)(L))	Z *CY* — Сброс
Инвертировать *A*	Неявная	CMA	2F	1	КОП	(A) ← $(\overline{A})$	
Сдвиг *A* вправо	»	RAR	1F	1	КОП		*CY*
Сдвиг *A* влево	»	RAL	17	1	КОП		*CY*

() — содержимое; (()) — косвенная регистровая адресация; · — И; + — ИЛИ; ⊕ — ИЛИ ИСКЛЮЧАЮЩЕЕ.

Используя одну или несколько команд циклического сдвига, можно тестировать весь заданный состав бит, а индикатор переноса может быть сброшен или установлен. Индикатор переноса может быть тестирован затем командой условного ветвления. Тест паритета является другим приложением использования команд сдвига. Паритет двойного числа определяется числом содержащихся в нем единиц: четный паритет — общее число единиц четное; нечетный паритет — общее число единиц нечетное. Заметим, что команды сдвига оперируют только данными аккумулятора и не требуют других операндов, расположенных в памяти или регистрах. Поэтому здесь способ Iадресации называется неявным, а иногда вообще не указывается. Многие МП обладают несколькими иными, чем типовой микропроцессор, типами команд сдвига.

Таким образом, команды логических операций используются для манипуляции с переменными по законам алгебры логики. Они могут быть использованы для тестирования и сравнения бит.

6.5. Состав команд операций передачи данных

Команды передачи данных составляют третью категорию команд типового микропроцессора. Они выполняют передачу данных из регистра в регистр, размещение данных в памяти, размещение извлеченных

из памяти данных в УВВ и установление индикатора переноса (табл. 6.5). Почти все команды передачи не влияют на индикаторы МП. Эта группа содержит множество команд, данные могут быть переданы из любой ячейки памяти в любой регистр или наоборот. Каждая команда передачи содержит адреса источника и назначения данных. Команды передачи данных не меняют содержимого источника данных.

Табл. 6.5 Состав команд передачи данных типового микропроцессора

Операция	Адресация	Мнемоника	КОП	Байты	Формат команд	Символика
Передать *L* в *A*	Регистровая	MOV *A*, *L*	7D	1	КОП	(A) ← (L)
Передать *H* в *A*	»	MOV *A*, *H*	7C	1	КОП	(A) ← (H)
Передать *A* в *L*	»	MOV *L*, *A*	6F	1	КОП	(L) ← (A)
Передать *A* в *H*	»	MOV *H*, *A*	67	1	КОП	(H) ← (A)
Передать *HL* в *PC*	»	PCHL	E9	1	КОП	(PC) ← (HL)
Передать *HL* в *SP*	»	SPHL	F9	1	КОП	(SP) ← (HL)
Загрузить *A* данными	Непосредственная	MVI *A*	3E	2	КОП данные	(A) ← (байт 2)
Загрузить *L* данными	»	MVI *L*	2E	2	КОП данные	(L) ← (байт 2)
Загрузить *H* данными	»	MVI *H*	26	2	КОП данные	(H) ← (байт 2)
Загрузить LOC (HL) в *A*	Косвенная регистровая	MOV *A*, *M*	7E	1	КОП	(A) ← ((H)(L))
Загрузить *HL* данными	Непосредственная	LXI *H*	21	3	КОП мл. байт ст. байт	(L) ← (байт 2) (H) ← (байт 3)
Загрузить *SP* данными	»	LXI *P*	31	3	КОП мл. байт ст. байт	(SP) ← (байт 2 + 3)
Загрузить *HL* из LOC	Прямая	LHL *D*	2A	3	КОП мл. адрес ст. адрес	(L) ← (байт 2 + 3) (H) ← ((байт 2 + 3)+1)
Загрузить *A* из LOC	»	LDA	3A	3	КОП мл. адрес ст. адрес	(A) ← ((байт 2 + 3))
Поместить *A* в LOC*aa*	»	STA	32	3	КОП мл. адрес ст. адрес	(адрес) ← (A)
Поместить *HL* в LOC	»	SHLD	22	3	КОП мл. адрес ст. адрес	(адрес) ← (L) (адрес + 1) ← (H)
Поместить *A* в LOC*aa* (HL)	Косвенная регистровая	MOV *M*,*A*	77	1	КОП	((H)(L)) ← (A)
Поместить *L* в LOC (HL)	То же	MOV *M*,*L*	75	1	КОП	((H)(L)) ← (L)
Поместить *H* в LOC (HL)	»	MOV *M*,*L*	74	1	КОП	((H)(L)) ← (H)
Ввести в *A* из порта в LOC*a*	Прямая	IN	DB	2	КОП адр. порта	(A) ← (адрес порта)
Вывести *A* в порт в LOC*a*	»	OUT	D3	2	КОП адр. порта	(адрес порта) ← (A)
Установить индикатор переноса	Неявная	STC	37	1	КОП	(CY) ← 1

) — содержимое; (()) — косвенная регистровая адресация; *PC*—счетчик команд; *SP* — указатель стека; устанавливается индикатор
CY только для команды *STC*.

Команда ввода идентична команде загрузки. Источник данных передачи является портом ввода, идентифицированным двоичным 8-разрядным числом (0—255ю), назначение этих данных — аккумулятор МП. Данные порта ввода 0000 1111, на который указывает второй байт команды, передаются в аккумулятор, исходя из порта ввода, идентифицированного LOC.

6.6. Состав команд операций ветвления

Команды ветвления составляют четвертую группу команд, которой снабжен типовой МП. Они приведены в табл. 6.6. Термины *ветвление* и *переход* в данном случае синонимы. Некоторые разработчики усматривают разницу между ними. Эти команды называют иногда *командами передачи управления.*

Обычно микро-ЭВМ выполняет команды последовательно. Шестнадцатиразрядный счетчик команд типового микропроцессора хранит всегда адрес следующей извлекаемой из памяти команды до ее выполнения. Содержимое его обычно повышается в каждом счете. Команды ветвления или перехода являются средством изменения значения содержимого счетчика команд и, следовательно, изменения нормальной последовательности выполнения программы.

Эти команды разделены на две группы: *безусловного перехода* и *условного перехода.* Первая команда в табл. 6.6 является командой безусловного перехода. Команда ПЕРЕЙТИ непосредственной адресации является трехбайтовой, используемой для изменения специфического адреса в счетчике команд МП. Можно рассматривать команду ветвления или безусловного перехода как способ загрузки новой информации об адресе в счетчик команд.

Табл. 6.6 Состав команд ветвления и перехода типового микропроцессора

Операция	Адресация	Мнемоника	КОП	Байты	Формат команды	Символика
Перейти в LOC	Непосредственная	JMP	C3	3	КОП мл. адрес ст. адрес	(PC) ← (адрес)
Перейти в LOC, если 0	»	JZ	CA	3	КОП мл. адрес ст. адрес	Если $Z=1$, то (PC) ← (адрес)
Перейти в LOC, если не 0	»	JNZ	C2	3	КОП мл. адрес ст. адрес	Если $Z=0$, то (PC) ← (адрес)
Перейти в LOC, если перенос	»	JC	DA	3	КОП мл. адрес ст. адрес	Если $CY=1$, то (PC) ← (адрес)
Перейти в LOC, если нет переноса	»	JNC	D2	3	КОП мл. адрес ст. адрес	Если $CY=0$, то (PC) ← (адрес)

Четыре последние команды ветвления в табл. 6.6 являются командами условного перехода. Эти команды повлекут за собой непосредственно загрузку адреса, только если будут выполнены специальные условия. В противном случае счетчик команд будет инкрементирован нормально.

Команды перехода или ветвления существуют практически во всех программах МП. Они очень эффективны как средство принятия решений и удобны для формирования циклов программы. Микропроцессоры особенно эффективны в случае выполнения повторяющихся задач. Циклы в программе являются очень эффективным методом ее сокращения. При написании программы с использованием циклов на ассемблере используются ***символические адреса*** в поле «метка». Они заменяются физическими адресами при транслировании программы в машинный код.

