С.Б. Спиридонов

ВЫЧИСЛИТЕЛЬНЫЕ СРЕДСТВА АВТОМАТИЗИРОВАННЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

Краткий курс лекций

Учебно-методическое пособие

Часть 1

Москва 2024

УДК 621.3.038(075.8)
ББК 32.97
С 72

Автор:
доцент кафедры «Системы обработки информации
и управления» МГТУ им. Н.Э. Баумана
Спиридонов Сергей Борисович

Рецензент:
д.т.н., профессор кафедры «Компьютерные системы
Автоматизации производства»
МГТУ им. Н.Э. Баумана
Мурашов Михаил Владимирович

Спиридонов С.Б.

С 72 Вычислительные средства автоматизированных систем обработки информации и управления. Краткий курс лекций: Учебно-методическое пособие. Часть 1. – М.: Издательство «Спутник +», 2024. – 115 с.

ISBN 978-5-9973-6857-9

Рассмотрены основные типовые архитектуры вычислительных устройств и систем, входящих в АСОИУ. Изложены классические методы проектных решений при проектировании типовых, базовых архитектур вычислительных средств. Рассмотрены архитектуры персональных компьютеров и наиболее распространённые периферийные устройства, и их интерфейсы.

Для студентов вузов, обучающихся по направлению подготовки «Информатика и вычислительная техника» специальности «Автоматизированные системы обработки информации и управления».

УДК 621.3.038(075.8)
ББК 32.97

Отпечатано с готового оригинал-макета.

ISBN 978-5-9973-6857-9

Содержание

Предисловие

Бурное развитие интегрированных схемотехнических решений во всех областях использования вычислительной техники и системах управления, требует наличия у студентов базовых знаний основ проектирования элементов и узлов вычислительных средств. Большинство структурных и схемотехнических решений, применяемых в компьютерах, микропроцессорных системах, микроконтроллерах и т.д. являются универсальными и типовыми.

Вычислительные средства АСОИУ – учебная дисциплина, в которой рассматривают архитектуру широкого класса компьютерных устройств, лежащих в основе проектных и конструкторских решении при создании средств вычислительной техники.

Учебно-методическое пособие разработано на основе лекций по дисциплине «Вычислительные средства АСОИУ», читаемых автором на кафедре «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана.

Опыт преподавания студентам основ проектирования показал, что теоретический материал прочно и быстро усваивается студентами только при практическом выполнении задач. Часть этой проблемы решается в ходе лабораторного практикума и при практическом выполнении курсовой работы.

Учебно-методическое пособие предназначено для студентов вузов, обучающихся по направлению подготовки 230100 «Информатика и вычислительная техника» специальности «Автоматизированные системы обработки информации и управления». Отдельные главы учебного пособия будут полезны слушателям второго высшего образования и кругу специалистов, занимающихся проектированием специализированных вычислительных средств автоматики и управления.

Модуль 1. Теоретические основы проектирования и построения ВС АСОИУ

Лекция 1

Архитектура вычислительных средств АСОИУ

В настоящий период архитектуру автоматизированных средств обработки информации и управления (АСОИУ) нельзя представить без повсеместного внедрения в её разнообразное и многочисленное по функциональному назначению оборудованию без встроенных средств вычислительной техники.

В АСОИУ представлено использование подавляющего большинства ЭВМ самой различной архитектуры и назначения. Из этого не трудно сделать вывод о необходимости рассмотрения ЭВМ как основного ядра АСОИУ и посвятить изучению её архитектуры и особенностям работы материалы данного лекционного курса.

Развитие и совершенствование ЭВМ связано с достижениями во многих областях, но в основном в:

- технологии производства, совершенствовании полупроводниковой элементной базы;
- успехи в совершенствовании архитектур ЭВМ;
- успехи в разработке операционных систем и прикладного программного обеспечения.

Под термином «вычислительные средства» будем понимать все разновидности архитектур ЭВМ, в том числе и микроконтроллеры, которые так или иначе применимы в спектре оборудования АСОИУ.

Любая ЭВМ может рассматриваться как совокупность технических средств, служащих для автоматизированной обработки дискретных данных по заданному алгоритму.

Любую ЭВМ можно рассматривать на различных уровнях взаимодействия её аппаратно-программных средств. Причем уровни рассмотрения этого взаимодействия могут быть различными:

- низший уровень: уровень взаимодействия электрических сигналов между элементами и узлами ЭВМ;

- высший уровень: взаимодействие узлов ЭВМ на уровне программных модулей;

- функциональный уровень каждого отдельного узла: функция и их реализация аппаратными - программными средствами.

Общая укрупнённая архитектура ЭВМ

Любая ЭВМ может быть сведена к следующей функциональной схеме рис. 1.1.

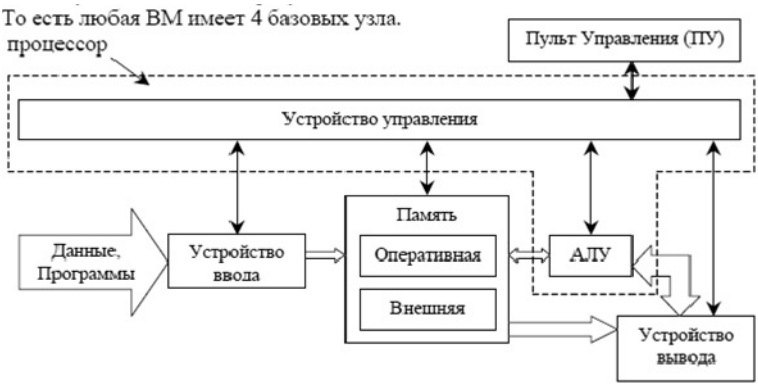


Рис.1.1. Состав ЭВМ из 4 базовых узлов.

В основе функционирования любой ВМ лежат два фундаментальных понятия в вычислительной технике:

1. понятие алгоритма.

2. принцип программного управления.

Алгоритм – одно из фундаментальных понятий математики и вычислительной техники. Международная организация стандартов (ISO) формулирует понятие алгоритм как «конечный упорядоченный набор чётко определённых правил для решения проблемы» (ISO 2382/1-93).

Основным свойствами алгоритма являются: дискретность, определённость, массовость и результативность.

Алгоритм – некоторая однозначно определенная последовательность действий, состоящая из формально заданных операций над исходными данными, приводящая к решению за конечное число шагов.

Свойства алгоритмов:

1. дискретность алгоритма (действия выполняются по шагам, а сама информация дискретна);
2. детерминированность (сколько бы раз один и тот же алгоритм не реализовывался для одних и тех же данных результат один и тот же);
3. массовость (алгоритм “решает задачу” для различных исходных данных из допустимого множества и дает всегда правильный результат)

Программа – описание алгоритма на каком-либо языке.

Принцип программного управления (ППУ) впервые был сформулирован Венгерским математиком и физиком Джоном фон Нейманом в 1946 году.

ППУ включает в себя несколько архитектурно – функциональных принципов.

1. Любой алгоритм представляется в виде некоторой последовательности управляющих слов – команд. Каждая отдельная команда определяет простой (единичный) шаг преобразования информации.
2. Принцип условного перехода. В процессе вычислений в зависимости от полученных промежуточных результатов возможен автоматический переход на тот или иной участок программы.
3. Принцип хранимой программы. Команды в ЭВМ представляются в такой же кодируемой форме, как и любые данные и хранятся в таком оперативном запоминающем устройстве (ОЗУ). Это значит, что если рассматривать содержимое памяти, то без какой-то команды невозможно различить данные и команды. Следовательно, любые команды можно принципиально обрабатывать как данные (информация в ЭВМ отличается не представлением, а способом ее использования).
4. Принцип двоичного кодирования.
5. Принцип иерархии запоминающих устройств (ЗУ).

В трудах фон Неймана определены основные устройства ЭВМ, с помощью которых возможно реализовать перечисленные выше принципы.

На рис. 1.2. изображены основные устройства такой ЭВМ.

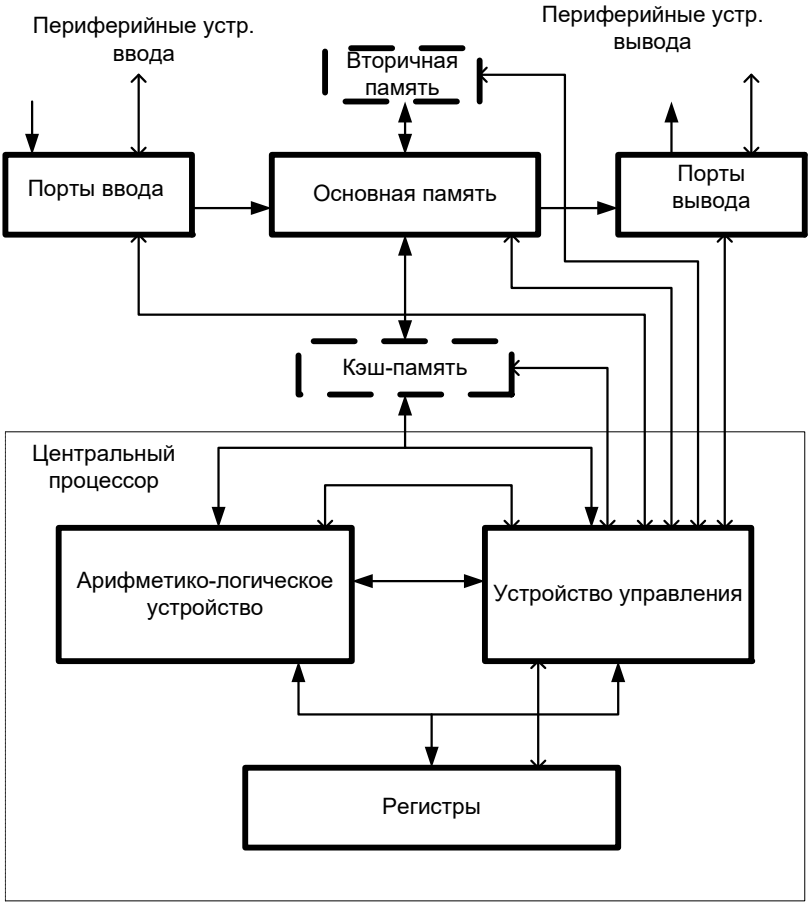


Рис. 1.2. Структура фон-Неймановской вычислительной машины, обведённые пунктиром узлы появились в структуре ЭВМ значительно позже.

Типичная фон-Неймановская ЭВМ содержит: память, устройство управления, арифметико-логическое устройство и устройство ввода-вывода.

Информация в ЭВМ поступает из подсоединённых периферийных устройств. Связь обеспечивают порты ввода и порты вывода.

Память представляет сложную многоуровневую структуру, реализованную в виде взаимодействующих запоминающих устройств, имеющие разные принципы хранения информации. Неотъемлемой частью современных ЭВМ стала кэш-память – память небольшой ёмкости, но высокого быстродействия. В неё копируются из основной памяти часто используемые команды и данные. Процессор выбирает команды и данные, находящиеся в быстрой кэш-памяти. Еще одной быстродействующей памятью являются регистры процессора для кратковременного хранения промежуточных результатов операций процессора.

Устройство управления (УУ) – важнейшая часть ЭВМ организующее автоматическое выполнение программ и обеспечивающая функционирование ЭВМ как единой системы. Управление вычислительным процессом сводится к выдаче нужного набора сигналов управления. Основной функцией УУ является формирование управляющих сигналов, отвечающих за извлечение команд и данных из памяти в порядке, определяемой программой.

АЛУ обеспечивает арифметическую и логическую обработку двух входных операндов и получение результата.

УУ и АЛУ тесно взаимосвязаны и их обычно рассматривают как единое устройство, известное под название центральный процессор.

Структуры ЭВМ

В настоящее время одинаковое распространение получили два типа структур:

- с непосредственными связями;
- на основе шины.

Представителем ЭВМ с непосредственными связями может служить фон-Неймановская ЭВМ. Вид связей определяет разрядность слов, скорость обмена и т.д. Узким местом у фон-Неймановских машин является канал связи между процессором и памятью.

Структура на основе общей шины предусматривает, что все устройства ЭВМ подключены к магистральной шине, служащей единственным трактом для потоков команд, данных и управления рис. 1.3.

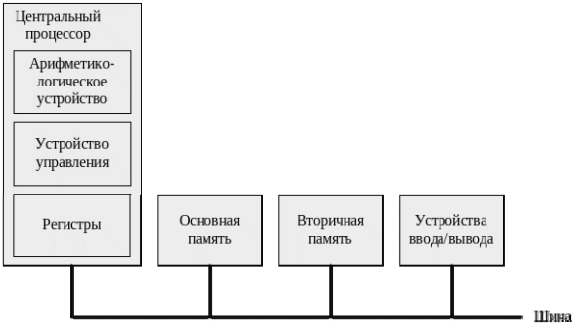


Рис. 1.3. Структура ЭВМ на базе общей шины.

Наличие общей шины существенно упрощает реализацию ЭВМ, позволяет легко менять состав и конфигурацию машины. Шинная архитектура получила широкое распространение в мини – и микро-ЭВМ. Основную нагрузку на шину создают обмены между процессором и памятью. Шинная архитектура в чистом виде оказывается недостаточно эффективной. Более распространённой стала архитектура с иерархией шин, в которой помимо магистральной имеется несколько дополнительных шин. Это приводит к снижению нагрузки на магистральную шину более эффективному использованию её пропускной способности.

Структуры вычислительных систем

Понятие «вычислительная система» предполагает наличие множества процессоров.

Первая структура – структура вычислительной системы с общей памятью. Роль связи с памятью осуществляет коммуникационная сеть, которая чаще реализуется общей магистральной шиной. Помимо недостатков, свойственных общей шине, имеется и достоинство: обмен информацией между процессорами не связан с дополнительными операциями и обеспечивается за счёт доступа к общим областям памяти.

Вторая структура – распределённая система, где общая память вообще отсутствует, а у каждого процессора есть своя локальная память. Часто такие системы объединяют отдельные ЭВМ. Обмен осуществляет коммуникационная сеть путём обмена сообщениями. Подобное построение ВС снимает ограничения, свойственные для общей шины, но приводит к дополнительным издержкам на пересылку между процессорами.

Исторические этапы развития средств вычислительной техники

В истории развития вычислительных средств исторически определились два этапа: этап механических калькуляторов и этап электрических и электронных вычислительных машин.

Механические калькуляторы в основе своей архитектуры имели взаимодействие зубчатых колес друг с другом. В разных странах были изобретены многочисленные механические вычислительные машины, использовавшие десятичную систему счисления. Механические калькуляторы позволяли выполнять сложение, вычитание, умножение и деление. Такой калькулятор сконструировал Ч. Томас в 1820 году.

В 1823 году профессор математики Кембриджского университета Ч. Бебидж построил первую механическую вычислительную машину. Из-за отсталой технологичности механической обработки нужных деталей и блоков эта машина до конца не была создана. Однако, она имела все основные функциональные блоки, из которых состоят сегодняшние типовые ЭВМ: вычислительное устройство (нынешний процессор), запоминающее устройство, устройство ввода данных с перфокарт, устройство вывода на печать результатов, блок управления, состоящий из множества пластин и штифтов.

Последним, самым усовершенствованным механическим калькулятором был арифмометр Однера, выпускавшийся до 70-х годов 20 века.

В теоретическом плане большой вклад в теорию вычислительной техники внёс венгерский математик Джон фон Нейман. Он опубликовал известные

тезисы, которые легли в основу проектных решений структуры и архитектуры ЭВМ не одного поколений.

В эпоху механической эры стоит отметить ещё несколько ключевых достижений:

- высказано предположение о возможном использовании двоичной системы счисления;
- аппарат булевой алгебры стали применять при описаниях и расчётах переключательных электрических схем,
- была изобретена схема катодного реле, которая легла впоследствии в основу схемы триггера, заменившего ненадёжные реле. В России триггер изобрел Бонч-Бруевич М.А.

Этап электронно-вычислительных машин претерпел деление на несколько поколений вычислительных устройств, отличавшихся элементной базой, функционально-логической организацией, конструктивно-технологическим исполнением, программным обеспечением и рядом других важных характеристик.

История ВТ отсчитывается с опубликования работы Джона фон Неймана. Впервые возможность построения цифровой ВМ была доказана английским математиком Тьюрингом в 1936 году. Он показал, что любой алгоритм реализуется с помощью его дискретного автомата, который был назван машиной Тьюринга. Независимо это же доказал Пост (машина Поста).

Физически первая цифровая ВМ была сконструирована в 1935 году фирмой Белл (США).

Такого же вида машина была сконструирована для специальных задач под руководством К. Цузе (1941, Германия). Попытка построения универсальной ЭВМ была предпринята Айкеном (США). Она получила название “Марк-1”. Спроектирована и изготовлена в Гарвардском университете.