6.7. Состав команд вызова подпрограмм и возврата в основную программу

Эти команды составляют пятую категорию состава команд типового МП. Их только две, и они приведены в табл. 6.7. Команды вызова (CALL) и возврата (RET) всегда используются парами. При их выполнении индикаторы не изменяются.

Табл. 6.7 Состав команд вызова подпрограмм и возврата в основную программу

Операция	Адресация	Мнемоника	КОП	Байты	Формат команды	Символика
Вызов подпрограммы	Непосредственная, косвенная регистровая	CALL	CД	3	КОП мл. адрес ст. адрес	((SP)—1)←(PCH) ((SP)—2)←(PCL) (SP)←(SP—2) (PC)←(адрес)
Возврат из подпрограммы	Косвенная регистровая	RET	C9	1	КОП	(PCL)←((SP)) (PCH)←((SP)+1) (SP)←(SP)+2

() — содержимое; (()) — косвенная регистровая адресация; *PC* — счетчик команд; *SP* — указатель стека.

Трехбайтовая команда CALL используется основной программой для перехода МП (или ветвления) к подпрограмме. Когда МП передает первую команду CALL, он находит адрес перехода в двух следующих байтах программы. Адрес следующей команды за CALL отправляется в стек, и МП переходит тогда в начало подпрограммы. Команды подпрограммы выполняются пока МП не передаст команду возврата (RET). Сохраняющийся в стеке адрес отыскивается счетчиком команд, и МП продолжает выполнение основной программы, принимая ее там, где он ее покинул. Подпрограмма может быть использована много раз в ходе выполнения одной и той же основной программы. Подпрограмма может быть расположена в ОЗУ или ПЗУ.

6.8. Состав команд прочих опреаций

Эти команды составляют последнюю категорию, которыми наделен типовой микропроцессор. Они сведены в табл. 6.8 и содержат команды помещения в стек, извлечения из стека, отсутствия операции и команду остановки. При их выполнении индикаторы не изменяются.

Команда PUSH *PSW* - помещает в стек *A* и индикаторы. Эта команда является однобайтовой, содержимое аккумулятора помещается первым, а регистра состояния - вторым.

Команда NOP (no operation - нет операций) соответствует отсутствию всякого выполнения операций в течение 1 или 2 мкс. Это однобайтовая команда, единственным эффектом которой является инкремент

счетчика команд. Никакой другой регистр не затрагивается. Эта команда используется как дополнение (когда одна или две команды отменены в ходе наладки) и связывает две части программы так, чтобы МП мог обратиться от одной к другой. Она может также служить для ввода интервала времени в цикл временной задержки.

Табл. 6.8 Прочие команды типового микропроцессора

Операция	Адресация	Мнемоника	КОП	Байт	Формат команды	Символика
Поместить в стек *A* и индикаторы	Косвенная регистровая	PUSH *PSW*	F5	1	КОП	((SP) — 1) ← (A) ((SP)—2) ← (индикаторы) (SP) ← (SP) — 2
Поместить в стек *HL*	То же	PUSH *H*	E5	1	КОП	((SP) — 1) ← (H) ((SP) — 2) ← (L) (SP) ← (SP) — 2
Извлечь из стека *A* и индикатора	»	POP *PSW*	F1	1	КОП	(индикаторы) ← ((SP)) (A) ← ((SP) + 1) (SP) ← (SP) + 2
Извлечь из стекла *HL*	»	POP *H*	E1	1	КОП	(L) ← ((SP)) (H) ← ((SP) + 1) (SP) ← (SP) + 2
Нет операций	Неявная	NOP	00	1	КОП	(L) ← ((SP)) (H) ← ((SP) + 1)
Останов	Неотделим	HLT	76	1	КОП	(SP) ← (SP) + 2 (PC) ← (PC) + 1

() — содержимое; (()) — косвенная регистровая адресация; *A* — аккумулятор; *SP* — указатель стека.

Команда HLT (останов) используется в конце программы для остановки микропроцессора. В этом случае только СБРОС или команда вызова прерывания может позволить новый запуск типового микропроцессора.

6.9. Способы адресации

Способы адресации нашего типового МП следующие: 1) неявный; 2) регистровый; 3) непосредственный; 4) прямой; 5) косвенный регистровый.

Два первых (регистровый и неявный) касаются операндов, расположенных в самом МП. Три последних (непосредственная, прямая и косвенная регистровая) — операндов, расположенных вне МП, т. е. в ячейках памяти или портах УВВ. Эти способы адресации присущи МП Intel 8080/8085.

Команды с *неявной адресацией* – те, которые не требуют операндов. Например, команда STC (восстановить индикатор переноса) не затрагивает другие регистры или индикаторы.

В случае *регистровой адресации* операнд отыскивается в одном из внутренних регистров МП. Оба эти типа команд всегда однобайтовые.

В случае *непосредственной адресации* операнд поступает в программную память из следующего байта (двух следующих байтов) за КОП. Такие команды занимают 2 или 3 байта.

В случае *прямой адресации* 2-й и 3-й байт команды прямо указывают на адрес операнда. Они являются адресами в прямом способе адресации, тогда как при непосредственной адресации эти же байты были операндами. Такие команды занимают 2 или 3 байта.

В случае *косвенной регистровой адресации* пара регистров *HL* указывает на адрес операнда в памяти. Рассмотрим пример: загрузить LOC *(H* и *L)* в A с мнемоникой MOV A, M. Этот тип команд всегда однобайтовый.

Другие МП снабжены иногда иными способами прямой адресации, а именно: нулевой или основной страницы; абсолютной; адресации действующей страницы или ожидаемой.

7. Проектирование микропроцессорных систем

7.1 Уровни представления микропроцессорной системы

Микропроцессорная система может быть описана на различных уровнях абстрактного представления.

Существующую микропроцессорную систему можно описать на любом известном уровне представления, но в начальной стадии проектирования ее можно описать только на концептуальном уровне.

В процессе разработки системы происходит переход от одного уровня ее представления к другому, более детальному. Каждая абстракция несет в себе только информацию, которая соответствует данному уровню, и не содержит каких-либо сведений относительно более низких уровней. Микропроцессорная система может быть описана, например, на одном из следующих уровней абстрактного представления:
1) "черный ящик";
2) структурный;
3) программный;
4) логический;
5) схемный .

На уровне "черного ящика" микропроцессорная система описывается внешними спецификациями; перечисляются внешние характеристики.

Структурный уровень создается компонентами микропроцессорной системы: микропроцессорами, запоминающими устройствами, устройствами ввода/вывода, внешними запоминающими устройствами, каналами связи. Микропроцессорная система описывается функциями отдельных устройств и их взаимосвязью, информационными потоками.

Программный уровень разделяется на два подуровня: команд процессора и языковой. Микропроцессорная система интерпретируется как последовательность операторов или команд, вызывающих то или иное действие над некоторой структурой данных.

Логический уровень присущ исключительно дискретным системам. На этом уровне выделяются два подуровня: переключательных схем и регистровых пересылок. Подуровень переключательных схем образуется вентилями и построенными на их основе операторами обработки данных. Переключательные схемы подразделяются на комбинационные и последовательностные; первые в отличие от последних не содержат запоминающих элементов. Поведение системы на этом уровне описывается алгеброй логики, моделью конечного автомата, входными/выходными последовательностями 1 и 0. Комбинационные схемы представляются таблицей истинности, в которой каждому входному набору значений сигналов ставится в соответствие набор значений сигналов на выходах. Последовательностные схемы могут описываться диаграммами или таблицами входов/выходов, в которых определены взаимно однозначные соответствия между входами схемы, внутренними состояниями (комбинациями значений элементов памяти) и выходами. Подуровень регистровых пересылок характеризуется более высокой степенью абстрагирования и представляет собой описание регистров и передачу данных между ними. Он включает в себя две части: информационную и управляющую. Информационная часть образуется регистрами, операторами и путями передачи данных. Управляющая часть определяет зависящие от времени сигналы, инициирующие пересылку данных между регистрами.

Схемный уровень образуется резисторами и конденсаторами. Показателями поведения системы на этом уровне служат напряжение и ток, представляемые в функции времени или частоты. Этот уровень описания дискретной системы широко используется в описаниях аналоговых систем и не является ни наинизшим из возможных, ни достаточным для полной характеристики системы.

7.2 Ошибки, неисправности, дефекты

В жизненном цикле микропроцессорной системы, как любой дискретной системы, выделяются три стадии: проектирование, изготовление и эксплуатация. Каждая из стадий подразделяется на несколько фаз, для которых существуют вероятности возникновения конструктивных или физических неисправностей, приводящих систему в неработоспособное состояние. Поэтому на каждой фазе необходимы процедуры тестового контроля, направленные на обнаружение и локализацию неисправностей. Процедура тестового контроля может быть определена как проведение экспериментов с "черным ящиком". Дискретная система любой сложности или часть такой системы может рассматриваться как "черный ящик" с множеством входов и выходов. Правильность функционирования этого "черного ящика" должна устанавливаться путем подачи входных сигналов и наблюдения ответных выходных сигналов системы. В тех случаях, когда поведение "черного ящика" отличается от нормального, характеризуемого его спецификацией или представлениями человека, говорят о наличии ошибки. Ошибка вызывается некоторой неисправностью, представляющей собой некорректное состояние внутри "черного ящика". Неисправности классифицируют в соответствии с их причинами: физическая, если причиной ее служат либо дефекты элементов, либо физическое воздействие окружающей среды; субъективная (внесенная, нефизическая), если ее причиной служат ошибки проектирования, неправильный монтаж элементов, грубые ошибки оператора. Физические неисправности - непредусмотренные, нежелательные изменения значения одной или нескольких логических переменных в системе. Субъективные неисправности - конкретные проявления недостатков программного и аппаратного обеспечения и неправильных действий оператора, имеющих место при выполнении дискретной системой предписанных спецификацией действий .

Под субъективными неисправностями подразумеваются неисправности нефизические, вызванные недостатками различных схем, конструкций, программ, средств эксплуатации - компиляторов, ассемблеров, программ автоматизации проектирования, инструкции по эксплуатации, процедур и средств контроля и т.д. Субъективные неисправности делят на проектные и интерактивные.

Проектные неисправности вызваны недостатками, вносимыми в систему на различных стадиях реализации исходного задания при структурном проектировании, разработке алгоритмов, написании программ, трансляции в машинный код, детальном логическом и техническом проектировании, а также при последующих модификациях аппаратного и программного обеспечения. Интерактивные неисправности возникают, когда в процессе работы, технического обслуживания или отработки системы оператор вводит в нее через интерфейс человек-машина ложную информацию, не соответствующую текущему состоянию системы. Как правило, это происходит в результате непонимания инструкции для оператора или вследствие неточностей ввода информации.

Ошибка - проявление неисправности (физической или субъективной). В зависимости от уровня иерархической структуры системы термин "ошибка" может иметь различный смысл. Так, для дискретного устройства он означает появление неверных двоичных сигналов ("0" вместо "1" и "1" вместо "0"); для программы ошибка означает отклонение поведения программы от заданного, приводящее к выдаче неверных результатов.

Следует четко разграничивать понятия "ошибка" и "неисправность". Неисправность может приводить или не приводить к ошибке в зависимости от состояния системы. В то же время возникновение ошибки обязательно говорит о существовании какой-то неисправности. Одна и та же ошибка может быть вызвана множеством неисправностей, а одна неисправность может служить причиной целого ряда ошибок. Например, если триггер, предназначенный для хранения кода переполнения разрядной сетки ЭВМ, вследствие неисправности все время находится в состоянии "0", то ошибки из-за неисправности не будет до тех пор, пока в процессе вычислений реально не возникнет арифметическое перепол-нение, при котором триггер останется в состоянии "0" вместо перехода в состояние "1". Однако даже и в этом случае такая ошибка процессора не обязательно приведет к ошибке на программном уровне, если в программе условие переполнения не проверяется и, следовательно, ни с какой стороны не влияет на ее дальнейшее поведение.

Дефекты - физические изменения параметров компонентов системы, выходящие за допустимые пределы. Их называют сбоями, если они носят временный характер, и отказами, если они постоянны.

Существует такая причинноследственная связь:

1) дефект, представляющий собой изменение в значениях параметров компонентов, вызывает неисправность, т.е. отклонение от заданного значения (значений) логической переменной (переменных) в точке дефекта;

2) неисправность приводит к подаче неверных логических значений на вход (входы) остальной части системы и может вызывать ошибки, проявляющиеся при последующей работе других исправных логических схем;

3) ошибки приводят к появлению неправильных результатов или к отклонению от нормального хода исполнения программы.

7.3 Отладка

О правильности функционирования микропроцессорной системы на уровне "черного ящика" с полностью неизвестной внутренней структурой можно говорить лишь тогда, когда произведены ее испытания, в ходе которых реализованы все возможные комбинации входных воздействий, и в каждом случае проверена корректность ответных реакций. Однако исчерпывающее тестирование имеет практический смысл лишь для простейших элементов систем. Следствием этого является тот факт, что ошибки проектирования встречаются при эксплуатации, и для достаточно сложных систем нельзя утверждать об их отсутствии на любой стадии жизни системы. В основе почти всех методов испытаний лежит та или иная гипотетическая модель неисправностей, первоисточником которой служат неисправности, встречающиеся в практике. В соответствии с моделью в рамках каждого метода предпринимаются попытки создания тестовых наборов, которые могли бы обеспечить удовлетворительное выявление моделируемых неисправностей. Любой метод тестирования хорош ровно настолько, насколько правильна лежащая в его основе модель неисправности.

Важным моментом является правильный выбор соотношения между степенью общности модели, стоимостью и степенью сложности формирования и прогона тестов, ориентированных на моделируемые неисправности. Чем конкретнее модель, тем легче создать для нее систему тестов, но тем выше вероятность того, что неисправность останется незамеченной. Если же модель неисправностей излишне общая, то из-за комбинаторного возрастания числа необходимых тестовых наборов и/или времени вычислений, требуемого для работы алгоритмов формирования тестов, она станет непрактичной и пригодной только для несложных систем.

7.4 Обнаружение ошибки и диагностика неисправности

Дефект не может быть обнаружен до тех пор, пока не будут созданы условия для возникновения из-за него неисправности, результат которой должен быть, в свою очередь, передан на выход испытуемого объекта, для того чтобы сделать неисправность наблюдаемой. Метод испытаний должен позволить генерировать тесты, ставящие испытуемый объект в условия, при которых моделируемые неисправности проявляли бы себя в виде обнаруживаемых ошибок. Если испытуемый объект предназначен для

эксплуатации, то при обнаружении ошибки необходимо произвести локализацию неисправности с целью ее устранения путем ремонта или усовершенствования испытуемого объекта.

Диагностика неисправности - процесс определения причины появления ошибки по результатам тестирования. Отладка - процесс обнаружения ошибок и определение источников их появления по результатам тестирования при проектировании микропроцессорных систем. Средствами отладки являются приборы, комплексы и программы .