Первая ЭВМ, разработанная на электронных компонентах, изготовлена в 1942 году (“Эниак”). Серийный выпуск в 1945-1946 годах. Разработана в Пенсельванском университете под руководством Мочли и Эккера. В 1943 году

под руководством Тьюринга была разработана ЭВМ “Колосус”. После рассекречивания архивов в 70-х годах оказалось, что первая ЭВМ была разработана в 1939 году выходцем из Германии Атонасовым, которая получила название “АВС”.

Первое поколение ЭВМ. (1945-1955 годы)

ЭВМ первого поколения отличала элементная база: электронные лампы, дискретные резисторы, конденсаторы, трансформаторы, ферритовые сердечники в структуре оперативной памяти. Были механические устройства для работы с перфолентами и перфокартами. Для ЭВМ первого поколения характерны большие размеры, большая потребляемая мощность, низкая надёжность, невысокое быстродействие.

В истории отечественного первого этапа развития ЭВМ можно выделить следующие достижения.

В нашей стране началом выпуска можно считать начало 50-х годов, создана ЭВМ “МЭСМ”. Разработана под руководством Лебедева. В 1952-1953 годах на этой основе, под руководством Мельникова и Бурцева была разработана “БЭСМ-1” (Большая электронная счетная машина).

А на ее основе был произведен серийный выпуск машины “БЭСМ-2”. В это же время в США выпускают машину “Эдвак”. Технические характеристики машины “БЭСМ-2” были гораздо выше. Это было связано с тем, что в “БЭСМ-2”, использовались два совершенно новых принципа: конвейеризации и стека. Для “БЭСМ-2”, быстродействие АЛУ составляло порядка 10000 операций в секунду.

В 1953 году была разработана машина “Стрела” под руководством Василевского. А так же в Московском Энергетическом институте под руководством академика Брука были разработаны ЭВМ получившие название “М”. В Минске был создан завод по производству ЭВМ, серийное производство машин “Минск”. В городе Пензе было создано ОКБ (отдел конструкторского бюро) под руководством академика Рамеева, где разработали и выпускали серийно ЭВМ “Урал”.

Структура ЭВМ первого поколения полностью соответствовали машине фон Неймана. Технические характеристики машин были значительно ниже характеристик современных ПК. Программирование велось в машинных кодах. Емкость ОЗУ – 2 тысячи слов. Ввод информации с перфоленты и кинопленки.

Второе поколение ЭВМ. (1955-1965 годы)

ЭВМ второго поколения отличались:

- использованием полупроводниковых дискретных транзисторов в элементном решении триггеров и других узлов на их основе;
- появление дисплеев с возможностью отображения информации по пиксельно;
- использование шин для организации взаимодействия основных устройств ЭВМ;
- многопрограммный принцип работы;
- появление алгоритмических языков в программном обеспечении.

У ЭВМ второго поколения уменьшились размеры, потребляемая мощность, уменьшилась стоимость, повысилась надёжность и быстродействие, вырос объём памяти.

В схемотехнике ЭВМ транзисторы позволяли обеспечить большую надежность, быстродействие и меньшее энергопотребление (среднее время отказа около 100 часов, тогда как на машинах первого поколение около 10 часов, энергоемкость на два порядка ниже, по сравнению с машинами первого поколения). Переход к печатному монтажу также улучшило надежность.

Начинается бурное развитие математического и программного обеспечения. Создание алгоритмических языков (Fortran, ALGOL). Создаются простейшие компиляторы и интерпретаторы. Становится нецелесообразна работа пользователя у пульта управления.

Основным режимом становится работа через операторов. Появляются многопрограммные ЭВМ. Многопрограммность достигается за счет

параллельной программной обработки. Для работы в пакетном режиме создаются первые мониторы и supervisor'ы. Вследствие чего происходит резкое увеличение использование ЭВМ второго поколения.

Третье поколение ЭВМ (1965-1980 годы)

Самой характерной особенностью третьего поколения ЭВМ является широкое применение интегральных схем (ИС) и многослойного печатного монтажа на платах типовых элементов замены (ТЭЗах). ИС является сложной транзисторной схемой, реализованной на кристалле в едином корпусе.

Помимо этого, третье поколение ЭВМ отличает:

- появление микропроцессора и микросхем памяти;
- применение магнитного диска, как главного носителя информации;
- рост количества внешних устройств ввода-вывода;
- возможность удалённой работы пользователей;
- многопользовательский режим работы;
- дальнейшее совершенствование операционных систем;
- унификация узлов и устройств, для организации архитектуры ЭВМ по требованиям потребителей;
- создание мини и микро-ЭВМ.

В конце 60-х годов появляются первые машины третьего поколения. Переход к третьему поколению ЭВМ связывают с серьезными архитектурными изменениями. Изменение технической базы связано с переходом на интегральную схемотехнику. Правда степень интеграции была небольшой. Вследствие чего произошло заметное увеличение надежности.

В машинах третьего поколения формируется концепция канала, начинается работа с распараллеливанием процессора, появляется микропрограммное управление, память становится иерархической по структуре, впервые вводится понятие агрегатирования.

Рис.1. 4. Функциональная структура ЭВМ третьего поколения.

Канал является основным структурным элементом. В структуре процессора и оперативной памяти появляются специальные устройства, которые организуют адресные механизмы (обеспечивающие адресацию, перемещение программы в памяти, взаимную защиту). В процессоре появляется несколько АЛУ (целочисленные, с плавающей арифметикой, для работы с адресами). Правда, эти устройства параллельно не работают, но для выполнения той или иной обработки выбирается определенное АЛУ. В памяти четко выделяется основная память, к которой процессор обращается непосредственно, и массовая память, емкость которой значительно больше емкости основной памяти, но непосредственно процессору она недоступна. Тем более данные с внешних устройств непосредственно недоступны процессору. Так как память иерархична, то создаются механизмы для управления памятью. Развивается и внутренняя память процессора (создаются предпосылки кэширования). В конце третьего поколения ЭВМ появляется концепция управления виртуальной памяти, развиваются внешние устройства и терминальное оборудование. Самое главное в тот период: унификация ЭВМ по конструктивно – технологическим параметрам. ЭВМ третьего поколения начинают выпускаться сериями или семействами, совместимыми моделями.

Дальнейшее развитие математического и программного обеспечения приводит к созданию пакетных программ для решения типовых задач, проблемно – ориентированных программных языков (для решения задач отдельной категории) и впервые создаются уникальные программные комплексы, – операционные системы (разработаны IBM). Представителями третьего

поколения были отечественные ЭВМ единой серии (ЕС ЭВМ), выпускавшиеся в разных комплектациях и моделях.

Четвёртое поколение. (1980-2000 годы)

Четвёртое поколение ЭВМ характеризуется применением больших интегральных схем (БИС) и сверхбольших интегральных схем (СБИС).

Это привело к снижению потребляемой мощности, увеличению быстродействия, улучшения многих других характеристик.

Ключевым фактом явилось появление персональных компьютеров. Началась эра персональных компьютеров.

Характерные свойства ЭВМ четвертого поколения:

1. Мультипроцессорность
2. Параллельно – последовательная обработка
3. Языки высокого уровня
4. Появляются первые сети ЭВМ

Технические характеристики ЭВМ четвертого поколения:

1. Средняя задержка сигнала 0.7 нс./вентиль (вентиль – типовая схема)
2. Впервые основная память – полупроводниковая. Время выработки данного из такой памяти 100-150 нс. Емкость 1012 –1013 символов.
3. Впервые применяется аппаратная реализация оперативной системы
4. Модульное построение стало применяться и для программных средств.

Основная внимание машин четвертого поколения было направлено на сервис (улучшение общения ЭВМ и человека).

Пятое поколение ЭВМ. (2000- по настоящее время)

Для ЭВМ пятого поколения свойственны следующие черты:

- дальнейшее совершенствование аппаратного и программного обеспечения;
- сближение персональных компьютеров и суперкомпьютеров, увеличение производительности;
- рост разработки компьютерных сетей и технологий.

Пятое поколение ЭВМ связывают с переходом к микропроцессорам. С точки зрения структурного построения характерна максимальная децентрализация

управления. С точки зрения программного и математического обеспечения – переход на работу в программных средах и оболочках. Для пятого и шестого поколения характерны многопроцессорные структуры, созданные на упрощенных микропроцессорах, которых очень много (решающие поля или среды). Создаются ЭВМ ориентированные на языки высокого уровня.

В этот период существуют две диаметрально противоположных тенденции:

1. Персонификация ресурсов.
2. Коллективизация ресурсов (коллективный доступ – сети).

Лекция 2

Основные характеристики ЭВМ

1. Операционные ресурсы.

Операционные ресурсы ЭВМ – это перечень возможностей ЭВМ.

Сюда включаются:

- способы представления информации в ЭВМ;
- система команд ЭВМ;
- способы адресации.

Операционные ресурсы ЭВМ напрямую связаны с аппаратными средствами, которые характеризуют степень приспособленности ЭВМ для решения тех или иных задач.

2. Емкость памяти (внешняя и основная) Основная память, какой бы большой она не была, всегда ограничена. Внешняя память не ограничена. Для характеристики компьютера используют емкость основной памяти. Использование памяти идет многобайтно, следовательно, доступ измеряется в байтах . Внешняя память – суммарная емкость всех накопительных устройств. Следовательно, необходимо использовать косвенную характеристику – количество накопителей, подключаемых к ЭВМ. В современных компьютерах есть также и сверхоперативная кэш-память, ее объем – один из важнейших параметров, влияющих на время решения задачи.

3. Быстродействие ЭВМ характеризует скорость обработки информации компьютером (число операций в секунду (V), время выполнения ($\tau=1/v$)). Но для различных операций эти показатели различны, следовательно, реальная характеристика – номинальное быстродействие ($V_н$)– количество коротких операций в единицу времени (обычно берут операцию “+”, а операнды хранятся во внутренних регистрах процессора (R-R)). Иногда также используют в качестве характеристики быстродействия – цикл обращения к основной памяти, а также эффективное быстродействие ($V_ф$) $V_ф=1/\sum p_i t_i$, где p_i – вероятность выполнения i-ой операции.

По содержанию производительность ЭВМ – это среднее число операций в единицу времени.

4. Производительность ЭВМ зависит от:

- быстродействия процессора;
- класса решаемых задач;
- порядка прохождения задачи через ЭВМ.

Для оценки числового выражения эффективности ЭВМ используют смеси команд.

Для научно-технических расчетов используют “Смесь Гибсона”

Таблица смеси Гибсона Табл.2.1.

Вид команды	Весовой коэффициент
Сложение и вычитание (с фикс. запятой)	33
Умножение (фикс. запятая)	0.6
Деление (фикс. запятая)	0.2
Сложение (плав. запятая)	7.3
Умножение (плав. запятая)	4
Деление (плав. запятая)	1.6
Логические операции	1.7
Безусловный переход	17.5
Условное ветвление	6.5

Виды производительности.

Пиковая – производительность ЦП без учёта времени обращения к ОП за операндами.

Номинальная – производительность ЦП с обращениями к ОП.

Эксплуатационная производительность – это производительность на реальной рабочей нагрузке, с использованием прикладных задач.

Единицы измерения производительности: MIPS – миллион команд в секунду.

MFLOPS – миллион операций над числами с пл. запятой в секунду.

GFLOPS – миллиард операций над числами с пл. запятой в секунду.

TFLOPS – триллион операций над числами с пл. запятой в секунду.

PFLOPS – квадриллион операций над числами с пл. запятой в секунду.

Оценки производительности.

Абсолютная оценка – определяется количеством элементарных работ, выполняемых в единицу времени. Относительная оценка – определяется для оцениваемой ЭВМ относительно базовой в виде индекса производительности. Пиковая производительность определяется средним числом команд типа «регистр-регистр», выполняемых в одну секунду, без учёта их статического веса в выбранном классе задач. Номинальная производительность определяется средним числом команд, выполняемых подсистемой «процессор-память» с учетом их статического веса в выбранном классе задач. Рассчитывается как правило по формулам и специальным методикам. Системная производительность – измеряется с помощью синтезированных типовых тестовых программ, реализованных на унифицированных языках высокого уровня. Они рассчитаны на использование базовых технических средств и позволяют измерять производительность для расширенных конфигураций технических средств. Эксплуатационная производительность – оценивается на основании использования данных о реальной рабочей нагрузке и функционировании ЭВМ при выполнении типовых производственных нагрузок в основных областях применения.

Методы определения производительности.

Расчётные - основаны на теоретических или эмпирических расчётах.

Экспериментальные – на основе применения аппаратно-программных измерителей.

Имитационные – моделирование для сложных ЭВМ.

4. Надежность ЭВМ. Надежность – свойство ЭВМ выполнять возложенные на нее функции в течение заданного промежутка времени, необходимого для решения поставленной задачи. В процессе функционирования ЭВМ возникают

отказы, связанные с неисправностью отдельных элементов либо соединений между ними.

По характеру проявлений отказы могут быть:

- внезапный отказ (механическое разрушение элементов);
- постепенный отказ (деградация параметров ЭВМ).

С точки зрения математического подхода – отказы — это случайное событие. Используется самая простейшая математическая модель – “Простейший поток отказов”. Если поток отказов простейший, то в качестве характеристики надежности используется величина интенсивности потоков отказа.

λ=1/Тр Тр – среднее время безотказной работы между двумя очередными отказами. Если ЭВМ можно ремонтировать, то после находки отказа

– работоспособность компьютера восстанавливается (Тв – среднее время восстановления) Тв – фактически время, которое происходит от момента обнаружения отказа до полного восстановления работоспособности. Для компьютеров с простейшим потоком отказов в качестве показателя используют показатель готовности: Кг = Тр/(Тр+Тв) который характеризует вероятность того, что в данный момент времени компьютер готов к решению требуемой задачи.

5. Показатель стоимости – суммарная стоимость всего оборудования, входящего в состав ЭВМ. Если возрастает количество оборудования ЭВМ, то в конечном итоге, будет расти не только стоимость, но будет расти и ее производительность. Путем статистического анализа была выведена связь между стоимостью и производительностью. Впервые это было установлено Найтом и получило название “Закон Гроша”.

$$V = K * S^2$$

k – константа определяется эмпирически.

Вывод, если не менять технологическую базу компьютеров, то:

- при росте стоимости ЭВМ растет количество оборудования и, следовательно, снижается скорость решения задачи.
- при росте стоимости ЭВМ растет объем оборудования и, следовательно,

увеличивается время ремонта. T∑=Tсч+Tр т.е. для данного уровня технологии всегда есть некоторая оптимальная стоимость, которая дает лучшие технические характеристики.

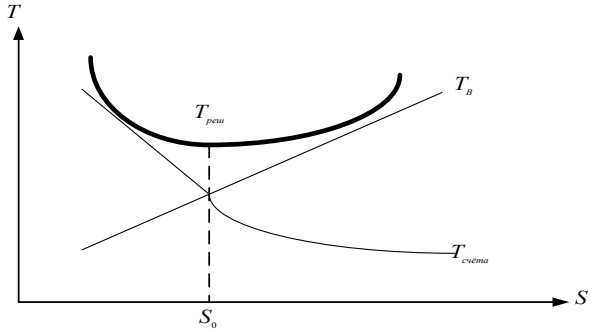


Рис. 2.1. График нахождения оптимальной стоимости ЭВМ.

Другие эксплуатационные характеристики ЭВМ:

- разрядность слов и кодовых шин интерфейса;
- типы системного и локального интерфейсов;
- ёмкость НЖМД;
- ёмкость и уровни кэш-памяти;
- тип видеоадаптера и видеомонитора;
- наличие сетевых средств;
- тип ОС и прикладного ПО.

Классификация ЭВМ

Классификация ЭВМ производится по следующим признакам:

1. По назначению. Обычно выделяют ЭВМ общего применения и ЭВМ ориентированные на вполне определенный класс задач.
2. По производительности. ЭВМ подразделяются по величине производительности.
3. По режимам работы.

а) однопрограммные ЭВМ

б) мультипрограммные ЭВМ (Эти ЭВМ должны иметь большую оперативную память, средства управления временем, ввода-вывода, средства позволяющие исключить влияния программ друг на друга);

в) ЭВМ для построения много машинных и многопроцессорных вычислительных систем (дополнительно к мультипрограммным ЭВМ должны реализовывать функции взаимного обмена между ЭВМ)

г) ЭВМ для работы в системах реального времени. (Говоря о машинах реального времени наиболее очевиден пример, когда ЭВМ управляет техническим объектом (автопилот). К ним предъявляют требования быстродействия и способность получать массу сигналов от внешних источников)

4. По способу структурной организации.

Для увеличения скорости ЭВМ в ее состав включают несколько процессоров. Различают:

а) Однопроцессорные ЭВМ

б) Мультипроцессорные ЭВМ, которые состоят как из однотипных, так и из разнотипных процессоров (неоднородные ЭВМ).