Точность, с которой тот или иной тест локализует неисправности, называется его разрешающей способностью. Требуемая разрешающая способность определяется конкретными целями испытаний. Например, при испытаниях аппаратуры в процессе эксплуатации для ее ремонта часто необходимо установить, в каком сменном блоке изделия имеется неисправность. В заводских условиях желательно осуществлять диагностику неисправности вплоть до уровня наименьшего заменяемого элемента, чтобы минимизировать стоимость ремонта. В лабораторных условиях в процессе отладки опытного образца необходимо определять природу неисправности (физического или нефизического происхождения). В случае возникновения и проявления дефекта требуется локализовать место неисправности с точностью до заменяемого элемента, а при проявлении субъективной неисправности - с точностью до уровня представления (программного, схемного, логического и т. д.), на котором была внесена неисправность, и места.

Так как процесс проектирования микропроцессорной системы содержит неформализуемые этапы, то отладка системы предполагает участие человека.

Свойство контролепригодности системы.

Успех отладки зависит от того, как спроектирована система, предусмотрены ли свойства, делающие ее удобной для отладки, а также от средств, используемых при отладке. Для проведения отладки проектируемая микропроцессорная система должна обладать свойствами управляемости, наблюдаемости, предсказуемости.

Управляемость - свойство системы, при котором ее поведение поддается управлению, т. е. имеется возможность остановить функционирование системы в определенном состоянии, и затем сноввa ее запустить. Наблюдаемость - свойство системы, позволяющее проследить за поведением системы, сменой ее внутренних состояний. Предсказуемость - свойство системы, позволяющее установить систему в состояние, из которого все последующие состояния могут быть предсказаны.

7.5 Функции средств отладки

Сроки и качество отладки системы зависят от средств отладки. Чем совершеннее приборы, имеющиеся в распоряжении инженера-разработчика, тем скорее можно начать отладку аппаратуры и программ и тем быстрее обнаружить ошибки, локализовать источники, устранение которых обойдется дороже на более позднем этапе проектирования.

Средства отладки должны:

1) управлять поведением системы или/и ее модели на различных уровнях абстрактного представления;
2) собирать информацию о поведении системы или/и ее модели, обрабатывать и представлять на различных уровнях абстракции;
3) преобразовывать системы, придавать им свойства контролепригодности;
4) моделировать поведение внешней среды проектируемой системы.

Под управлением поведением системы или ее модели понимаются определение и подача входных воздействий для запуска или останова системы или ее модели, для перевода в конкретное состояние последних. Чтобы определить место субъективной неисправности, которая может быть внесена на любой стадии проектирования, необходимо уметь собирать информацию о поведении системы и представлять ее в тех формах, которые приняты для данного проекта. Например, это могут быть временные диаграммы, принципиальные электрические схемы, язык регистровых передач, ассемблер и др.

В общем случае нельзя локализовать источник ошибки проектируемой системы, имея информацию о поведении системы только на ее внешних выводах, поэтому проектируемую систему преобразовывают. Например, прежде чем изготовлять однокристальную микроЭВМ с теми или иными "зашивками" ПЗУ, программы отлаживают на эмуляционном кристалле, у которого магистраль выведена на внешние контакты и вместо ПЗУ установлено ОЗУ.

7.6 Этапы проектирования микропроцессорных систем

Микропроцессорные системы по своей сложности, требованиям и функциям могут значительно отличаться надежностными параметрами, объемом программных средств, быть однопроцессорными и многопроцессорными, построенными на одном типе микропроцессорного набора или нескольких, и т.д. В связи с этим процесс проектирования может видоизменяться в зависимости от требований, предъявляемых к системам. Например, процесс проектирования МПС, отличающихся одна от другой содержанием ПЗУ, будет состоять из разработки программ и изготовления ПЗУ.

При проектировании многопроцессорных микропроцессорных систем, содержащих несколько типов микропроцессорных наборов, необходимо решать вопросы организации памяти, взаимодействия с процессорами, организации обмена между устройствами системы и внешней средой, согласования функционирования устройств, имеющих различную скорость работы, и т. д. Ниже приведена примерная последовательность этапов, типичных для создания микропроцессорной системы:

1. Формализация требований к системе.
2. Разработка структуры и архитектуры системы.
3. Разработка и изготовление аппаратных средств и программного обеспечения системы.
4. Комплексная отладка и приемосдаточные испытания.

Этап 1. На этом этапе составляются внешние спецификации, перечисляются функции системы, формализуется техническое задание (ТЗ) на систему, формально излагаются замыслы разработчика в официальной документации.

Этап 2. На данном этапе определяются функции отдельных устройств и программных средств, выбираются микропроцессорные наборы, на базе которых будет реализована система, определяются взаимодействие между аппаратными и программными средствами, временные характеристики отдельных устройств и программ.

Этап 3. После определения функций, реализуемых аппаратурой, и функций, реализуемых программами, схемотехники и программисты одновременно приступают к разработке и изготовлению соответственно опытного образца и программных средств. Разработка и изготовление аппаратуры состоят из разработки структурных и принципиальных схем, изготовления прототипа, автономной отладки. Разработка программ состоит из разработки алгоритмов; написания текста исходных программ; трансляции исходных программ в объектные программы; автономной отладки.

Этап 4. см. Комплексная отладка.

На каждом этапе проектирования МПС людьми могут быть внесены неисправности и приняты неверные проектные решения. Кроме того, в аппаратуре могут возникнуть дефекты.

7.7 Источники ошибок

Рассмотрим источники ошибок на первых трех этапах проектирования.

Этап 1. На этом этапе источниками ошибок могут быть: логическая несогласованность требований, упущения, неточности алгоритма.

Этап 2. На данном этапе источниками ошибок могут быть: упущения функций, несогласованность протокола взаимодействия аппаратуры и программ, неверный выбор микропроцессорных наборов, неточности алгоритмов, неверная интерпретация технических требований, упущение некоторых информационных потоков.

Этап 3. На этом этапе источниками ошибок могут быть: при разработке аппаратуры - упущения некоторых функций, неверная интерпретация технических требований, недоработка в схемах синхронизации, нарушение правил проектирования; при изготовлении прототипа - неисправности комплектующих изделий, неисправности монтажа и сборки; при разработке программных средств - упущения некоторых функций технического задания, неточности в алгоритмах, неточности кодирования.

Каждый из перечисленных источников ошибки может породить большое число субъективных или физических неисправностей, которые необходимо локализовать и устранить. Обнаружение ошибки и локализация неисправности являются сложной задачей по нескольким причинам: во-первых, из-за большого числа неисправностей; во-вторых, из-за того, что различные неисправности могут проявляться одинаковым образом. Так как отсутствуют модели субъективных неисправностей, указанная задача не формализована. Имеются определенные успехи в области создания методов и средств обнаружения ошибок и локализации физических неисправностей. Эти методы и средства широко используются для проверки работоспособного состояния и диагностики неисправностей дискретных систем при проектировании, производстве и эксплуатации последних.

Субъективные неисправности отличаются от физических тем, что после обнаружения, локализации и коррекции больше не возникают. Однако, как следует из перечня источников ошибок, субъективные неисправности могут быть внесены на этапе разработки спецификации системы, а это означает, что даже после самых тщательных испытаний системы на соответствие ее внешним спецификациям в системе могут находиться субъективные неисправности.

Процесс проектирования - итерационный процесс. Неисправности, обнаруженные на этапе приемосдаточных испытаний, могут привести к коррекции спецификаций, а следовательно, к началу проектирования всей системы. Обнаруживать неисправности необходимо как можно раньше, для этого надо контролировать корректность проекта на каждом этапе разработки.

7.8 Проверка правильности проекта

Основные методы контроля правильности проектирования следующие: верификация - формальные методы доказательства корректности проекта; моделирование; тестирование.

Существует много работ по верификации программного обеспечения, микропрограмм, аппаратуры. Однако эти работы носят теоретический характер. На практике пока используют моделирование поведения объекта и тестирование.

Для контроля корректности проекта на каждом этапе проектирования необходимо проводить моделирование на различных уровнях абстрактного представления системы и проверку правильности реализации заданной модели путем тестирования. На этапе формализации требований контроль корректности особо необходим, поскольку многие цели проектирования не формализуются или не могут быть формализованы в принципе. Функциональная спецификация может анализироваться коллективом экспертов или моделироваться и проверяться в опытном порядке для выявления, достигаются ли желаемые цели. После утверждения функциональной спецификации начинается разработка функциональных тестовых программ, предназначенных для установления правильности функционирования системы в соответствии с ее функциональной спецификацией. В идеальном случае разрабатываются тесты, целиком основанные на этой спецификации и дающие возможность проверки любой реализации системы, которая объявляется способной выполнять функции, оговоренные в спецификации. Этот способ - полная противоположность другим, где тесты строятся применительно к конкретным реализациям. Независимая от реализации функциональная проверка обычно заманчива лишь в теоретическом плане, но практического значения не имеет из-за высокой степени общности.

Автоматизация утомительной работы по составлению тестовых программ не только сокращает продолжительность периода конструирования/отладки за счет получения тестовых программ на этапе конструирования (поскольку они могут быть сгенерированы сразу после формирования требований к системе), но и позволяет проектировщику изменять спецификации, не заботясь о переписывании всех тестовых программ заново. Однако на практике разработке тестов часто присваивают более низкий приоритет по сравнению с проектом, поэтому тестовые программы появляются значительно позже его завершения. Но даже если детальные тесты оказываются подготовленными, часто практически нецелесообразно запускать их на имитаторе, так как детальное моделирование требует больших затрат средств на разработку программ и времени на вычисление, в результате большая часть работы по отладке должна откладываться до момента создания прототипа системы.

После обнаружения ошибки должен быть локализован ее источник, чтобы провести коррекцию на соответствующем уровне абстрактного представления системы и в соответствующем месте. Ложное определение источника ошибки или проведение коррекций на другом уровне абстрактного представления системы приводит к тому, что информация о системе на верхних уровнях становится ошибочной и не может быть использована для дальнейшей отладки при производстве и эксплуатации системы. Например, если неисправность внесена в исходный текст программы, написанной на языке ассемблера, а коррекция проведена в объектном коде, то дальнейшая отладка программы ведется в объектном коде; при этом все преимущества написания программы на языке ассемблера сводятся на нет.

7.9 Автономная отладка

Процесс отладки прототипа проектируемой системы должен начинаться с отладки аппаратуры и отладки программ.

Отладка аппаратуры предполагает тестирование отдельных устройств микропроцессорной системы - процессора, ОЗУ, контроллеров, блока питания, генератора тактовых импульсов путем подачи тестовых входных воздействий и приема ответных реакций. Тестовые входные воздействия и ответные реакции определяются, исходя из спецификаций на устройства, а также структурных схем устройств. При этом проверяются реальная аппаратура прототипа, спецификации, структурные схемы и отлаживаются тесты. После отладки отдельных устройств проверяется их взаимодействие. Процессор системы работает с шинами адресов, данных и управления. Анализируя их сигналы, можно проконтролировать выполнение программы в процессоре.

Поскольку ША и ШД синхронные, их работу лучше всего проверить с помощью методов логических состояний. Перед анализом последовательностей данных на этих шинах необходимо удостовериться в том, что сигналы, управляющие взаимодействием процессора с другими устройствами, выдаются в соответствующем порядке. Поскольку ШУ состоит из линий, работающих асинхронно, необходимо просматривать сигналы многих линий в течение одного и того же промежутка времени. Для анализа асинхронной работы линий управления необходимо также наблюдать за сигналами на них при возникновении определенного события, чтобы можно было четко разделить и идентифицировать различные состояния линий. Например, среди сигналов ШУ могут быть сигналы длительностью всего несколько наносекунд, но могут также возникать кратковременные ложные узкие импульсы, вызванные перекрестными помехами или шумами.

После того как доказана работоспособность ШУ, проводится дальнейшая проверка работы аппаратуры при различных режимах адресации процессора и кодах выбираемых данных. Для проверки выполнения процессором инструкций разрабатывается тестовая программа, которая помещается в ОЗУ или ППЗУ. При этом проверяется временная диаграмма сигналов и прохождения данных в системе (как осуществляется передача информации по отношению к строб-сигналам). Если тестовая программа - системный проверяющий тест пройдет успешно, можно утверждать, что автономно аппаратура отлажена.

При автономной отладке аппаратуры могут потребоваться приборы, умеющие: а) выполнять функции аналогового прибора, т. е. измерять напряжение и ток; воспроизводить форму сигнала, подавать импульсы определенной формы и т. д.; б) подавать последовательность сигналов одновременно на несколько входов в соответствии с заданной временной диаграммой или заданным алгоритмом функционирования аппаратуры, представленным в спецификации на языке высокого уровня, или другим способом; собирать значения сигналов многих линий в течение одного и того же промежутка времени, который определяется задаваемыми, программируемыми событиями - комбинацией или последовательностью сигналов на линиях, например, ложным сигналом на линии; обрабатывать и представлять собранную информацию либо в виде временной диаграммы, либо в виде диаграммы или таблицы логических состояний, либо на языке высокого уровня, например, языке регистровых передач.

Для автономной отладки аппаратуры широко используются осциллографы, вольтметры, амперметры, частотомеры, генераторы импульсов, позволяющие отлаживать аппаратуру на схемном уровне. Чтобы автономно отладить аппаратуру МПС на более высоком уровне, применяют логические анализаторы, генераторы слов, пульты, комплексы диагностирования.

7.10 Отладка программ

Отладка программ микропроцессорной системы проводится, как правило, на тех же ЭВМ, на которых велась разработка программ, и на том же языке программирования, на котором написаны отлаживаемые программы, и может быть начата на ЭВМ даже при отсутствии аппаратуры МПС. При этом в системном программном обеспечении ЭВМ должны находиться программы (интерпретаторы или эмуляторы), моделирующие функции отсутствующих аппаратных средств. Так, разработка и автономная отладка программных средств может вестись на больших ЭВМ, миниЭВМ, микроЭВМ, система команд которых не совпадает с системой команд используемого микропроцессора. Кроме того, при отладке программ может отсутствовать внешняя среда микропроцессорной системы, ее также необходимо моделировать.

Проверка корректности программ, т.е. проверка соответствия их внешним спецификациям, осуществляется тестированием. Программы проверяются на функционирование с различными исходными данными. Результаты функционирования программ сравниваются с эталонными значениями.

Отладка программ подразделяется на следующие этапы: планирование отладки; составление тестов и задания на отладку; исполнение программ; информирование о результатах исполнения программ по заданным исходным данным; анализ результатов, обнаружение ошибок и локализация неисправностей.

Существует два способа начального тестирования программ: пошаговый режим и трассировка программ.

В пошаговом режиме программа выполняется по одной команде за один раз, а пользователь анализирует содержимое памяти, регистров и т.д., чтобы проверить, соответствуют ли результаты ожидаемым. Пошаговый режим может быть трудоемким, если средства отладки будут требовать отдельных команд после каждого шага для того, чтобы показать необходимую информацию в понятном для пользователя виде. Имеются средства отладки, автоматически показывающие после каждого шага содержимое регистров процессора и ячеек памяти, используемых в последней команде, и несколько следующих команд. Пошаговый режим является весьма мощным средством предварительного тестирования, так как позволяет обнаруживать неисправности, прежде чем они существенно исказят программу и данные. Кроме того, неоднократно проходя отдельными шагами через один и тот же участок объектной программы, программист может легко изменять содержимое регистров и ячеек памяти, особенно если средства отладки имеют динамически обновляемый дисплей, и тем самым проверить работу программы в разных условиях. Этот интерактивный режим отладки программы позволяет разработчику постоянно упреждать, что будет делать его программа, и оперативно обнаруживать ошибку. Однако пошаговый режим с автоматическим показом результатов возможен только тогда, когда средства отладки содержат в своем составе дисплей с прямым доступом в память, так как после каждого шага на экране дисплея нужно показывать большой объем информации.