Основная цель мультипроцессирования – получение сверх высокой производительности вычислительных систем (ВС). Как правило, такие системы содержат несколько десятков, сотен или тысяч сравнительно простых процессоров, и их число позволяет увеличивать производительность. Принципиально такие системы ориентируются на большой круг

задач, которые допускают эффективное распараллеливание вычисление на регулярную структуру (связи между процессорами, как правило, фиксированы). Вообще-то не каждая задача достаточно хорошо распараллеливается на заданную ВС. ВС с параллельной обработкой также классифицируются. В качестве такой классификации выступает классификация по Флину. В ее основе лежит способ организации параллелизма ВС (множественность). Этот параллелизм определяется как максимальное число одновременных команд или операндов, которые

находятся на одинаковой или какой-то определенной стадии выполнения. Согласно классификации, предложенной Флином существует 4 разновидности ВС:

1 ОКОД (SISD), одиночный поток команд одиночный поток данных)

Рис. 2.2 а). Такое структурное построение характерно для классических машин фон Неймана.

Линейная организация вычислительного процесса обуславливает весьма низкую эффективность аппаратных средств (велик коэффициент простоя). Для повышения работы такой структуры применяются методы локального параллелизма – совмещенная или опережающая выборка команд, расслоение памяти, но, как правило, это требует дополнительных аппаратных затрат.

2. ОКМД (SIMD, одиночный поток команд множественный поток данных) Рис. 2.2 в). Для данной ВС обычный поток команд воздействует на несколько процессорных блоков одновременно, которые обрабатывают различные данные по одной команде. Память в такой ВС является разделенной.

Первоначально типовыми представителями таких ВС были супер-ЭВМ (ILLIAC IV, команда STARAN, PEPE, ПС-300). ВС с такой структурной организацией направлены на решение задач с естественным параллелизмом. В современных ЭВМ это реализовано в Pentium MMX.

3. МКОД (MISD, множественный поток команд одиночный поток данных) Рис.2.2 б). Эту ВС обычно рассматривают как результат идей локального параллелизма. Иначе их называют конвейерные ВС. Операционная часть Вс является регулярной и представляет собой цепочку последовательно (линейно) соединенных процессорных блоков, которые образуют конвейер процессора.

Впервые такую ВС разработал академик Лебедев.

4. МКМД (MIMD, множественный поток команд множественный поток данных) – общий случай мультипроцессорной системы Рис.2.2 г). В общем случае связи между элементарными процессорами являются изменяемыми. Такая ВС позволяет повысить не только производительность, но и надежность. Как правило отказ одного процессора не приводит к выходу из строя всей

системы. При такой организации ВС возникают сложности взаимодействия управления, при решении одной задачи. Иногда MIMD называют «моделью коллектива вычислителей»

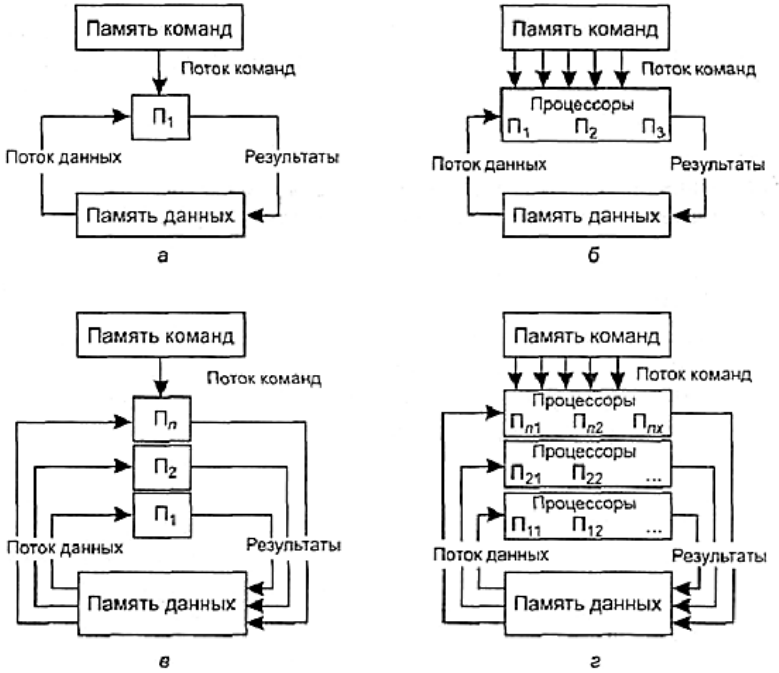


Рис. 2.2. Классификация вычислительных систем по Флину.

Лекция 3

Представление данных в ЭВМ

Типы данных, с которыми работает ЭВМ представлены на рис. 3.1.

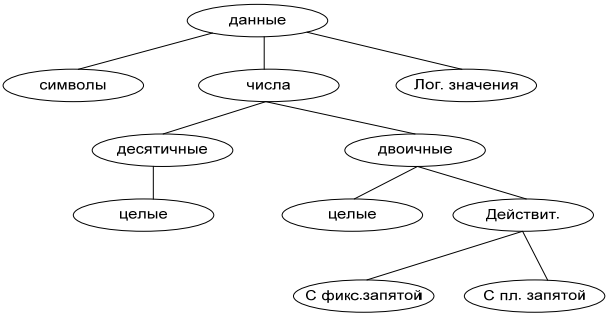
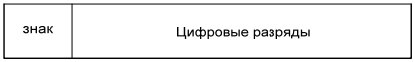


Рис. 3.1 Основные типы данных ЭВМ.

1. Целые двоичные числа со знаком.



Кодировка знака: 0 – «+», 1 – «-». Диапазон представления:

$$[-(2^n-1);+(2^n-1)]$$

Особенности выполнения операций с целыми числами:

- при сложении или вычитании – возможность переполнения из-за нехватки разрядов и инициирование прерывания;
- при умножении результат требует 2n разрядов;
- деление. C=A/B; в качестве результата берётся целая часть частного. Недопустимо деление на ноль. Делитель не может быть равен нулю.

2. Двоичные числа с фиксированной запятой. Диапазон представления:

$$[-(1-2^{-n});+(1-2^{-n})]$$

Точность представления: 2^{-n}. Особенность выполнения арифметических операций:

- при сложении и вычитании результат по модулю должен быть меньше 1;
- при операции умножения: результат занимает 2n разрядов, младшие n отбрасываются.

- при операции деления:

$$B \neq 0$$

$$|A / B| < 1$$

3. Двоичные числа с плавающей запятой.

Знак мантиссы	Мантисса (дв. число с фикс. запятой)	Знак порядка	порядок

Структура числа: число с плавающей запятой состоит из знака мантиссы (указывающего на отрицательность или положительность числа), мантиссы (выражающей значение числа без учёта порядка), знака порядка, порядка (выражающего степень основания числа, на которое умножается мантисса). Представление числа:

$$Z = \pm M * d^{\pm P},$$

где d-основание числа с плавающей точкой.

$$d = 2^n, n = 1, 2, 3.....$$

Нормальная форма и нормализованная форма чисел с плавающей запятой.

Нормальной формой числа с плавающей запятой называется такая форма, в которой мантисса (без учёта знака) находится на полуинтервале [0; 1).

Такая форма записи имеет недостаток: некоторые числа записываются неоднозначно (например, 0,0001 можно записать в 3 формах:

$$0,001*10^{-1} \quad 0,01*10^{-2} \quad 0,1*10^{-3}$$

Нормализованная форма, в которой мантисса десятичного числа принимает значения от 1 (включительно) до 10 (не включительно), а мантисса двоичного числа принимает значения от 1 (включительно) до 2 (не включительно).

В такой форме любое число (кроме 0) записывается единственным образом.

Недостаток заключается в том, что в таком виде невозможно представить 0, поэтому представление чисел в информатике предусматривает специальный признак (бит) для числа 0.

Так как старший разряд (целая часть числа) мантиссы двоичного числа (кроме 0) в нормализованном виде равен «1», то при записи мантиссы числа в ЭВМ старший разряд можно не записывать, что и используется в стандарте IEEE754.

Алгоритм перевода вещественных чисел с плавающей точкой:

- записать число в нормализованном виде;
- вычислить смещённый порядок: к порядку числа прибавить десятичное число 127;
- записать код числа в заданном формате по стандарту IEEE754.

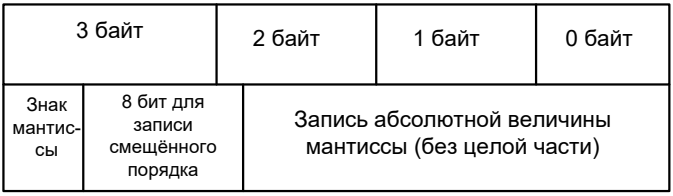


Рис. 3.2. Структура двоичного числа с плавающей запятой по стандарту IEEE754.

Из-за ограничений на разрядность мантиссы и порядка возможны ситуации:

- потеря значимости: М=0, П не =0 -машинный ноль;
- «исчезновение порядка»;

$$П < -2(2^n - 1)$$

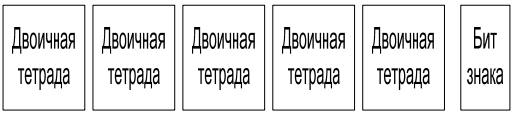
- деление на 0.

По принятому стандарту IEEE 754

структура представления чисел с плавающей запятой принята следующей:

4. Десятичные целые числа.

Числа обрабатываются последовательно разряд за разрядом начиная с разрядов младшей тетрады.



Применяются упакованный и зонный форматы представления.

В упакованном формате кодируются в байте две десятичные цифры, а в зонном формате одна, старшие 4 бита содержат зону из четырёх единиц. Длина числа может быть произвольная.

5. Строки символов.

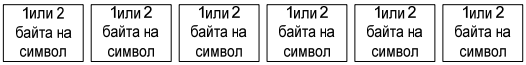


Таблица кодировки — это стандарт, ставящий в соответствие каждому символу алфавита свой порядковый номер. Международным стандартом для персональных компьютеров стала таблица ASCII. На практике можно встретиться и с другой таблицей — КОИ-8.

Unicode. Это 16-разрядная кодировка, т.е. в ней на каждый символ отводится 2 байта памяти. Такая кодовая таблица допускает включение до 65536 символов.

6. Логические значения. Как правило это многоразрядные двоичные последовательности. Например: 10000110100011;

1110001100100

7. Видеоинформация. Может быть динамической и статической.

Динамическая – 70 раз в сек. Смена кадра. Слайд-фильм можно отнести к статическому виду. В ЭВМ существует два вида представления графических изображений: растровый (матричный) и векторный способы.

В растровых изображение представляется прямоугольной матрицей точек-пикселов, положение которых соответствует координатам точек на экране. Кроме координат, пиксел характеризуется цветом, градацией яркости. 32 бита кодирует 4096 млн. цветов. Известные видеоформаты :

BMP, GIF, PCX, jPEG, TIFF, PNG.

В векторных форматах изображение задаётся не пикселями, а графическими примитивами, которые могут быть описаны математически. Это могут быть:

- линии и ломаные линии;
- многоугольники;
-окружности и эллипсы;
- сплайны;
- безигоны.

8. Аудиоинформация.

Частоты аудиосигналов лежат в диапазоне от 15 до 20 кГц, а сигналы являются аналоговыми. С помощью АЦП аудиосигналы переводятся в двоичный код.

Для высококачественного представления рекомендуется16-разрядное представление амплитуды сигнала (2 в 16 степени градаций уровня звука) и частота выборки порядка 40 кГц (промежуток времени между последовательными выборками не более 25 мкс.). Типовые форматы аудиоданных: AVI; WAV; MIDI; AIF; MPEG.

Классификация машинных команд

Базовая машинная команда - команда, которая определяет в процессоре операцию, без учета способов адресации, которые могут быть применены в данной команде, не учитывая какие регистры общего назначения используются при выполнении программы. Если учитывать регистры в командах, то их число может достигать 200-400. В различных программах, а также в программах для различных ЭВМ, частота появления различных команд оказывается разная. Основной характеристикой любой команды, является ее формат.

Все команды любой ЭВМ при рассмотрении разделяют
по следующим признакам.

1. По функциональному назначению:

- команды передачи данных;
- команды обработки данных;
- команды передачи управления;

- дополнительные команды.

2. По адресности команды:

- безадресные;

- одноадресные;

- двухадресные;

- прочие.

3. По способам адресации:

- по способам адресации данных;

- по способам адресации команд.

4. По способу кодирования операций:

- команды с фиксированным полем кода операций;

- команды с расширяющимся полем кода операций.

5. По длине команды:

- однобайтные;

- двухбайтные;

- трёхбайтные;

- многобайтные.

Функциональное назначение команды определяет ее код операции.

Команды передачи данных – группа команд включающая в себя три подгруппы.

Первая подгруппа. Это команды обмена внутри процессора.

Команды регистровой пересылки, которые обычно имеются в системе команд процессора, обеспечивает либо однонаправленный обмен, либо взаимный обмен. С точки зрения длины команд, самые короткие – команды пересылки, как правило двухадресные. В отдельных случаях, команды могут быть одноадресные, если существует фиксированный регистр адресной команды, не определенный мнемоникой, а машинным форматом.

Вторая подгруппа. Команды обмена процессора с памятью. Связаны с передачей данных из памяти в регистры и из регистров в память. Если используется команда MOV и возможности адресации достаточно большие, то

с помощью команды MOV в отдельных ЭВМ возможна пересылка память – память.

Третья подгруппа. Команды передачи кодов между процессором и периферией. Происходит передача данных между процессором и периферийным устройством. В ЭВМ используются два принципиально разных варианта для обмена с внешними устройствами:

- специальные команды ввода-вывода (in, out). Такие команды применяются, если внешнее устройство имеет автономное адресное пространство памяти. В таких командах, как правило, адресуется только один операнд, другой операнд располагается в аккумуляторе.
- единая команда MOV. Используется в тех случаях, если регистры внешних устройств рассматриваются как часть общего адресного пространства компьютера. Это позволяет оперировать с внешними устройствами, как с обычными ячейками памяти компьютера. Разделение внешних устройств и памяти производится на аппаратном уровне.

Команды обработки данных.

Классификация команд обработки данных ведётся в зависимости от операций, которые они выполняют над данными:

1. арифметические;
2. логические;
3. команды сдвига;
4. команды обработки строк.

1. Базовые арифметические команды предназначены для задания арифметических операций над какими-то операндами. Любая арифметическая операция двухместная (пример: ADD dst,src; Схема вычислений: dst:=(dst)*(src) (dst)*(src)⇒dst). Если использовать аккумуляторный принцип, то ACC:=(ACC)*(src). Команды арифметических операций формируют практически всегда признаки результата операций. Базовой арифметической операцией является арифметическое сложение (сложение двоичных кодов, т.к. сложение без знаковое). Большинство ЭВМ не ограничиваются операцией

сложения, имеется еще вычитание двоичных кодов (SUB dst,src). Эта операция не коммутативная. Сама по себе команда вычитания двоичных кодов обеспечивает вычитание без знаковых кодов. Но сформулировав должным образом коды можно и обрабатывать данные со знаком. Если обрабатывать многобайтные данные, то т.к. система счисления двоичная позиционная, то обработка начинается с младшего разряда. Как правило, для этих случаев имеются специальные команды (ADC dst,src; SBB dst,src; Схема вычисления dst:=(dst)*(src) ± (CY)).

Если в арифметических операциях могут участвовать операнды разной длины, то предварительно короткий операнд увеличивается до длины длинного (для целочисленной арифметики), причем здесь идет выравнивание по правому краю, а расширение идет с помощью знака. Это может реализовываться автоматически, либо за счет команд расширения знака (SXT). Как правило, расширение команды происходит в фиксированном регистре, следовательно, эти команды безадресные. Если этого нет, нужно писать специальные процедуры расширения. Операция сравнения кодов (CMP dst,src). По содержанию это команда вычитания. Фактически схема вычисления (dst)-(src). Все признаки результата по этой команде формируются, а результат никуда не заносится.