Исполнение программ осуществляется по шагам последовательно во времени и в соответствии с заданиями, содержащимися в операторах. При этом производится переработка значений переменных и определение оператора приемника. Если в ходе исполнения программы регистрируется последовательность операторов, реализуемых на каждом шаге процесса, то получается трасса или маршрут исполнения программы, который для конкретной программы зависит только от значений исходных данных.

Трассировка программ больше пригодна для отладочных средств, имеющих медленный, последовательный терминал. Программа-отладчик выполняет непрерывно команду за командой и выводит содержимое регистров процессора на терминал после каждого шага. Некоторые отладчики выводят также на терминал команды в дизассемблерной форме. Но при этом способе содержимое памяти не выводится на терминал и разработчик должен сам делать выводы об изменениях в ней. Отслеживание программы продолжается автоматически до тех пор, пока не будет остановлено извне. Результатом трассировки программы будут данные на экране дисплея или же в случае использования в качестве терминала печатающего устройства - длинная распечатка с ходом выполнения программы. Программист, анализируя

эти данные, может обнаружить ошибки. Трассировка программ не дает, однако, возможности изменять содержимое памяти и регистров и может послужить причиной того, что программа разрушит себя или свои данные прежде, чем отслеживание будет остановлено.

Отдельные участки программы после проверки, используя пошаговый режим или трассировку, можно объединить и проверить с помощью установки контрольных точек, вводимых в программу и прерывающих ее исполнение, для передачи управления программе-отладчику. По контрольным точкам можно по желанию выполнить избранные участки программы и проанализировать результаты. Контрольные точки устанавливаются обычно для конкретной команды, но в некоторых системах предусматриваются прерывания программы при чтении или записи данных в определенные ячейки памяти. Возможны и более сложные условия прерывания программы.

Расстановка контрольных точек предполагает, что программист связывает с ней точный адрес памяти. Для некоторых отладчиков программист задает абсолютный шестнадцатеричный адрес. Последние отладчики допускают символьные значения адресов, которые программист определяет в исходной программе; это позволяет значительно экономить время, распечатывая после каждого редактирования и транслирования программы новую копию листинга.

При тестировании можно планировать проверку всех возможных маршрутов исполнения программы для разных исходных переменных. Однако это реализуемо только для очень простых программ небольшого объема при малых диапазонах изменения исходных данных. Поэтому при планировании отладки программ применяют критерии полноты тестирования, которые, однако, не гарантируют полной проверки программ. Выбор критерия зависит от наличия ресурсов для тестирования и структурной сложности отлаживаемой программы. Критерии характеризуются глубиной контроля программ и объемом проверок.

В процессе отладки основная часть неисправностей в программах обнаруживается и затем устраняется. Однако всегда возможен пропуск нескольких неисправностей.

Средства отладки программ должны:

а) управлять исполнением программ (останавливать, изменять порядок, запускать и т. д.);

б) собирать информацию о ходе выполнения программы;

в) обеспечивать обмен информацией (диалог) между программистом и ЭВМ на уровне языка программирования;

г) моделировать работу отсутствующих аппаратных средств микропроцессорной системы.

7.11 Комплексная отладка микропроцессорных систем

Как правило, микропроцессорная система - это система реального времени, т. е. корректность ее функционирования зависит от времени выполнения отдельных программ и скорости работы аппаратуры. Поэтому система считается отлаженной после того, как рабочие программы правильно функционируют на действительной аппаратуре системы в реальных условиях. Дополнительным свойством, которым должны обладать средства комплексной отладки по сравнению со средствами автономной отладки, является возможность управления поведением МПС и сбора информации о ее поведении в реальном времени.

Тенденция развития средств отладки микропроцессорных систем состоит в объединении свойств нескольких приборов в одном комплексе, в создании универсальных средств, пригодных для автономной отладки аппаратуры, генерации и автономной отладки программ и комплексной отладки системы. Эти средства позволяют вести разработку и отладку, постепенно усложняя аппаратуру и программы. При этом разработка, изготовление и отладка планируются поэтапно с нарастанием сложности; новая, неотлаженная аппаратура и программа вводятся в создаваемую систему, присоединяются к проверенной ее части.

Если отладка программ ведется с использованием эмуляционного ОЗУ, а затем изготовляются микросхемы ПЗУ, то микропроцессорная система должна быть протестирована.

Средства отладки на последних этапах не должны влиять на правильность функционирования системы, вносить задержки, дополнительные нагрузки.

При комплексной отладке наряду с детерминированным используется статистическое тестирование, при котором МПС проверяется при изменении исходных переменных в соответствии со статистическими законами работы источников информации. Полнота контроля работоспособности проектируемой системы возрастает за счет расширения диапазона возможных сочетаний переменных и соответствующих им логических маршрутов обработки информации.

Существуют пять основных приемов комплексной отладки микропроцессорной системы: 1) останов функционирования системы при возникновении определенного события; 2) чтение (изменение) содержимого памяти или регистров системы; 3) пошаговое отслеживание поведения системы; 4) отслеживание поведения системы в реальном времени; 5) временное согласование программ.

Комплексная отладка завершается приемосдаточными испытаниями, показывающими соответствие спроектированной системы техническому заданию. Для проведения комплексной отладки МПС используют логические анализаторы и комплексы: оценочные, отладочные, развития микропроцессоров, диагностирования, средств отладки.

8. Отличия Intel 8086 (88) от современных микропроцессоров

Чипы 8086/88 завершали не только десятилетие, но были и последними моделями, воплощавшими определенную концепцию микроархитектуры. Несмотря на впечатляющие различия маргинальных моделей по вычислительным возможностям, они были достигнуты, в основном, благодаря экстенсивному развитию: увеличивались количество транзисторов, регистров, разрядность, тактовая частота. Последующее десятилетие ознаменовалось появлением концептуально новых моделей.

ПРОЦЕССОРЫ 80-х

Опираясь на архитектуру 8086 и учитывая запросы рынка, фирма Intel разработала два микропроцессора . 80186 и 80286. Отличающийся высокой степенью интеграции, микропроцессор 80186 был разработан для рынка встроенных управляющих систем.

В феврале 1982 г. Intel выпустила микропроцессор 80286 (286 в дальнейшем), где реализован ряд возможностей, позволявших рассматривать его как «настоящий процессор». Это был первый процессор, который мог выполнять программы своего предшественника. Впоследствии программная совместимость стала характерной чертой для семейства процессоров Intel.

Чип 286 был полностью 16-разрядным с 24-разрядной шиной адреса, содержал 134 тыс. транзисторов и работал на частотах вплоть до 12 MHz, что позволяло ему достичь производительности 2,66 MIPS. Процессор имел два режима работы: режим реального адреса (реальный режим) и защищенный режим виртуального адреса (защищенный режим). В первом режиме процессор вел себя, как более быстродействующий микропроцессор 8086. Во втором режиме он мог адресовать 16 MB физической памяти и до 1 GB виртуальной, и, что самое главное, защищенный режим позволял реализовать многозадачность. Этот процессор выпускался почти шесть лет и, по некоторым данным, был установлен примерно на 15 млн компьютеров. Именно этот чип использовался в IBM PC AT.

Микропроцессор 80286 предназначался для рынка персональных компьютеров и рабочих станций, где требовались программная совместимость и высокая производительность. В высокопроизводительном микропроцессоре 80286 впервые были микросхемно реализованы такие современные возможности, как многозадачность и управление виртуальной памятью.

Для него были разработаны такие периферийные компоненты, как математический сопроцессор lntel287, микросхемы ввода-вывода, контроллер DMA и другие периферийные устройства системы.

В октябре 1985 г. появился процессор 80386, представляющий собой значительный шаг вперед по сравнению со своим предшественником. Прежде всего, это был первый полностью 32-разрядный процессор для ПК, который мог «перемалывать» вдвое больше данных за один такт, чем 286. Для своего времени он являлся единственным 32-разрядным микропроцессором, совместимым снизу вверх (начиная от 8086). Это сохраняло актуальность существующего прикладного ПО, стоимость которого достигала 6,5 млрд. долл. Процессор содержал 275 тыс. транзисторов, работал на частотах до 33 MHz, что позволяло повысить производительность до 11,4 MIPS. Его специфическими особенностями являлись истинная многозадачность, встроенное управление памятью, виртуальная память с разделением на страницы, защита программ и большое адресное пространство (4 GB адресуемой памяти и 64 TB виртуальной). Аппаратная совместимость с предыдущими моделями сохранялась посредством динамического изменения разрядности магистрали.

Периферийные компоненты для Intel386 оптимизированы на достижение высокой производительности 32-битных систем и включают в себя такие основные 32-битные периферийные устройства, как контроллер DMA 82380, контроллеры кэш-памяти 82385, 82395 и математический сопроцессор Intel387.

Каждое новое поколение архитектуры, развиваемое фирмой Intel, обеспечивает значительный рост производительности. В 1988 г, фирма Intel предложила микропроцессор Intel386SX, представляющий собой удешевленный вариант популярного ЦП Intel386 с 16-битной шиной данных. Он выполняет все 32-битные программы, написанные для ЦП Intel386. К концу 1988 г. микропроцессор Intel386SX был хорошо принят широким кругом пользователей.

В 1989 г. фирма Intel представила микропроцессор Intel386DX. Он объединяет 1,2 миллиона транзисторов, имеет производительность, вдвое превышающую производительность ЦП Intel386, и обладает 100%-ной совместимостью с прежними изделиями семейства Intel386.

В 1990 г. фирмой Intel был представлен первый член семейства архитектуры Intel386 с высокой степенью интеграции . микропроцессор Intel386SL При объединении ЦП Intel386SL с однокристальной микросхемой 82360SL, удовлетворяющей стандарту ISA, обеспечивается малое потребление энергии и габариты, требуемые в портативных персональных компьютерах. Вместе с тем обеспечивается полная совместимость с программным обеспечением предшествующего поколения.

Процессор Intel486SX был представлен в октябре 1989 г. Этот процессор обеспечивает переход на технологию Intel486 для малых учрежденческих систем. В 1991 г. фирма Intel также представила микропроцессор Intel486DX с рабочей частотой 50 МГц, повысивший производительность семейства Intel486DX на 50 процентов. Он содержал 1,2 млн транзисторов, базовая модель показывала

производительность 20 MIPS. Итак, это был первый процессор со встроенными математическим сопроцессором и кэш-памятью первого уровня. Для его старших моделей (Intel486DX2, 1992 г.) впервые была применена технология умножения частоты, позволявшая работать на частотах, кратных частоте системной шины, — чип разгонялся до 120 MHz. Микроархитектура чипа впервые использовала концепции RISC-архитектуры и реализовала полноценный пятистадийный конвейер (выборка инструкции, декодирование, формирование адреса, выполнение и запись в память результата), позволявший выполнять одну инструкцию за такт. Таким образом, это был первый скалярный процессор в семействе x86. Кроме этого, генерация 486-х знаменовала изменение парадигмы пользовательского интерфейса - переход от командной строки к технологии «указать-и-выбрать» (point-and-click).

Хотя последняя модель 486DX4 вышла в марте 1994 г., можно сказать, что этот микропроцессор завершил 80-е, так как символом нового десятилетия стало следующее поколение процессоров Intel.

ПРОЦЕССОРЫ 90-х

Поскольку Intel не могла на законных основаниях защитить имя «586» от использования его другими компаниями, она решила отказаться от шаблона 80x86 для своих последующих моделей. Ее следующий чип вместо «лагерного» номера 586 получил звучное имя Pentium, которое, тем не менее, явно указывало на пятое поколение. Для того чтобы закрепить свою исключительность (а может, и по технологическим соображениям), для нового процессора был разработан и новый интерфейс с материнской платой — Socket 7. Кроме этого, Intel унифицировала дизайн семейства чипсетов, куда вошли Neptun, Triton FX, HX, VX и TX (последний - для Socket 7).

Первый процессор Pentium 60 (число указывает тактовую частоту в MHz) был анонсирован в марте 1993 г. Он имел 32-разрядную архитектуру, однако разрядность шины памяти была увеличена до 64. На кристалле располагалось 3,1 млн транзисторов, и при частоте 60 MHz он показывал производительность 100 MIPS. Такая производительность достигалась за счет усовершенствованной микроархитектуры процессора. Два пятистадийных конвейера для операций над целыми числами, позволявшими выполнять две независимые инструкции за такт, и восьмистадийный конвейер для вещественной арифметики почти удваивали его вычислительные возможности по сравнению с чипом 486 такой же частоты. Увеличению производительности способствовало также использование небольшого кэша, известного как Branch Target Buffer (BTB), с помощью которого реализовывался механизм динамического предсказания ветвлений. Когда по ходу исполнения программы встречалось ветвление (условный переход), в буфере запоминались инструкция и адрес перехода. Эта информация использовалась для предсказания перехода при повторном выполнении данной инструкции. Механизм предсказания ветвлений был чрезвычайно эффективен, если программа содержала большое количество циклов. Кэш-память первого уровня разделялась на два буфера по 8 KB каждый - отдельно для данных и инструкций (так называемая гарвардская архитектура). Кэш данных имел два интерфейса — по одному на каждый конвейер, что позволяло за один такт поставлять данные для двух отдельных операций. Это был суперскалярный (более одной инструкции за такт) процессор, программно-совместимый с предыдущими моделями семейства x8б.

Первые процессоры Pentium были очень «горячими», и единственный выход при 0,35 мкм конструкторских нормах состоял в том, чтобы понизить напряжение питания ядра, что и было сделано в следующих моделях. Максимальная частота процессора Pentium составляет 200 MHz.

Pentium Pro, представленный в ноябре 1995 г., был процессором шестого поколения. Он состоял из двух чипов: непосредственно процессора и кэш-памяти второго уровня, объем которой в старших моделях (Pentium Pro 200, август 1997 г.) достигал 1 MB. Объединение на одном кристалле процессора и кэш-памяти позволяло работать с ней на частоте процессора. Это увеличивало производительность, однако стоимость такой пластины была довольно высокой, и этот процессор не стал массовым. Правда, Intel и нацеливала его на рынок серверов и высокопроизводительных рабочих станций.

Кроме этой новинки, модификации подверглась и микроархитектура процессора. Прежде всего, он стал суперконвейерным - количество стадий конвейера для целочисленных операций увеличилось с 5 до 14. Это стало возможным благодаря тому, что длинные CISC-инструкции перед выполнением транслировались в несколько коротких RISC-подобных инструкций. Далее в процессоре был реализован механизм выполнения инструкций с нарушением очередности их следования (out-of-order), или так называемое спекулятивное выполнение (speculative execution). Напомним, что традиционные процессоры в случае, когда выполняемая инструкция потребовала данные, которых не оказалось в кэш-памяти, или, чего доброго, они вообще находятся на диске, «тупо» ждут, когда данные будут им доставлены (механизм обработки потока инструкций). Вместо этого процессор Pentium Pro просматривает вплоть до 20 инструкций вперед и выполняет их в зависимости от их готовности, а не от порядка следования в программе. Результаты выполненных инструкций хранятся в специальном буфере и подаются на выход только тогда, когда подходит их очередь, предусмотренная программой (механизм обработки потока данных). Были расширены также возможности блока предсказания ветвлений, который мог теперь предсказывать множественные ветвления.