Однооперандные арифметические команды. Т.к. второй операнд имеет фиксированное значение (как правило, оно =1), тогда команды INC dst; DEC dst; , а схемы вычислений dst:=(dst) ±1. Эти команды очень используемы при разработке счетчиков, индексов. Неприменимы для многобайтных данных, позволяют сохранить признаки для следующих ветвлений. Команды умножения и деления. В системах команд малых ЭВМ эти команды отсутствуют, но если в системе команд имеются эти команды, то они применяются для беззнаковых данных. Формат команды: MUL dst,src; DIV dst,src; Если взять команды ADD и SUB, то форматы результата и операндов практически совпадают, здесь же они не совпадают принципиально. Для

хранения произведения обычно используются фиксированные регистры, чаще всего это аккумулятор с расширителем. Аналогично для целочисленного деления. В общем случае делимое имеет двойную длину, следовательно, как правило, приемник результата тоже фиксированный регистр (аккумулятор с расширителем). Очень часто в системах команд вводят команды умножения и деления с учетом знака (IMUL dst,src; IDIV dst,src;). Эти команды обычно ориентированы на использование базовых форматов компьютера. Для обработки многобайтных данных при умножении и делении, сначала делают декомпозицию, а затем строить умножение или деление многобайтных данных. Если машина ориентирована на научно - технические расчеты, то требуются операции над данными с плавающей запятой и там они имеются. Говоря о малых ЭВМ, то, собственно говоря, команд с плавающей арифметикой нет. Но операция сложения и вычитания выполняются подпрограммами или с помощью сопроцессора. Команды десятичной арифметики.

Основу десятичной арифметики в любых ЭВМ составляют команды двоичной арифметики. Обычно к эти командам относят команды десятичной коррекции. Обычно эти команды зависят от того, какой формат используется (упакованный или неупакованный). Не зависимо от формата, команды основаны на аккумуляторном способе (т.е. команды десятичной коррекции безадресные). В этом случае десятичная обработка состоит из двух фаз: 1) соответствующее двоичное действие 2) коррекция с учетом десятичного числа

2. Команды логических операций.

Логические команды в системе команд ЭВМ играют не только вспомогательную роль, но в логических задачах могут быть основными операторами обработки. Для этого система логических операций в ЭВМ должна быть функционально полной. Как правило, а систему команд закладываются избыточные логические операции. Операции выполняются побитно и одновременно справа на лево.

Команды логической обработки одно и двух операндные. Одноместную операцию реализует отрицание: NOT dst; Схема dst:= □(dst). Двухоперандные логические операции реализуют: &, ∪, ⊕. AND dst,src; OR dst,src; XOR dst,src; dst:=(dst)*(src);. Чаще всего логические операции применяются для решения трех возникающих проблем.

1. Маскирование и выделение определенных разрядов операндов. Реализуется с помощью операции конъюнкции (&) с заданной маской.
2. Формирование требуемых значений в требуемых битах (с помощью ∪).
3. Инвертирование определенных битов (с помощью ⊕) .

Логические операции избирательно действуют на флаги, т.е. часть флагов после выполнения операции не изменяются (OVR:=0; CY:=0), некоторые могут иметь неопределенное значение(AF:=?). В системе команд логических операций еще существуют команды (TEST dst,src);

Схема (dst)&(src)). В следствии выполнения этой операции, результат никуда не записывается, но формируются все флаги.

3. Команды сдвигов.

Все реализуемые команды сдвигов могут быть разделены по признакам:

1. вид сдвига (арифметический или логический)
2. направление сдвига
3. характер сдвига (простой или циклический)
4. по количеству разрядов, на которое сдвигается операнд после выполнения операции.

Формат команды содержит как минимум три поля.

Под логическим сдвигом понимается сдвиг числового кода операнда без учета его числового эквивалента.

Арифметический сдвиг – сдвиг числового кода операнда с учетом его числового эквивалента.

В том случае если для представления числовых значений не используются специальным образом формируемые коды (т.е. данные беззнаковые), то различий между логическим и арифметическими сдвигами нет. Если же для

представления числовых значений с учетом знака, применяются специальные коды (прямой, обратный, дополнительный), то арифметический сдвиг принципиально отличается от процедуры логического сдвига.

Пример:

1. Логический сдвиг.

E5 = 11100101 ACC:=L2(ACC) результат:

Точная запись: ACC:=L2(ACC(5:0)).00 Правый сдвиг аналогично.

Кольцевой сдвиг на 2 бита: ACC:=L2(ACC(5:0)).ACC(7:6)

2. Арифметический сдвиг.

Сдвиг числа влево и вправо соответствует делению или умножению числа на основание системы счисления.

11100101 двоичный код числа –27.

При арифметическом сдвиге влево 00101

сдвиге вправо: 11001 в дополнительном коде.

Лекция 4

Команды

Команда (или инструкция) представляет собой входное воздействие для управления микропроцессором. Её основное назначение инициировать выполнение конкретной операции над данными (операндами команды).

Команда инициирует шаги по реализации микропроцессором заданной операции над данными, указывает место их нахождения, тип данных и место расположения итогового значения результата операции над операндами. Машинное представление команды в памяти, состоящее из двоичного кода, называется объектным кодом команды.

Машинные команды

Под термином машинная команда понимается действие, инициализируемое на оборудовании процессор. Для отличия от команды, машинную команду именуют микрокомандой. Для выполнения микрокоманды требуется время, именуемое тактом. Микрокоманда в виде множества сигналов переводит систему в другое состояние, которое хранит память системы. По завершению выполнения всей предписанной последовательности микрокоманд, из памяти считывается многоразрядное слово сигналов управления Y, которые подаются на исполнительные устройства. Упорядоченная во времени совокупность микрокоманд образует команду, а совокупность команд и данных составляет программу.

Процесс выполнения команды. Под командным циклом понимается время, затрачиваемое на выполнение команды. Командный цикл разбивается на машинные циклы. Машинным циклом называется промежуток времени между двумя обращениями процессора у ОЗУ или внешнему устройству по системной шине.

Код команды представляет собой многоразрядное двоичное число, в котором можно выделить две части:

- код операции, задающий вид операции, выполняемой данной командой;
- код адреса операндов.

Коды команд хранятся в оперативной памяти.

Этапы выполнения команды:

- в начале первого машинного цикла по адресу, который задаётся содержимым программного счётчика, из ОЗУ считывается код команды, подлежащей выполнению. Код команды поступает в регистр команд микропроцессора, входящий в состав его устройства управления;
- при дешифрации кода команды определяется вид выполняемой операции и адреса участвующих в операции операндов, а также необходимое количество машинных циклов для выполнения команды. Команда выполняется за один цикл, если не требуется считывания операндов из памяти или записи операндов в память. Иначе количество машинных циклов возрастает и может доходить до 10..15;
- после считывания кода текущей команды содержимое программного счётчика увеличивается на 1. При выборке очередной команды содержимое программного счётчика поступает на шину адреса, обеспечивая считывание из ОЗУ следующей команды выполняемой программы. При реализации условных или безусловных переходов происходит загрузка в программный счётчик нового содержимого;
- в соответствии с выполняемой операцией устройство управления формирует необходимые сигналы для реализации машинных циклов и требуемую последовательность микрокоманд в каждом цикле.

Адресация команд

Для выполнения последовательности команд центральному процессору необходимо знать, где находятся данные для выполнения команды и как их получить для обработки, по какому адресу и в каком устройстве искать следующую для исполнения команду. Из этого следует задача адресации команд и данных.

Технология адресации команд.

При выполнении линейной части программ для выборки команд используется программный счётчик. Адрес следующей команды в текущей

команде не указывается. Необходимость указания адреса в текущей команде возникает при выполнении передачи управления подпрограмме, расположенной по другому адресу в памяти. Для таких случаев есть такие команды, как CALL, JMP, RET и др.

Технология адресации данных.

При выполнении команды из содержимого её адресных полей должно быть определено, откуда загрузить необходимые операнды и по какому адресу, и в каком устройстве сохранить результат операции. Источниками и приёмниками операндов могут служить регистровая память, память данных или порты ввода-вывода. Адрес операнда, как правило, именуют исполнительным адресом, а способ формирования исполнительного адреса – способом адресации.

Способ адресации – это правило определения адреса и операнда на основе информации в адресной части команды.

Эффективность способа адресации влияет на временные затраты и затраты на определённый необходимый состав оборудования. Критериями выбора способа адресации являются: емкость ЗУ для хранения программы; время выполнения программы; эффективность использования ячеек памяти при хранении программы. Эффективность способа адресации можно характеризовать двумя показателями: затратами оборудования на выборку операндов и затратами оборудования для обращения к памяти и затратами времени на доступ к данным.

Наибольшее распространение в вычислительных устройствах получили следующие способы адресации:

1. Непосредственная адресация. При этом способе адресации не требуется указания адреса операнда. Само значение операнда записывается в одном из полей команды после кода операции. Рис. 4.1.

Рис. 4.1. Структура команды при непосредственной адресации.

2. Прямая адресация. При прямой адресации адресная часть команды содержит непосредственный (прямой) адрес операнда в памяти. Рис. 4.2.

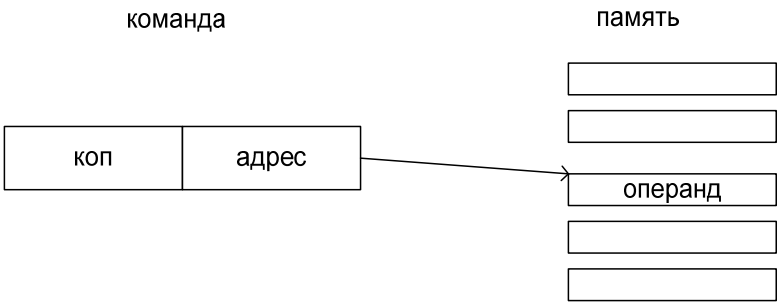


Рис. 4.2. Структура команды при прямой адресации и указание на расположение операнда в памяти.

3. Косвенная адресация. В случае косвенной адресации адресная часть команды содержит косвенный адрес, адресный код команды в этом случае указывает адрес ячейки памяти, в которой находится адрес операнда или команды. Косвенная адресация широко используется в малых и микро ЭВМ, имеющих короткое машинное слово, для преодоления ограничений короткого формата команды рис. 4.3.

4. Регистровая адресация. Данный тип адресации применяется, когда промежуточные результаты хранятся в одном из рабочих регистров центрального процессора (регистрах общего назначения (РОН)). Поскольку регистров значительно меньше, чем ячеек памяти, то небольшого адресного поля может хватить для адресации рис. 4.4.

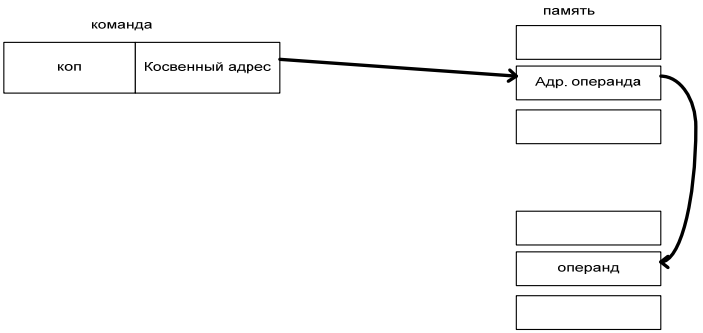


Рис. 4.3. Структура команды с косвенной адресацией и указание пути для доступа к операнду в памяти.

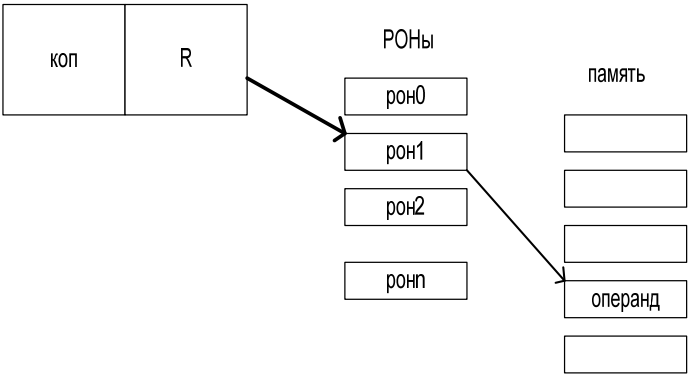


Рис. 4.4. Структура команды при регистровой адресации и указание пути к операнду.

5.Стековая адресация. Стековая память широко используется в современных ЭВМ. Хотя адрес обращения в стек отсутствует в команде, он формируется схемой управления рис. 4.5.

Для чтения записи доступен только один регистр v вершина стека. Этот способ адресации используется, в частности, системой прерывания программ при вложенных вызовах подпрограмм. Стековая память реализуется на основе обычной памяти с использованием указателя стека и авто индексной адресации.

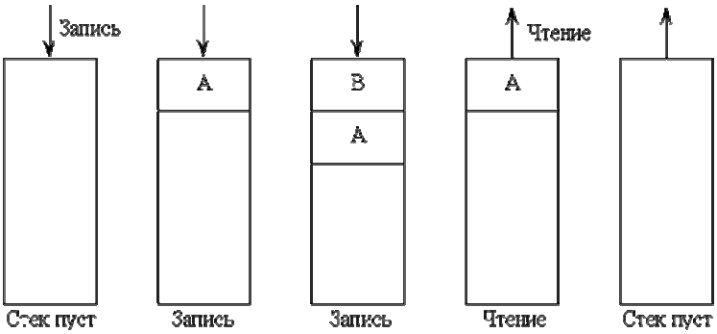


Рис. 4.5. Схема извлечения и записи операндов в случае стековой адресации.

Запись в стек производится с использованием автодекрементной адресации, а чтение - с использованием автоинкрементной адресации.

6. Адресация с модификацией адресов. Особенность использования такого типа адресации определился на основании вывода, что при решении математических задач и обработки данных характерна цикличность вычислительных процессов, когда одни и те же процедуры выполняются над различными операндами, упорядоченно расположенными в памяти.

Программирование циклов существенно упрощается, если после каждого выполнения цикла обеспечено автоматическое изменение в соответствующих командах их адресных частей согласно расположению в памяти обрабатываемых операндов. Такой процесс называется модификацией команд, и основан на возможности выполнения над кодами команд арифметических и логических операций.

7. Индексная адресация. Для работы программ с массивами, требующими однотипных операций над элементами массива, удобно использовать индексную адресацию. Последовательность использование полей команды при индексной адресации иллюстрирует рис. 4.6.

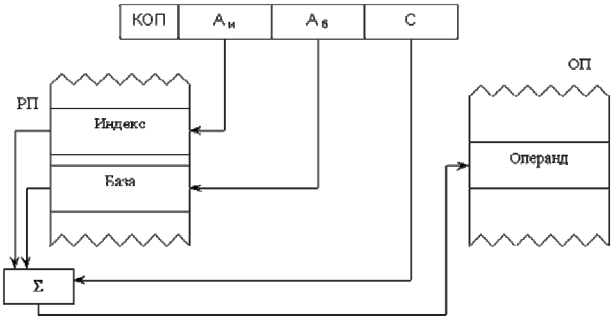


Рис. 4.6. Структура команды при индексной адресации и последовательность определения адреса операнда.

Адрес i-того операнда в массиве определяется как сумма начального адреса массива операнда, задаваемого смещением S, и индекса I, записанного в одном из регистров регистровой памяти, называемым индексным регистром.

Адрес индексного регистра задается в команде полем адреса индекса Аи.

В каждом i-том цикле содержимое индексного регистра изменяется на постоянную величину, как правило, это 1.

8. Прямая адресация с модификацией. В полях команды содержится Рон, в котором текущий индекс, а во втором поле команды базовый (начальный) адрес.

9. Регистровая адресация с модификацией. Поля команды содержат два значения РОН: в первом значение индекса, второй содержит значение базового адреса.

10. Страничная адресация. Используется при обращениях к оперативной памяти рис. 4.7.

Рис. 4.7 Структура команды при использовании страничной адресации.

Форматы команд

Совокупность полей, требующихся для выполнения операции, задаваемой конкретной командой именуется форматом команды. Формат команды должен определять: функциональное назначение операции в виде кода операции, адреса источников данных, адрес места сохранения результата операции данной команды, адрес следующей выполняемой команды.

В структуре форматов команды выделяют операционную и адресную часть полей команды. В поле операционной части указывается код команды, в адресной части поля адресов операндов (при прямой адресации) и адреса сохранения результата.

Наибольшее распространение получили следующие форматы команд рис. 4.8.

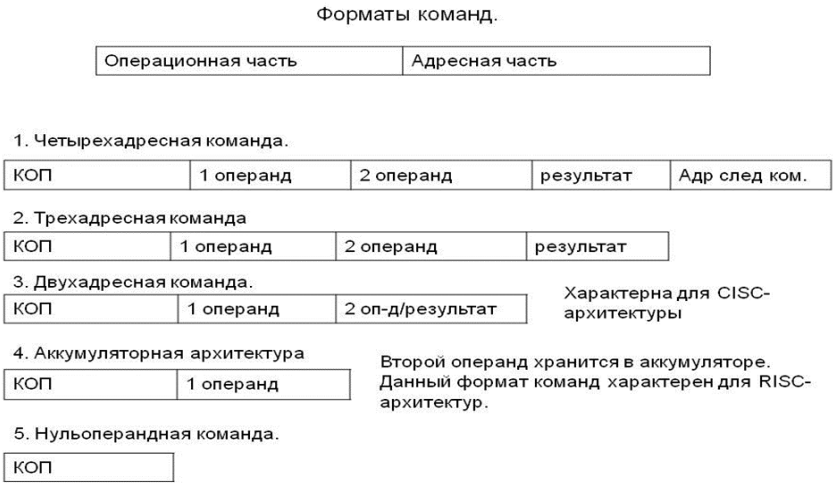


Рис. 4.8. Набор применяемых в процессорах форматов команд.