Комбинация улучшенного механизма предсказания ветвлений, анализа потока данных и спекулятивного выполнения инструкций, которая была названа Intel «динамическим выполнением» (Dynamic Execution), позволила почти вдвое увеличить производительность Pentium Pro по сравнению с предыдущей моделью.

Процессор Pentium Pro из-за высокой стоимости не смог вытеснить своих предшественников с этого сектора рынка. К тому же, и конкуренты не сидели сложа руки. Поэтому нет ничего удивительного в том, что Intel продолжала развивать линию процессоров Pentium. В январе 1997 г. ею был предложен публике процессор Pentium с технологией MMX, предназначенной для повышения скорости выполнения мультимедиа-приложений. Эта идея не блистала новизной - нечто подобное уже было реализовано компаниями Sun (SPARC VIA) и Hewlett-Packard (HP-PA MAX), что первоначальное название этой технологии MultiMedia extension было изменено Intel на MatrixMath extension.

Несмотря на нежелание широкой публики бросить все и бежать покупать процессоры Pentium Pro, Intel не собиралась лишать массы передовой технологии. С этой целью Pentium Pro разделили на две части: процессор — отдельно, кэш — отдельно, соединив, правда, их автономной шиной, работающей на половинной частоте процессора (вся шинная архитектура получила название Dual Independent Bus - двойная независимая шина). Добавив к одной из половинок технологию MMX и упаковав все в процессорный модуль (так называемый Single Edge Contact Cartridge - S.E.C), который требовал разработанного и запатентованного разъема (Slot 1), корпорация представила в мае 1997 г. новый процессор с почти королевским именем — Pentium II (Pentium Второй). Процессор содержит 7,5 млн транзисторов и работает на частотах от 233 до 450 MHz, при этом, начиная с частоты 333 MHz, выполняется по 0,25 мкм технологии.

А будущее микропроцессорной техники связано сегодня с двумя новыми направлениями - нанотехнологиями и квантовыми вычислительными системами. Эти пока еще главным образом теоретические исследования касаются использования в качестве компонентов логических схем молекул и даже субатомных частиц: основой для вычислений должны служить не электрические цепи, как сейчас, а положение отдельных атомов или направление вращения электронов. Если "микроскопические" компьютеры будут созданы, то они обойдут современные машины по многим параметрам.

Рекомендуемая литература

1. Балашов Е. П., Григорьев В. Л., Петров Г. А. Микро- и миниЭВМ. Л.: Энергоатомиздат, 1984. 376 с.

2. Микропроцессоры: В 3-х кн. / Под ред. Преснухина. М.: Высшая шко-ла, 1986. Кн.1. 495 с. Кн. 2. 383 с. Кн. 3. 351 с.

3. Токхайм Р. Микропроцессоры: Курс и упражнения / Пер. с англ. Под ред. Грасевича. М.: Энергоатомиздат, 1987. 338 с.

4. Майоров С. А., Кириллов В. В., Приблуда А. А. Введение в микроЭВМ. Л.: Машиностроение, Ленингр. отд., 1988. 303 с.

5. Морисита И. Аппаратные средства микроЭВМ / Пер. с япон. М.: Мир, 1988. 279 с.

6. Соучек Б. Микропроцессоры и микроЭВМ / Пер. с англ. Под ред. А. И. Петренко. М.: Сов. радио, 1979. 517 с.

7. Гибсон Г., Лю Ю.-Ч. Аппаратные и программные средства микроЭВМ / Пер. с англ. В. Л. Григорьева, Под ред. В. В. Сташина. М.: Финансы и статистика, 1983. 255 с.

8. Гивоне Д., Россер Р. Микропроцессоры и микрокомпьютеры: Вводный курс / Пер. с англ. М.: Мир, 1983. 463 с.

1. Нестеров П. В. Микропроцессоры. Архитектура и ее оценка. М.: Высшая школа, 1984. 104 с.

2. Уокерли Дж. Архитектура и программирование микроЭВМ: В 2-х кн. / Пер. с англ. М.: Мир, 1984. Кн. 1. 486 с. Кн. 2. 359 с.

3. Хосе М. Ангуло. Микропроцессоры: Архитектура, программирование и проектирование систем. Тбилиси: Ганатлеба, 1989.

4. МикроЭВМ / Пер. с англ., Под ред. А. Дирксена. М.: Энергоиздат, 1982. 328 с.

5. Хоровиц П., Хилл У. Искусство схемотехники. М.: Мир,1983. Т. 2. 590 с.

6. Вуд А. Микропроцессоры в вопросах и ответах / Пер. с англ. М.: Энергоатомиздат, 1985. 185 с.

7. Басманов А. С., Широков Ю. Ф. Микропроцессоры и однокристаль-ные микроЭВМ: Номенклатура и функциональные возможности / Под ред. В. Г. Домрачева. М.: Энергоатомиздат, 1988. 127 с.

8. Мячев А. А., Иванов В. В. Интерфейсы вычислительных систем на базе мини- и микроЭВМ / Под ред. Наумова Б. Н. М.: Радио и связь, 1986. 248 с.

9. Интерфейсы систем обработки данных: Справочник / Под ред. А. А. Мячева. М.: Радио и связь, 1989.

10. Байцер Б. Архитектура вычислительных комплексов. М.: Мир, 1974. Т.1,2.

11. Водовозов В. М., Осипов В. О., Пожидаев А. К. Практическое введе-ние в информационные системы / ГЭТУ. СПб, 1995.

Содержание

Введение

1. Классификация микропроцессоров

2. Архитектура микропроцессоров
2.1 Основные характеристики
2.2 Структура типового микропроцессора
2.3 Логическая структура
2.4 Устройство управления
2.5 Особенности программного и микропрограммного управления операциями
2.6 Система команд
2.7 Форматы команд микропроцессора К1810ВМ86
2.8 Режимы адресации
2.9 Типы архитектур

3. Организация ввода/вывода в микропроцессорной системе
3.1 Программная модель внешнего устройства
3.2 Форматы передачи данных
3.3 Параллельная передача данных
3.4 Последовательная передача данных
3.4.1 Синхронный последовательный интерфейс
3.4.2 Асинхронный последовательный интерфейс
3.5 Способы обмена информацией
3.5.1 Программно-управляемый ввод/вывод
3.5.2 Организация прерываний в микроЭВМ
3.5.3 Организация прямого доступа к памяти

4. Память микропроцессорной системы
4.1 Микросхемы памяти в составе микропроцессорной системы
4.2 Буферная память
4.3 Стековая память

5. Микропроцессор Intel 8086(88)
5.1 Поставляемая разработчиком информация
5.2 Схема и назначение выводов
5.3 Архитектура микропроцессора
5.4 Использование регистра адреса/данных
5.5 Использование указателя стека

6. Программирование микропроцессора
6.1 Машинный код и ассемблер
6.2 Простой состав команд
6.3 Состав команд арифметических операций
6.4 Состав команд логических операций
6.5 Состав команд операций передачи данных
6.6 Состав команд операций ветвления
6.7 Состав команд вызова подпрограмм
6.8 Состав команд прочих операций

7. Проектирование МПС
7.1 Уровни представления микропроцессорной системы
7.2 Ошибки, неисправности, дефекты
7.3 Отладка
7.4 Обнаружение ошибки и диагностика неисправности
7.5 Функции средств отладки
7.6 Этапы проектирования микропроцессорных систем
7.7 Источники ошибок
7.8 Проверка правильности проекта
7.9 Автономная отладка микропроцессорных систем
7.10 Отладка программ
7.11 Комплексная отладка микропроцессорных систем

8. Отличия Intel 8086 (88) от современных микропроцессоров