Пятый (нуль операндный) формат команды используется при применении стека для размещения операндов и результата выполнения операции.

Лекция 5

Теоретические основы логического проектирования операционных устройств

Согласно Фон-Неймановской классификации первоначальным операционным устройством (ОУ) можно считать АЛУ, выполнявшее арифметические и логические операции, присущие определённому типу ЭВМ. Современная архитектура вычислительных средств уже характеризует АЛУ не как единое устройство, а в виде комплекса операционных устройств, которым свойственны свои операции и типы данных. В наборе операционных устройств ЭВМ такие ОУ:

- ОУ обработки целых чисел;
- ОУ обработки чисел с фиксированной запятой;
- ОУ обработки чисел с плавающей запятой;
- ОУ для логической обработки данных;
- ОУ десятичной арифметики.

Совершенствование аппаратной структуры ОУ ведёт к уменьшению доли программной реализации функций, что приводит к увеличению быстродействия.

ОУ различаются структурным базисом, т.е. наполнением базовых составных элементов.

Самые распространённые компоненты ОУ в его структурном базисе следующие:

- регистры, обеспечивающие кратковременное хранение слов;
- управляемые шины, предназначенные для передачи слов данных;
- комбинационные схемы, реализующие вычисление логических условий и выполнение микроопераций по сигналам от устройства управления.

Использование классической методики проектирования ОУ [2] даёт путь проектирования канонической структура ОУ. Каноническая структура предполагает максимальную производительность по сравнению с другими вариантами структур, однако по затратам оборудования является избыточной.

Принципы микропрограммного управления операциями.

Для выполнения операций над информацией в ЭВМ используют различные операционные устройства (ОУ) - процессоры, контроллеры, шинные формирователи и т.д. Функция ОУ - выполнение заданного множества операций F={f 1,… f G} над входными словами D={d1,… dH} с целью вычисления выходных слов.

R={r1,… rQ} где R=fg(D), g=1,G

Функциональная и структурная организация ОУ в современных ЭВМ базируется на следующих принципах микропрограммного управления:

1. Любая операция fg, g=1,G ,реализуемая устройством, рассматривается как сложное действие, которое разделяется на последовательность элементарных действий над словами информации, называемых микрооперациями.

2. Для управления порядком следования микроопераций используются логические условия, которые в зависимости от значений слов, преобразуемых микрооперациями, принимают значения "истина" или "ложь" (I или 0).

3. Процесс выполнения операций в ОУ описывается в форме алгоритма, представленного в терминах микроопераций и логических условий и называемого микропрограммой. Микропрограмма определяет порядок проверки значений логических условий и следования микроопераций, необходимой для получения результатов.

4. Микропрограмма является основой для определения структуры и функционирования ОУ во времени.

Применение принципов микропрограммного управления позволяет формализовать и автоматизировать проектирование ОУ различного назначения.

В общем случае любое ОУ разделяется на две части - операционный и управляющий автоматы: Рис. 5.1.

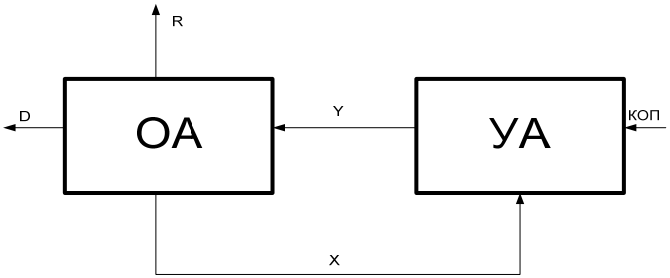


Рис. 5.1. Типовая структура операционного устройства.

Операционный автомат (ОА) служит для хранения слов информации, выполнения набора микроопераций и вычисления значений логических условий, т.е. ОА выполняет действия над информацией. Микрооперации (МО), реализуемые ОА, инициируются множеством управляющих сигналов Y={y1 ,…,yM}, с каждым из которых связана определенная МО.

Значения логических условий, вычисляемых в ОА, отображаются множеством осведомительных сигналов X={x1,…,xL}, каждый из которых связан с определенным логическим условием.

Управляющий автомат (УА) генерирует последовательность управляющих сигналов Y={y1,…,yM}, предписанную микропрограммой и соответствующую значениям логических условий. Таким образом, УА задает порядок выполнения действий в ОА, вытекающий из алгоритма выполнения операции.

В вычислительном устройстве может выполняться несколько различных операций. Выбор наименования операции, которую необходимо выполнить, задается кодом операции (КОП).

Таким образом, любое ОУ - процессор, контроллер ввода-вывода, устройство управления внешними устройствами - являются комбинацией ОА и УА. ОА, реализуя действия над словами информации, является исполнительной частью устройства, работой которого управляет УА, генерирующей последовательности управляющих сигналов.

Основные характеристики ОУ

Основные характеристики ОУ - быстродействие и затраты оборудования.

Вероятности pi определяются классом решаемых задач. ОУ функционирует в дискретном времени t = 0,1,2.... Промежуток между двумя последовательными моментами t и (t+1) дискретного времени называется тактом работы ОУ или машинным тактом.

В течение такта формируется набор управляющих сигналов, выполняются соответствующие МО и вычисляются значения логических условий. Длительность такта T зависит от сложности МО (сложение, сдвиг) и логических условий и от быстродействия элементов, из которых построено ОУ. С учетом этого среднее время выполнения операции будет tоп =T * n

где n - среднее число тактов, за которое реализуется одна операция.

T – такт работы ОУ. Затраты оборудования в ОУ определяются суммарной стоимостью C элементов, из которых состоит ОУ.

Наибольшее влияние на быстродействие ОУ и затраты оборудования оказывает набор МО Y={ym}и логических условий X={xl}.

Например, если ОУ должно реализовать заданное множество операций F={f 1,… f G } над словами D для получения результатов R, то оно может быть построено различными способами. Рассмотрим два из них:

1. Множество всех операций F={f 1,… f G } реализуется всего одной МО с помощью комбинационной схемы за один такт (это вполне возможно).

2. Множество всех операций F={f 1,… f G } выполняется с помощью элементарной МО типа функции Шеффера над отдельными разрядами.

На выходе схемы формируется единственное логическое условие X, значение которого 0 или I определяется результатом МО. Теоретически доказано, что в терминах функций трёх известных функционально полных наборов булевых функций можно описать любой алгоритм, т.е. система является функционально полной. В этом случае ОА в каждом такте может выполнять одну МО под управлением УА. При этом ОА выполняет операции в виде последовательности простейших операций над отдельными разрядами.

Рассмотренные два варианта построения ОА обладают диаметрально противоположными свойствами:

Рассмотренные два варианта построения ОА обладают диаметрально противоположными свойствами:

1. Первый вариант является наиболее быстродействующим, второй - наименее быстродействующим.

2. В первом ОУ управляющий автомат отсутствует, в то время как во втором ОУ на него приходится максимальная доля затрат оборудования.

3. ОА в первом устройстве наиболее сложен по конструкции, а во втором - наиболее прост.

Между двумя рассмотренными вариантами существует целый ряд вариантов с промежуточными характеристиками.

Исходные данные для проектирования ОУ

Обычно при проектировании ОУ бывает известным:

- перечень входов D и выходов R устройства,
- набор операций $F=\{f_1,\ldots f_G\}$
- ограничение на среднее время выполнения операции.

Требуется спроектировать схему ОУ, обеспечивающую реализацию заданных функций с заданным быстродействием при минимальном количестве оборудования.

Для проектирования схем ОУ выполняемые операции должны быть описаны в виде микропрограмм.

Микропрограмма, описывающая работу ОУ безотносительно к его структуре, называется функциональной микропрограммой (ФМП).

ФМП содержит в себе алгоритм выполнения операции, рекомендуемый проектировщиком, и используется как основа для выбора структуры ОУ.

Для записи ФМП разработан язык функционального микропрограммирования - Ф-язык. Средства языка обеспечивают описание слов, МО, логических условий и порядка их выполнения.

Язык функционального микропрограммирования (ЯФМП)

ЯФМП - язык предназначен для описания работы операционных устройств на уровне микропрограмм безотносительно к их структуре. Рассмотрим

инженерную версию языка, в которой используется общепринятая математическая символика, таблицы в их обычной форме и графическое представление схем алгоритмов.

Машинно-ориентированная форма языка описана в книге [2]:

Средствами ЯФМП - языка описываются ФМП, определяющие алгоритмы выполнения операций в устройствах.

Лекция 6

ЯФМП: описание слов и массивов, двоичные операции и двоичные выражения

Функциональная микропрограмма состоит из двух частей:

1. Описание слов и массивов, устанавливающее типы и форматы слов.

2. Содержательный граф микропрограммы, который определяет алгоритм в виде описаний микроопераций и логических условий.

Описание слов и массивов.

Слово - это основной элемент информации в ФМП. Описание слова содержит: наименование, формат и тип слова, характеризующий способ присваивания и использования значений слова. Наименование и формат слова задаются в виде: C(n1:n2)

где C - идентификатор

n1 и n2 - номера старшего и младшего разрядов слова.

Разряды слова нумеруются справа налево неотрицательными целыми числами. Причем n2 < n1 .

Так, описание A(0:31) определяет 32-разрядное слово A, описание B(1:8) - 8-е разрядное слово B.

Описание одноразрядного слова состоит только из одного идентификатора. Так, описания Z,ПП,ТР – определяют три одноразрядных слова.

Массив - совокупность слов, имеющих одинаковую длину.

Массив описывается в виде M=[m1:m2](n1:n2)

где M- идентификатор массива

m1 и m2 - границы номеров слов, составляющих массив,

причем m2>m1

n1 и n2 - номера старшего и младшего разрядов слова. Например, описание РП[0:15](0:31) представляет массив из 16-ти 32-разрядных слов.

Действия в микропрограмме могут производиться как над словами, так и над полями.

Поле - часть слова, имеющая самостоятельное значение. Поля описываются следующими конструкциями:

1) C(p1:p2) ,C(p)

где C - идентификатор слова C(n1:n2)

p1 и p2 – номера старшего и младшего разрядов поля, причем p1<p2 ,n1≤p1<n2, n1<p2 ≤ n2

p – номер разряда слова C(n1:n2), где n1≤ p ≤n2

Здесь первая конструкция выделяет (p2-p1+1) -разрядное

поле, а вторая - P разряд слова C.

Полям можно присваивать собственные имена. Наименования полей вводятся следующими описаниями:

G(n1:n2)= C(p1:p2); H=C(p)

где G и H - имена полей слова C, причем n1-n2 = p2-p1.

В зависимости от способа использования все слова бывают четырех типов:

I) Входные - значения им присваиваются вне микропрограммы (МП), а используются - внутри МП (тип-1).

2) Внутренние - значения присваиваются и используются внутри МП (тип L).

5) Вспомогательные - аналогичны внутренним, но значения сохраняют только в течение одного машинного такта (A).

4) Выходные - значения присваиваются внутри МП, а используются - вне МП (тип - O).

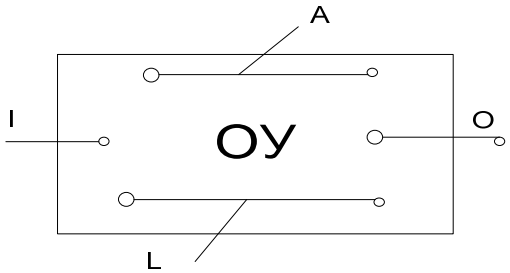


Рис. 6.1. Графическая интерпретация типов слов.

Некоторые слова могут характеризоваться сразу несколькими типами – IL,LO,ILO.

Все слова, которые используются в ФМП, должны быть описаны в виде таблицы:

Таблица описания слов и полей ФМП. Табл. 6.1.

ТИП	Наименование и формат	ПОЛЯ	ПОЯСНЕНИЯ
I	Д(0:15)	-	Входное слово
L	A(0:15)	ЗнA=A(0)	1-й операнд
L	B(0:15)	ЗнB=B(0)	2-й операнд
LO	C(0:15)	ЗнC=C(0)	Результат
L	Сч(0:15)	-	Счетчик
A	X	-	Признак

Двоичные выражения.

Для записи преобразований, выполняемых микрооперациями, и для записи условий используются двоичные выражения.

Двоичное выражение — это последовательность первичных двоичных выражений, соединенных знаками двоичных операций.

Двоичные выражения определяют правило вычисления двоичного значения путем выполнения операций над первичными двоичными выражениями.

Первичные двоичные выражения — это константа, слово, элемент массива, или поле.

Разрешается использовать двоичные, восьмеричные, десятичные и шестнадцатеричные константы:

10102, 468, 9310, 1C516

Слова в двоичных выражениях представляются своими идентификаторами, т.к. их разрядность определена в описании. Например, Д, A, B, C, Сч, ЗнA, ЗнC.

Слово, являющееся элементом массива записывается в виде M[i], где M - идентификатор массива, а i - номер слова в массиве, т.е. $m1 \leq i \leq m2$.

Поля записываются в виде A(0), A(1:4), A(14:15), где указывается идентификатор слова и границы поля или представляются непосредственно идентификаторами, имеющими поля: ЗнA, ЗнB, ЗнC.

В двоичных выражениях элементы массивов и поля используют наравне со словами, представляя соответствующие двоичные значения.

Первичные двоичные выражения - константы, слова и поля объединяются в двоичные выражения с помощью двоичных операций (см.табл. 6.2.).

Старшинство операций Табл. 6.2.

Операция	Наименование	Старшинство
¬	Инверсия	1
·	Составление (конкатенация)	2
∧	Конъюнкция	3
∨	Дизъюнкция	4
⊕	Сложение по модулю 2	4
+	Сложение	5
⊞	Циклическое сложение	5
_	Вычитание	5

При выполнении бинарных операций (∧,∨,⊕,÷,*,) операнды совмещаются по младшим разрядам. При этом операнд с меньшим числом разрядов дополняется нулями со стороны старших разрядов до выравнивания длины.

Порядок вычисления двоичного выражения определяется скобками, старшинством операций и последовательностью операции одинакового старшинства.

Микрооперации.

В синтаксической смысле микрооперация (МО) - это оператор присваивания, посредством которого слову или полю присваивается значение двоичного выражения.

МО состоит из левой части, знака присваивания:= и двоичного выражения.

В левой части оператора присваивания указывается слово, поле или составное слово вида A.B.

Рассмотрим примеры записи микроопераций:

1. A:=0 - присвоение слову A нулевого значения.
2. B(1):=1 - присвоение полю слова B значения I.
3. A:=B - присвоение слову A значения слова B.

4. C:=C+1 - увеличение значения слова С на I.

5. C:=C-1 - уменьшение значения слова С на I.

6. C:=C+A - увеличение значения слова С на значение слова А.

7. C:=C+!A+1 - сложение С с дополнительным кодом слова А,
т.е. вычитание А из С.

8. A.B:=X - присваивание слову А старших разрядов слова Х,
а слову В - младших.

Операция присваивания выполняется следующим образом. Сначала вычисляется значение двоичного выражения, стоящего в правой части оператора присваивания, а затем вычисленное значение присваивается слову, стоящему в левой части оператора присваивания.

МО выполняется за один такт. Если в левой части оператора указано
l-разрядное слово, а двоичное значение содержит r, разрядов, то

- при l < r слову присваивается l младших разрядов двоичного выражения, а (r-l) старших разрядов двоичного выражения отбрасываются.
- при l > r старшим (l-r) разрядам присваиваются "0" и r младшим разрядам - значение двоичного выражения.

Классификация микроопераций.

Исторически выделяют следующие классы микроопераций:

1) установки, 2) инвертирования, 3) передачи, 4)сдвига, 5) счета, 6) сложения, 7) бинарные логические, 8) комбинированные.

1) Микрооперация установки - присваивает слову значение константы.

Например,

A:=0 C:=1 К(1:17):=12710

B:=1111 DC:=1F16

2) МО инвертирования -изменяет значение слова на инверсное. Например, A:=! A,

C(0): =!C(0)

3) МО передачи - присваивает слову значение другого слова, в том числе инверсии или составного слова:

A:=B C:=!A

A(0):=B(0) C:=11.A(1:15)

4) МО сдвига - изменяет положение разрядов слова по отношению к начальному путем перемещения каждого разряда на К позиций влево или вправо.

Любая МО сдвига может быть представлена в форме оператора присваивания. Рассмотрим различные сдвиги слова А(1:32);

1. A:=A(2:32).0 - сдвиг на I разряд влево с введением 0 в освобождающийся при сдвиге разряд.

2. A:=A(2:32).A(1) - циклический сдвиг на I разряд влево.

3. A:=00.A(1:30) - сдвиг на 2 разряда вправо с введением нулей в освобождающиеся разряды.

4. A:=A(31:32).A(1:30) - циклический сдвиг на два разряда вправо.

Для сокращения записи МО сдвига в Ф-языке используются две стандартные процедуры:

а) RK(A) - удаление из двоичного выражения А К младших (правых) разрядов, т.е. сдвиг А на К разрядов вправо.

б) LK(A) - удаление из двоичного выражения А К старших (левых) разрядов, т.е. сдвиг А на К разрядов влево. С использованием этих процедур, рассмотренных выше примеры будут иметь вид:

1) A: =L1(A.0)

2) A: =L1(A.A(1))

3) A: =R2(00.A)

4) A: =R2(A(31:32).A)

МО сдвига надо отличать от МО передачи. Так если А(1:31) и В(1:31) - 31-разрядные слова, то

A: =0.B(1:30)

будет не МО сдвига, а МО передачи слова В со сдвигом на один разряд вправо, т.е. передача поля В(1:30).

В ЭВМ различают три типа сдвигов:

логические, арифметические и циклические.

При логическом сдвиге сдвигаются все разряды слова, а освобождающиеся разряды заполняются нулями. Выдвигаемые разряды теряются.

При арифметическом сдвиге знаковый разряд не сдвигается. Освобождающиеся при сдвиге разряды числа, представленного в дополнительном коде, заполняются содержимым знакового разряда числа (0 - для положительных чисел, I - для отрицательных). Выдвигаемые разряды теряются.

При циклическом сдвиге крайние разряды слова как бы соединяются между собой так, что выдвигающиеся разряды слова поступают в освобождающиеся позиции этого же слова.

5. МО счета - изменяет значение слова на единицу:

A:=A+1 C:=A+1

B:=B-1 C(1:4):=C(1:4)+1

6. МО сложения - присваивает слову значение суммы слагаемых:

C:=C+A C:=A+B

C:=C+!B C:=A+!B+1

7. Бинарные логические МО - присваивают слову значение, получаемое поразрядным применением логических операций к парам соответствующих разрядов слагаемых:

C:=A∧B; C:=A∨B; C:=A⊕B; C:=A∨!B

Так, если A(1:4)=1011 и B(1:4)=1010, то

A∧B=1010;A∨B=1011;A⊕B=0001;A∨!B=1111

8. Комбинированные МО - содержат несколько действий, присущих

МО разных классов: C:=A∨B+A∨D

Совместимость микроопераций

Для реализаций машинной операции в вычислительном устройстве необходимо выполнить несколько МО. Причем, некоторые из них могут выполняться параллельно во времени, в то время как другие - только строго последовательно. Микрооперации, которые могут выполняться одновременно,

называются совместимыми, а МО, которые не могут выполняться одновременно, называются несовместимыми.

Совместимость МО обусловлена:

1. Содержанием МО - это так называемая функциональная совместимость. Она определяется алгоритмом.

2. Ограничениями структуры вычислительного устройства - это структурная совместимость.

В функциональных микропрограммах, описывающих алгоритм выполнения операции без привязки к конкретной структуре вычислительного устройства, параллельно могут выполняться только функционально совместимые МО. Структура обычно вносит ограничения на количество параллельно выполняемых МО. Поэтому, если структура задана, то совместимыми называются структурно совместимые МО.

Две МО S1:=f1(S2) и S3:=f3(S4), где S1 ,S2 , S3 , S4 $\subset$S

подмножества слов из S, называются функционально совместимыми, если S1$\cap$S3=Ø т.е. МО присваивают значение различным словам. Например, S1:=f1(S2, S3) и S2:=f2(S2) функционально совместимы, а МО S2:=f2(S2) и S2:=f3(S3) -функционально несовместимы. Так как совместное выполнение МО S1:=f1(S2, S3) и S2:=f2(S2) сводится к одновременному вычислению значений двоичных выражений f1(S2, S3) и f2(S2) , после чего эти значения одновременно присваиваются разным словам S1 и S2 , указанным в левых частях операторов присваивания.

Условием совместимости n МО является совместимость каждой пары МО.

Функционально совместимые МО могут оказаться структурно несовместимыми. Это происходит если МО используют общее оборудование вычислительного устройства.

Логические условия

Логическое условие представляет из себя булеву функцию, которая может быть «истина» и «ложь». Булева функция состоит из первичных булевых

выражений, которые связываются между собой знаками булевых операций: !(инверсия),∧,∨,⊕.

В качестве первичных булевых выражений используются одноразрядные слова или поля, а также отношения. Отношения имеют вид C1*C2 ,где * - знак операции отношения: =,≠ < , ≤, >, ≥ .

Например, A(0), A(0)∨A(1), A(0)⊕A(1), A=0,B ≥2

Содержательный граф микропрограммы

В вычислительной технике алгоритмы выполнения машинных операций обычно описывают в графической форме – содержательным графом МП. Граф строится с использованием вершин 4-х типов и дуг связывающих эти вершины.

Типы вершин графа

1. Вершина «начало». Определяет начало микропрограммы. Начальная вершина - отмечает начало алгоритма и имеет единственный выход.

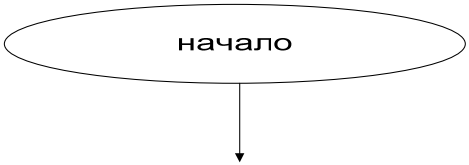


Рис. 6.2. Начальная вершина.

2. Функциональная вершина. Функциональная вершина - определяет совокупность совместимых МО, выполняемых одновременно в данном такте. В функциональную вершину может входить любое количество дуг (n≥1) и из вершины выходит только одна дуга.

3. Условная вершина. Условная вершина - используется для разветвления вычислительного процесса. Причем, если условие имеет значение 0, то выход из блока выполняется по дуге, отмеченной символом "0". В условную вершину может входить любое число дуг (n≥1), а выходят всегда две дуги.

4. Конечная вершина. Конечная вершина - отмечает конец МП. В нее может входить любое число дуг (n≥1).

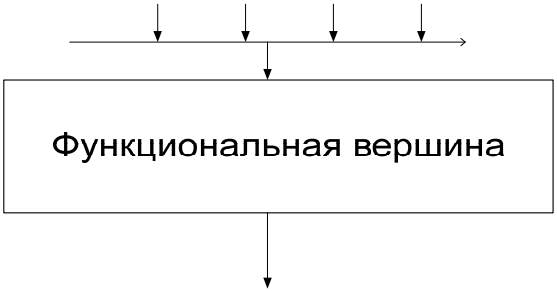


Рис. 6.3. Функциональная вершина.

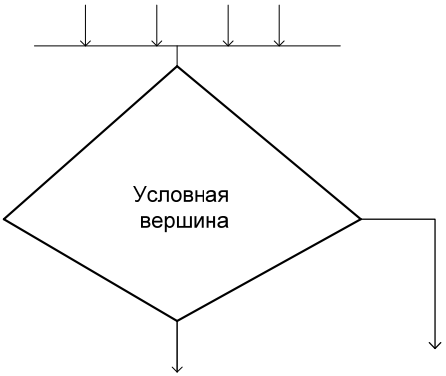


Рис. 6.4. Условная вершина.

Рис. 6.5. Конечная вершина.

Граф МП считается корректным, если выполняются следующие условия:

1. Граф должен содержать одну начальную и одну конечную вершины.

2. В любую вершину, кроме начальной, должна входить хотя бы одна дуга.

3. Из каждого выхода каждой вершины должна выходить одна дуга.

4. При всех допустимых значениях слов должен существовать путь из начальной вершины в конечную.

Пример составления функциональной микропрограммы

Пусть вычислительное устройство предназначено для умножения двоичных чисел с фиксированной запятой, представляемых в формате:

Знак	Цифровые разряды

Произведение будем представлять в том же формате с учетом округления. Для умножения используем алгоритм вычисления произведения, начиная от младших разрядов множителя:

I. Произведение полагается равным нулю.

2. Если младший разряд множителя равен I, то произведение увеличивается на значение модуля множимого.

3. Произведение и множитель сдвигаются на I разряд вправо, в результате чего в младший разряд множителя вводится очередная цифра множителя.

4. Действия 2 и 3 повторяют для обработки всех 4 цифр множителя.

5. Произведение округляется по значению 5-го разряда.

6. Если знаки сомножителей одинаковы, то произведению присваивается знак плюс (0), иначе - минус (I).

Сначала определим слова, которые необходимы для выполнения умножения по этому алгоритму.

1. Операнды – множимое и множитель – будем представлять слова A и B заданного формата. Значения словам A и B присваиваются вне программы, а используются они внутри программы. Следовательно, их тип – 1 L.

2. Произведение будем определять словом C того же формата. Произведение вычисляется внутри программы и выдается во вне. Следовательно, тип слова C - L O.

3. Так как все цифры множителя обрабатываются одинаково, то целесообразно организовать цикл.

4. Для определения момента завершения цикла необходим счетчик числа повторений. В нашем случае количество повторений равно 4. Поэтому в начале операции счетчик устанавливается в состояние 4 и после обработки каждой цифры его значение должно уменьшаться на 1. Переход счетчика в "О" свидетельствует об окончании цикла.

5. Чтобы закодировать в счетчике значение 4 надо 3 двоичных разряда. Следовательно, счетчик должен иметь формат Су (1:3). Значение счетчика используется только в МП, поэтому слово Су должно иметь тип L.

6. Произведем описание всех слов МП:

Тип	Слово	Поле	Пояснение
1	2	3	4
IL	A(0:4)	A(0)	множимое
IL	B(0:4)	B(0)	множитель
L0	C(0:4)	C(0)	произведение
L	C4(1:3)		счетчик циклов

Функциональная МП, описывающая алгоритм операции умножения, состоит из описания слов и содержательного графа МП, который строится на основе алгоритма выполнения операции умножения. Выполнение операции начинается с присваивания произведению С нулевого значения и установки счетчика СЧ в начальное состояние 4. Эти МО являются совместимыми, что позволяет объединить их в один функциональный оператор.

Вычисление произведения начинается с анализа значения младшей цифры множителя. Если B(0)=1, то произведение С увеличивается на значение модуля множимого, представляемого в разрядах A(1:4). В следующем операторе производится сдвиг цифровых разрядов 1:4 множителя В с целью выделения следующей цифры множителя, сдвиг произведения С и уменьшение значения счетчика на 1. Для сохранения младших разрядов произведения разряд С(4) передается в старший разряд слова В, освобождающихся при сдвиге множителя.

Для определения момента окончания цикла проверяется условие C4=0. До окончания обработки 4 цифр множителя счетчик находится в состоянии, отличном от 0. Поэтому в МП выполняется переход на обработку следующей цифры. Когда C4=0, выработка произведения заканчивается. Округление результата производится добавлением 1 к 4 разряду слова C, если разряд B(1)=1. Если B(1)=0, то округление не производится. МО C(4):=A(4)⊕B(4) определяет знак произведения. Если знаки сомножителей одинаковые, то A(0)⊕B(0)=0 и произведение будет положительным. Если же знаки сомножителей различны, то A(0)⊕B(0)=1 и произведение будет отрицательным.

Оценка времени выполнения МП.

Быстродействие операционного устройства характеризуется временем ton выполнения операции: ton = T * N

где T – длительность такта ОУ

N – количество тактов, необходимые для выполнения МП.

В общем случае количество тактов N зависит от значения операндов. Поэтому значение N характеризуют средним числом тактов.

В операционном устройстве в зависимости от его структуры за один такт может выполняться различное число действий – МО и проверок логических условий. В каждом такте могут выполняться только структурно совместимые МО. Содержательный граф МП учитывает только функциональную совместимость МО, но не учитывает структурной совместимости.

Для оценки числа тактов N используют временной граф МП, в котором в отличии от содержательного графа ФМП каждая операторная вершина может содержать только структурно совместимые МО.

Во временном графе МП обычно операторных вершин получается больше, чем в функциональном графе МП, так как структурно несовместимые МО приходится выносить в дополнительные вершины. При этом каждая вершина временного графа МП будет обрабатываться в устройстве за один такт.

Условные вершины во временном графе обычно не указываются, так вычисление значений условий всегда может выполняться параллельно с выполнением МО. И для их вычисления не требуется дополнительных тактов. Поэтому вычисление и анализ логических условий происходит одновременно с выполнением МО и соответствует одной из операторных вершин.

Лекция 7

Логическое проектирование операционного автомата

В функциональной организации центрального процессора выделяются две его функциональные структуры устройств: операционный и управляющий автоматы.

Операционный автомат предназначен для приема, хранения и выдачи слов информации, для выполнения МО и вычисления логических условий. Для реализации этих функций ОА необходим набор элементов, достаточный для построения структуры ОА с заданными свойствами.

Такой набор элементов называется структурным базисом ОА. Для реализации ОА наиболее широко используется следующий набор элементов: триггеры и регистры – для хранения слов, шины – для передачи слов и комбинационные схемы для вычисления значений функций.

Многие из перечисленных устройств были изучены в курсе «Схемотехника дискретных устройств» и подробно изложены в методических пособиях по данному курсу.

Операционный автомат выполняет следующие функции:

- хранит множество входных слов, выходных слов и внутренних слов;
- выполняет операцию из набора его операций для получения результата;
- вырабатывает условные осведомительные сигналы, которые определяют конкретное логическое условие.

Рис. 7.1. иллюстрирует функциональную структуру операционного автомата.

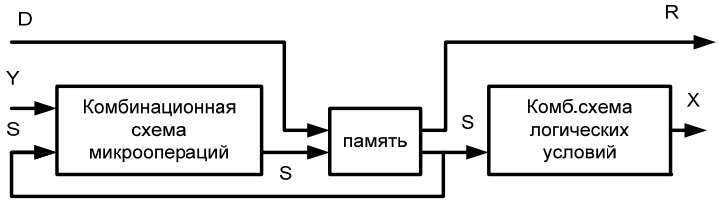


Рис. 7.1. Функциональная схема операционного автомата.

D-обрабатываемы данные операционным автоматом, Y – управляющие сигналы, инициирующие микрооперации, S- внутренние слова, X – осведомительные сигналы (условия), R – результат выполнения микроопераций.

Для инициации микрооперации в операционном автомате все возможные в нём микрооперации связываются с множеством управляющих сигналов Y=(y1,y2…yn), которые являются внешними для операционного автомата и порождаются в соответствии с заложенным алгоритмом в управляющем автомате.

Более подробно разберём применение шин в структуре операционного автомата. Для передачи бита информации (двоичной переменной) необходима одна цепь. Совокупность цепей, используемых для передачи слова, называется шиной.

Шина получает наименование передаваемого по ней слова и описывается точно также, как и слово: A (1 : n). Цепи в шинах обозначаются номерами или разрядами слова.

Изображение цепей рис. 7.2.:

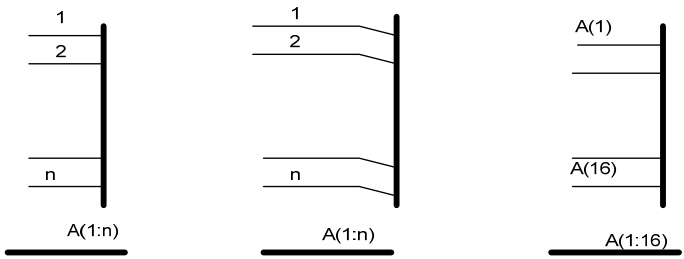


Рис. 7.2. Примеры обозначений цепей на изображении шин.

Для реализации передачи вида B: = A используются управляемые шины рис. 7.3.

yi - управляющий сигнал, инициируемый МО передачи B: = A.

B:= A1, если y1=1; B:=A2, если y2=1,……, B:=An, если yn=1.

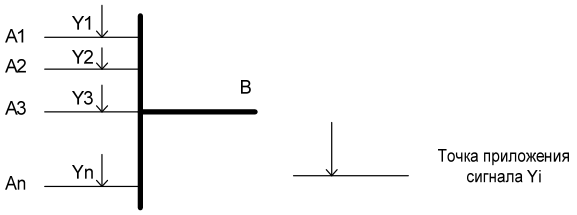


Рис. 7.3 Реализация управляющего переключение на выходную шину B

Управляемая шина реализует функцию вида:

0, если yi = 0; B = Ai, если yi = 1

Каждая шина должна иметь свой собственный уникальный идентификатор, а также входные и выходные цепи с собственными идентификаторами.

Шины могут изгибаться, разветвляться, пересекаться.

По шинам можно передавать информацию от одного источника к нескольким приёмникам. (управляемое демультиплексирование).

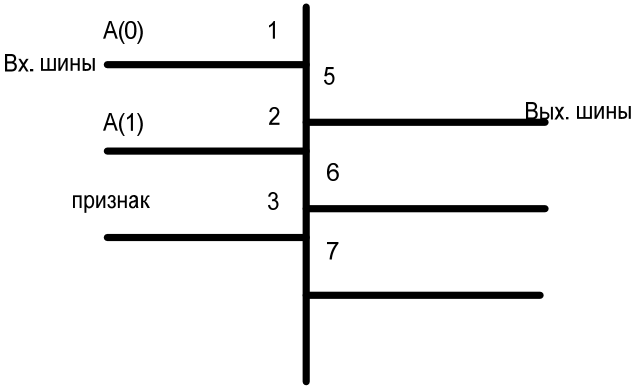


Рис. 7.4. Условно-графическое обозначение шины.

По шинам можно передавать информацию от многих источников к одному приёмнику. (управляемое мультиплексирование).

В ряде случаев к одному приемнику B должны передаваться разные слова A1,…,Ак.

В каждый момент времени приемник B может принять только одно слово, так что различные слова могут передаваться только последовательно во времени.

В этом случае каждая из них А1 … Ак, подключаемая к общему приемнику должна быть управляемой и по сигналу ук = 1 осуществлять передачу В: = Ак. Совокупность управляемых шин с общим выходом называется мультиплексором

Описание операционного и управляющего автоматов

Обычно операционное устройство ЭВМ предназначено для выполнения некоторого множества операций:

F = f1, …, fG

Каждая функциональная программа М1, … MG, описывающая алгоритмы выполнения соответствующих операций f1, …, fG в ОУ, несет в себе информацию о функциях операционного и управляющего автоматов, из которых состоит ОУ.

Перед тем, как проектировать ОА и УА, необходимо предварительно описать работу каждого из них.

Описание операционного автомата

ОА определяется множеством слов S, с которыми оперирует автомат, множеством МО Y и множеством логических условий X. Каждое из этих множеств может быть определено для каждой операции из соответствующей ФМП Mi, i = I,G в виде Si , Yi и Xi . Но эти множества характеризуют ресурсы ОА только частично - они достаточны только для выполнения i-той операции.

Для того, чтобы охарактеризовать ресурсы ОА, необходимые для выполнения всех G операций, требуется построить объединённые множества S, Y и X :

$$S = \bigcup_{i=1}^{G} Si \; ; \quad У = \bigcup_{i=1}^{G} Yi \; ; \quad X = \bigcup_{i=1}^{G} Xi$$

Построение множества S – обобщённого описания слов.

Обычно в ОА затраты оборудования тем меньше, чем меньше слов содержится в S, а точнее, чем меньше суммарное число разрядов в словах S. Поэтому при объединении подмножества слов S1, …, SG необходимо отождествлять между собой слова, принадлежащие разным подмножествам, т.е. одни и те же слова использовать в разных ФМП. Для этого отождествляемые слова для простоты объединения должны иметь одинаковые идентификаторы. В этом случае процесс объединения множеств S1, … SG сводится к перечислению во множестве S слов с различными идентификаторами. Причём, если слова имеют разную длину, то во множество S включается слово с максимальной длиной.

Построение множества Y - получается путём выборки из содержательных графов алгоритмов операторов присваивания, т.е. путём перечисления всех попарно различимых МО. Построение множества условий X – производится тоже путём перечисления всех попарно различимых условий.

Таким образом, описание ОА представляется таблицами, содержащими описание слов, список МО и список логических условий.

Пример

Пусть в ОУ реализованы две операции, функциональные МП для которых характеризуются следующие множествами рис. 7.5.

Номер МП	Микрооперации	Форматы слов	Условия
1.	C: = 0 C: = 4 C_4: = C + A C: = R 1 (0.C) C_4: = C_4 – 1	C (0:4) C_4 (1:3) A (0:4)	A = 0 C_4 = 0
2.	C: = 0 C: = A + B C_4 = 14 C_4 = C_4 - 1	C (054) C_4 (1:4) A (0:4) B (0:4)	C_4 = 0 B > 0
Обобщенная МП	C: = 0 C_4: = 4 C: = C + A C: = R 1 (0.C) C_4: = C_4 – 1 C: = A + B C_4: = 14	C (0:4) C_4 (1:4) A (0:4) B (0:4)	A = 0 C_4 = 0 B > 0

Рис. 7.5. Пример обобщённой МП из МП1 и МП2.

Синтез канонической структуры операционного автомата

Для проектирования структуры ОА необходимы следующие исходные данные

1) Структурный базис.

2) Множество входных D = d1, d2, ..., dn , выходных R = r1, r2, ..., ra и внутренних слов S = S1, ..., SN

3) Множество МО Y = y1, ..., yM

4) Множество логических условий X = X1, ..., Xl

В этом случае структура ОА проектируется следующим образом:

1. Внутренним словам S1, ..., SN ставятся в соответствие регистру S1, ..., SN с длинами n1, ..., nN равными длинами слов. Если слово Si разделяется на поля, в регистре Si выделяются соответствующие разряды регистра.

2. Входным словам d1, ..., dn ставятся в соответствие входные полосы (входы)

d1, ..., dн структурной схемы. Каждый вход d1, ..., dн соединяется с регистром S1, ..., SN шиной.

3. Выходным словам R1, ..., Rq ставятся в соответствие выходные полосы (выходы) R1, ... ,Rq структурной схемы. Каждый регистр S1 ... ,Sn соединяется с выходами R1 ... ,Rq шиной.

4.Каждой МО yi Є Y, описываемой оператором вида

Sl = φm (S\βi. ... ,Sβi)

ставится в соответствие комбинационная схема φm, входы которой подключаются к выходам регистров Sβ1 …, Sβк , а выходы соединяются управляемой шиной с регистром Sα. Управляемая шина отмечается сигналом Ym, инициирующим МО Sα = φm (Sβ1 …, Sβк).

Для МО передачи Sα:= Sβ не требуется комбинационная схема. Поэтому МО передачи реализуется управляемой шиной, соединяющей регистр Sβ с регистром Sα и отмеченной соответствующим управляющим сигналом Ym.

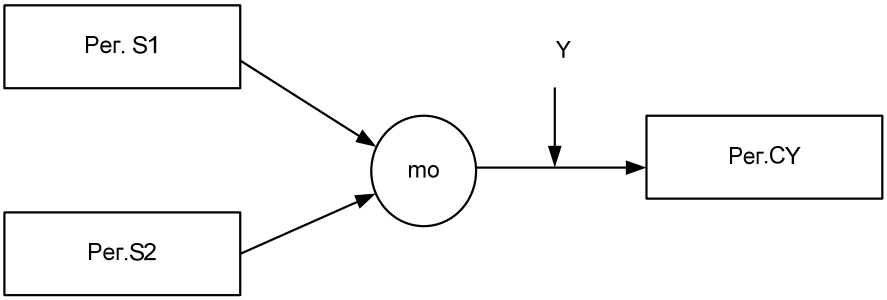


Рис. 7.6. Каноническая структура управляемой микрооперации.

$$S_\alpha := \varphi_m (S_1, \ldots, S_k)$$

Аналогично реализуется МО установки Sα:= const. В этом случае начало управляющей шины отмечается const, а сама шина управляющим сигналом.

5. Каждому логическому условию Xl = φl (S1, …, Sк), l = 1, L ставится в соответствие комбинационная схема Fi:

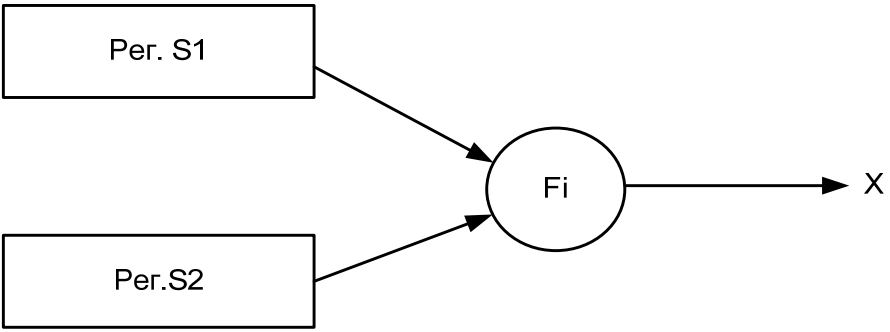


Рис. 7.7. Каноническая структура выработки условия X.

$$X_l = \varphi_l(S_1, \ldots, S_k)$$

Входы комбинационной схемы соединяются с выходами регистров S1, …, Sк, а выход отмечается осведомительным сигналом Xl.

Описание микроопераций для операции умножения табл.7.2.:

Описание микроопераций операции умножения. Табл. 7.2 .

Слово	Микрооперация	Управляющий сигнал Y	Логическое условие	Обозначение X
A(4:0)	A:= Швх	y1	A(4)=0	X1
B(4:0)	B:= Швх	y2	Сч=0	X2
C(4:0)	B:=R1(C(0).B)	y3	B(4)=0	X3
Сч(3:0)	C:=0	y4		
Швх(4:0)	C:=C+A	y5		
Швых(4:0)				
	C:=R1(0.C)	y6		
	Сч:=4	y7		
	Сч:=Сч-1	y8		

Для рассмотренного примера ФМП умножения каноническая структура ОА будет иметь вид:

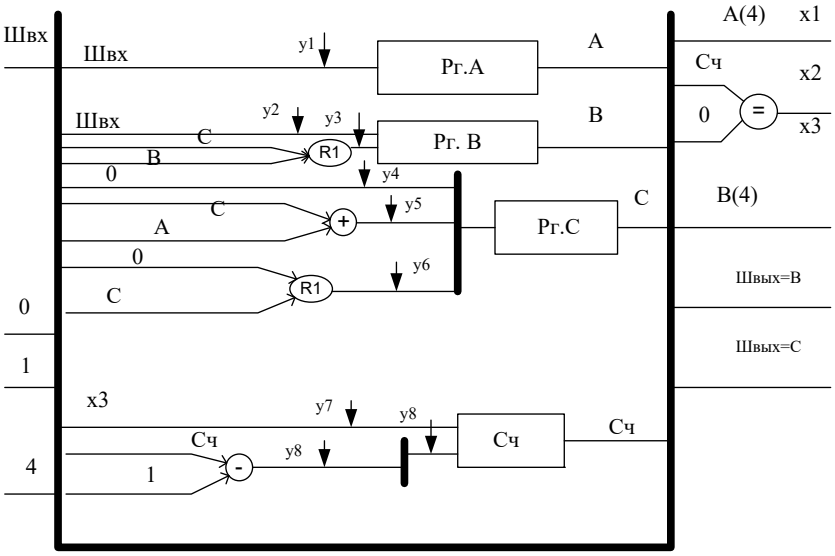


Рис. 7.8. Функциональная схема классического ОА для операции умножения.

Как видно из структурной схемы ОА состоит из трех частей:

- памяти S
- комбинационной схемы φ, которая реализует МО: У1, …, УМ
- комбинационные схемы Fi, вычисляющей значения логических условий X.

Разделим множество комбинационных схем φ на отдельные подмножества φ1, …, φN , каждое из которых будет обслуживать только один из регистров S1, …, Sn. Т.е. комбинационные схемы, обслуживающие i-тый регистр выделим в i-тое подмножество, φi.

Схему, состоящую из регистра Si и комбинационной части φii можно рассматривать как элементарный автомат, который будем называть операционным элементом. В этом случае ОА разбивается на совокупность

операционных элементов, число которых определяется количеством слов в МП, и схемы выработки логических условий.

ОА работает под управлением УА по тактам. В каждом такте выполняются следующие действия:

1. УА формирует один или несколько управляющих сигналов Уα, …,Уω.
2. Под действие управляющих сигналов комбинационные схемы φ вычисляют значения двойничных выражений φα, …, φω.
3. Вычисленные значения записываются в соответствующие регистры.
4. Комбинационные схемы Ψ вычисляют новые значения осведомительных сигналов X.

ОА работают на основе синхронного принципа. При этом длительность такта постоянна и определяется максимальным временем, необходимым для выполнения любой МО и вычисления значения любого логического условия. Такты задаются генератором синхронизирующих сигналов, который вырабатывает синхронизирующие импульсы с периодом следования равным длительности такта Т. Эти сигналы используются для синхронизации моментов переключения триггеров и регистров.

Эффект гонок в автоматах.

В ФМП встречаются МО, присваивающие слову Si значения, которое зависит от значения того же слова Si. Например, Si:= Si + Sj. При реализации подобных МО возникают вредные эффекты – гонки сигналов и проскок состояний. Эффект гонок заключается в следующем.

В момент поступления управляющего сигнала инициирующего МО, в схеме сигналы начинают передаваться с выходов регистров через комбинационные схемы φ к выходам триггеров, регистров, которым должны присваиваться новые значения, вычисленные комбинационными схемами. Под воздействием этих значений триггеры регистров должны переключаться в новые состояния. Однако, сигналы, поступающие на входы триггеров, приходят не одновременно из-за различных временных характеристик электрических цепей. Поэтому в течение некоторого времени в регистре будет храниться

неверное значение, которое может повлиять на весь результат МО, либо вызвать формирование ошибочного значения логического условия, т.е. может прекратиться выработка одних сигналов возбуждения под воздействием других. Такое явление называется гонками. Результат гонок – нарушение работы ОА и искажение результатов.

Проскок состояний происходит, если ОА за один такт несколько раз меняет свое состояние. Гонки и проскок состояний нарушают устойчивость функционирования автомата.

Эффект гонок и проскока состояний полностью исключаются, если в регистрах использовать синхронные триггеры с задержкой или триггеры с динамическими входами. Эти триггеры реагируют только на значения сигналов, существовавшие к моменту начала переключения, а все дальнейшие изменения сигнала не сказываются на состоянии триггера.

Лекция 8

Построение операционных элементов на основе регистров, счётчиков, мультиплексоров

Операционным элементом является одно из устройств операционного автомата, выполняющий запроектированные на него микрооперации.

Итоговая задача проекта операционного автомата на основе анализа необходимым к выполнению микрооперациям, определить и зафиксировать в таблицах соответствующий перечень управляющих сигналов {Y} для инициации соответствующего отклика на исполнительном устройстве. Для этого всё множество устройств, которые входят в состав ОА разбивают на однотипные операционные элементы (регистры, счётчики, дешифраторы, мультиплексоры) и определяют для каждого из них все микрооперации и управляющие сигналы, а также условные сигналы, которые они должны выработать для передачи в управляющий автомат (УА).

Примеры построения операционных элементов и определение их входных и выходных сигналов.

1. Построение операционного элемента на основе регистра:

Возможные операции на регистре:

1). C:= A

2). C:=0

3). C: = R1(1.C)

4). C: = R1(0.C)

5). C: = L1(C.0)

Берём за основу микросхему универсального регистра рис. 8.1.

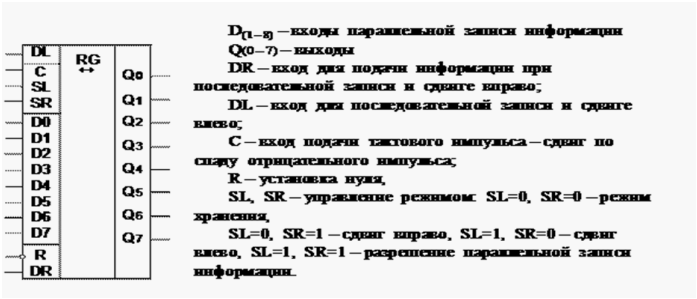


Рис. 8.1. Условно-графическое обозначение универсального регистра.

Таблица функций регистра. Табл. 8.1.

Режимы работы	SR	SL	R	C	Выполняемая микрооперация
Хранение	0	0	1	*	F:=F
Сдвиг влево	0	1	1	1	F:=L1(F.DL)
Сдвиг вправо	1	0	1	1	F:=R1(DR,F)
Запись	1	1	1	1	F:=D
Сброс	*	*	0	*	F:=0

Таблица описания работы операционного элемента на регистре

Табл.8.2.

Y	Микрооперация	S0	S1	R	DL	DR
y1	C:=A	1	1	0	*	*
y2	C:=0	*	*	1	*	*
y3	C:=R1(1.C)	1	0	0	*	1
y4	C:=R1(0.C)	1	0	0	*	0
y5	C:=L1(C.0)	0	1	0	0	*

Синтезируем комбинационную схему. Для каждого из входов регистра определяем булеву функцию, зависящую от значений указанных для микроопераций управляющих сигналов:

SR=y1+y3+y4

SL= y1+y5

R=y2

DL=0

DR=y3

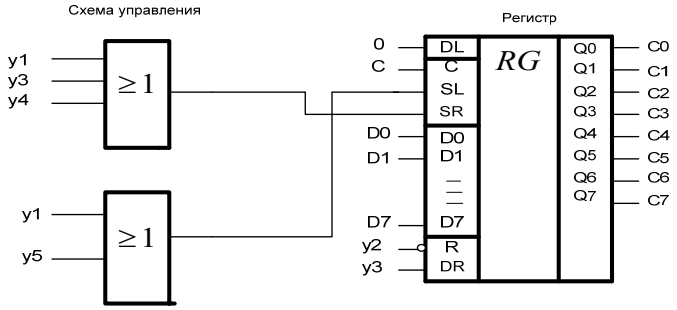


Рис. 8.1 Синтезированная структура операционного элемента на основе регистра.

2. Синтез операционного элемента на основе счётчика.

Перечень режимов работы счётчика иллюстрирует табл. 8.3.

Режимы работы счётчика Табл.8.3.

Режимы работы	S0	S1	R	C	Микрооперации
Запись	1	1	0	1	F:=D
Сложение	1	0	0	1	F:=F+1
Вычитание	0	1	0	1	F:=F-1
Хранение	0	0	0	*	F:=F
Сброс	*	*	1	*	F:=0

Рассматриваемый счётчик является реверсивным и выполняет при:

S0=1 и S1=0 увеличение счёта; S0=0 и S1=1 уменьшение счёта.

Таблица описания операционного элемента на основе счётчика. Табл. 8.4.

Y	Микрооперация	S0	S1	R	+1	Sm	C
y1	F:=A	1	1	0	*	0	1
y2	F:=B	1	1	0	*	1	1
y3	C:=C+1	1	0	0	1	*	1
y4	C:=C-1	0	1	0	1	*	1
y5	C:=0	*	*	1	*	*	*

S0=y1+y2+y3; S1= y1+y2+y4; +1= y3+y4 (сигнал разрешения счёта); Sm=y2 (сигнал выбора направления загрузки кода начала отсчёта).

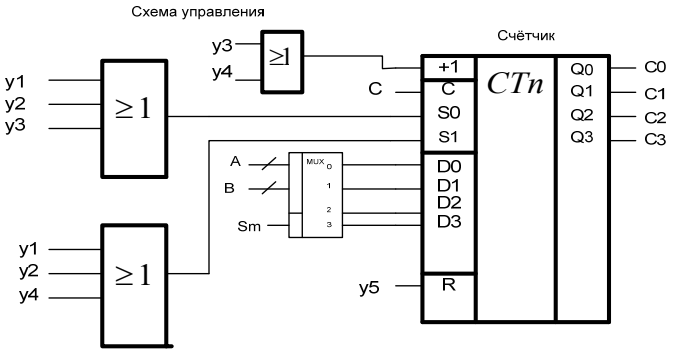


Рис.8.2. Синтезированная структура операционного элемента на основе счётчика.

3. Синтез операционного элемента на основе мультиплексора.

Мультиплексоры используют для подключения нескольких входных шин к одной выходной, либо для выдачи констант на выходную шину. Пусть мультиплексор должен осуществлять следующие передачи информации в выходную шину F.

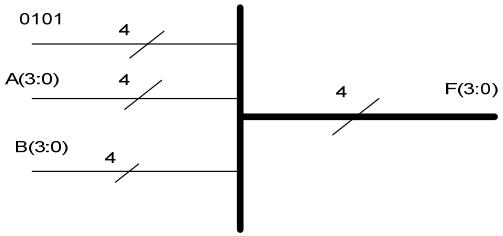


Рис. 8.3. Формализованная схема шинного представления работы мультиплексора.

В этом случае работу мультиплексора можно описать таблицей табл.8.5.

Y	Микрооперация	F3	F2	F1	F0
y1	F:=0101	0	1	0	1
y2	F:=A	A(3)	A(2)	A(1)	A(0)
y3	F:=B	B(3)	B(2)	B(1)	B(0)
В остальных случаях		0	0	0	0

В соответствии с таблицей сигналы на шине А опишутся булевыми функциями:

F(3)= y2*A(3)+y3*B(3) ; F(2)= y2*A(2)+y3*B(2); F(1)=y2*A(1)+y2*B(1);

F(0)=y2*A(0)+y3*B(0)

По булевым функциям построим комбинационную схему мультиплексора рис.8.4

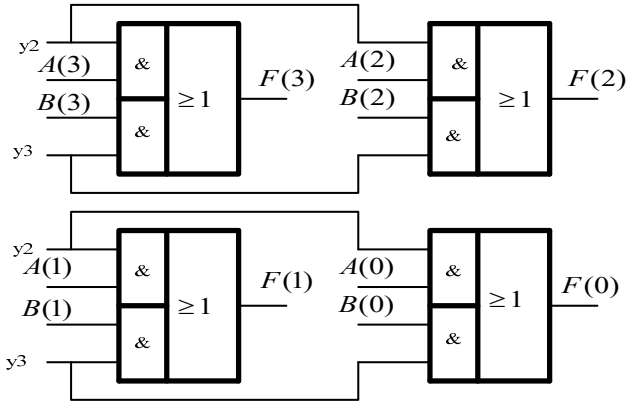


Рис. 8.4. Операционный элемент на основе одноразрядных мультиплексоров.

Лекция 9

Комбинационные схемы и цифровые автоматы

Любое устройство ЭВМ является преобразователем дискретной информации в соответствии с заданным алгоритмом. В общем случае оно имеет n входов и к выходов, на которых присутствуют соответственно входные и выходные сигналы. Каждые входной (выходной) сигнал представляет собой символ (букву) входного(выходного)алфавита. В качестве букв используют обычные двоичные цифры – «0» и «1». (Иногда десятичные).

Преобразование информации в ЭВМ производится электронными устройствами (логическими схемами) двух классов: комбинационными схемами (КС) и цифровыми автоматами (ЦА).

Функциональность КС поясняет схема рис.9.1.

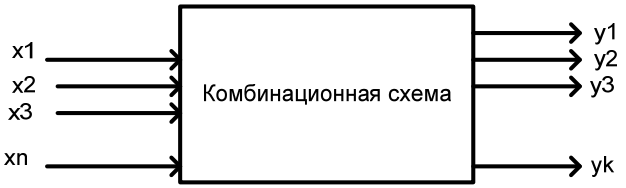


Рис. 9.1. Функциональная структура комбинационной схемы.

Совокупность выходных сигналов (выходное слово y) в дискретный момент времени ti однозначно определяется входными сигналами (входным словом x), поступившим на входы в тот же момент времени. КС не имеет памяти. Закон функционирования КС определяется, если соответствие между словами ее входного и выходного алфавита задано. Это соответствие можно задать таблично – в виде таблиц истинности либо в аналитической форме с использованием булевых функций. Для комбинационной схемы значения её выходных сигналов полностью определяются множеством сигналов на её входах.

У цифрового автомата (ЦА) существует понятие «память цифрового автомата».

ЦА в отличие от КС имеет некоторое конечное число внутренних состояний (Q):

$$Q=\{q0, q1 , \ldots, qr\}$$

В отличии от комбинационной схемы выходные сигналы цифрового автомата определяются не только значениями входных сигналов, но и состоянием памяти цифрового автомата на момент подачи новой комбинации сигналов на его входах. Под воздействием входного слова ЦА переходит из одного состояния в другое и выдает выходное слово.

Выходное слово в ЦА в каждый момент времени ti определяется входным словом, поступившим в этот момент времени на вход автомата, и внутренним состоянием автомата, которое явилось результатом воздействия на ЦА входных слов в предыдущие моменты дискретного времени.

Комбинация входного слова и текущего состояния ЦА в данном такте определяют не только выходное слова, но и то состояние, в которое перейдет ЦА к началу следующего такта.

ЦА обязательно должен содержать память, которая обеспечивает фиксацию состояния. Память ЦА строится на триггерах, регистрах, счетчиках и т.д.

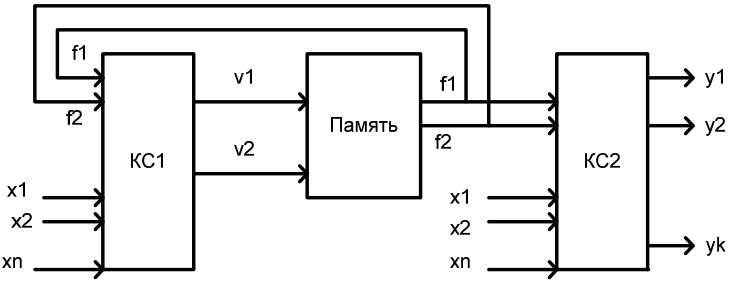


Рис. 9.2. Функциональная структура цифрового автомата.

ЦА состоит из памяти и двух КС— КС1 и КС2. Сигналы состояния ЦА (F) по цепям прямей связи поступают на входы КС2 и по цепям обратной связи на входы КС1. На входы КС1 и КС2 поступают также входные сигналы X.

Выходное слово вырабатывается в КС2. Выходные сигналы КС1 (V) переводят ЦА (его память) в новое состояние.

Одновременность появления сигналов на входах всех схем достигается с помощью тактирующих синхроимпульсов, поступающих в память ЦА.

q0 - начальное состояние

q(t+1)=A[q(t),x(t)] - автомат МИЛИ

y(t)=B[q(t),x(t)]

y(t)=B[q(t)] - автомат МУРА.

Описание управляющего автомата

УА вырабатывает управляющие сигналы Y= y1, …,ум каждый из которых инициирует соответствующую МО Mi , ...,Мм (одну, либо несколько). Выработка управляющих сигналов производится в соответствии со значениями осведомительных сигналов X = Xi, ... , Xl

Для описания работы УА используется закодированный граф МП (граф схема МП).

Закодированный граф МП получается из содержательного графа МП путём замены МО, указанных в операторных вершинах, идентификаторами управляющих сигналов, инициирующих эти МО, и замены логических условий, содержащихся в условных вершинах, соответствующими идентификаторами осведомительных сигналов.

В управляющих автоматах увеличение количества вершин в закодированном графе МП влечёт за собой увеличение затрат оборудования в УА, поэтому при проектировании операционного устройства, реализующего ряд МП Mi , ... , Mc, выгоднее синтезировать не G различных УА –для каждой МП свой, - а единый обобщённый УА, реализующий все МП. Обобщённый УА строится на основе объединённого закодированного графа Г, объединяющего в себе закодированные графы Гi, ... ,Гg отдельных МП. Объединённый граф Г в общем случае имеет меньше вершин, чем их суммарное количество в G

графах, так как при объединении отдельные операторные и условные вершины обычно сливаются.

Построение управляющих блоков на основе автоматов МИЛИ и МУРА.

Управляющий блок (УБ) предназначен для организации работы операционного блока. На вход управляющего блока подают код операции и признаки/оповещающие сигналы. Управляющий блок в итоге будет представлен структурой управляющего автомата.

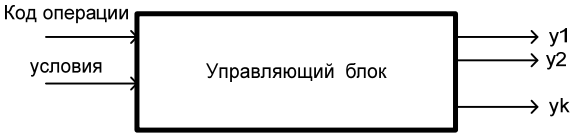


Рис. 9.3. Входящие и выходящие сигналы управляющего блока.

На выходе УБ снимается последовательность управляющих сигналов y1- yn. Для каждой операций вырабатывается своя последовательность, определяемая соответствующей микропрограммой. В общем случае УБ можно рассматривать как цифровой автомат. Для которого:

X={x1, x2 ,…, xl} - множество входных сигналов, которое определяется кодом операции и множеством признаков - логических условий и осведомительных сигналов.

Y={y1, y2 ,…, ym} - множество выходных сигналов, - управляющих сигналов, инициирующих выполнение микроопераций в операционном блоке.

При этом закон функционирования автомата, задающий порядок преобразования входной последовательности X(0),X(1)…X(k) в выходную последовательность Y(0),Y(1)…Y(k) определяется микропрограммой.

Реализация автомата по схеме МИЛИ.

Пусть, например, микропрограмма задана в виде графа. Для ее реализации используем автомат МИЛИ с законом функционирования:

A(t+1)=f1[A(t),X(t)]; Определение номера (двоичного кода) вершины перехода; Y(t)=f2[A(t),X(t)]. Определение управляющего сигнала

микрооперации. Пример закодированного графа с помеченными вершинами представлен на Рис. 9.4.

Где A={a1,…an} – множество состояний автомата.

a1- начальное состояние автомата (начальная вершина).

Рассматриваемая микропрограмма оперирует с четырьмя логическими условиями: X={x1, x2, x3, x4} и пятью микрооперациями

Y={y1, y2, y3, y4, y5}

Поэтому автомат должен иметь четыре входа / x1, x2, x3, x4/ и пять выходов / y1, y2, y3, y4, y5/. Необходимый набор состояний автомата определим методом разметки графа микропрограммы, которая производится в следующем порядке: I. Символом a1 отмечают вход первой вершины, следующей за на¬чальной, а также вход конечной вершины.

2. Входы вершин, следующих за операторными, отмечаются символами a2, a3,…

3. Входы двух различных вершин, за исключением конечной не могут быть отмечены одинаковыми символами.

4. При этом предполагается, что в начальном состоянии a1 автомат формирует нулевые значения на всех выходах / y1, y2, y3, y4, y5/.

После окончания работы автомат возвращается в это же состояние.. Вход вершины может отмечаться только одним символом. Например из вершины a1 в вершину a4 есть два пути: через вершину a2 при x1=0 и x2=0 и через вершину a3 при x1=1 и x2=0.

В нерабочем состоянии /микропрограмма не выполняется/ автомат находится в начальном состоянии a1.

При запуске /инициирование микропрограммы/ он сохраняет это состояние в течении первого машинного такта /и вырабатывает в первом такте выходные сигналы/.

После окончания первого такта он переключается в очередное состояние и в автомате начинает выполняться соответствующий набор микроопераций.

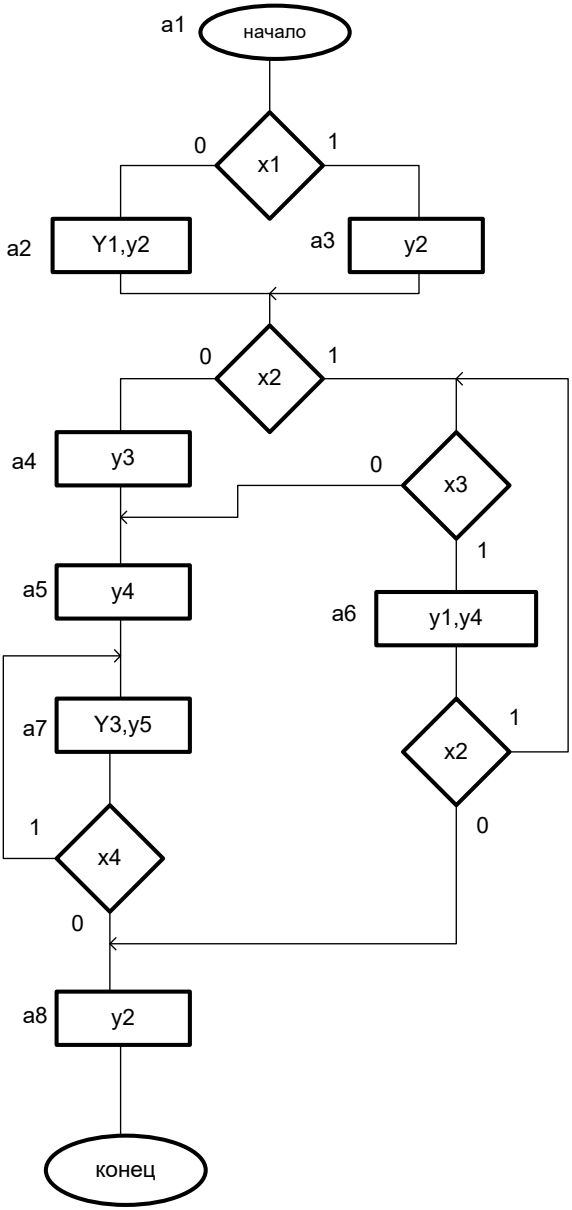


Рис. 9.4. Закодированный граф с помеченными вершинами.

Момент окончания микропрограммы отмечается возвратом автомата в состояние a1. (вершина конец).

Лекция 10

Синтез управляющих автоматов по схеме Мура

Основное назначение управляющих автоматов.

Управляющий автомат взаимосвязан с операционным автоматом.

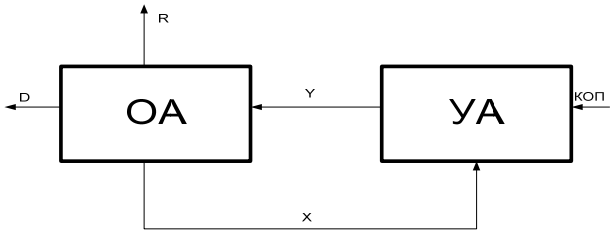


Рис. 10.1. Структура связи операционного и управляющего автоматов.

Входными данными для управляющего автомата является множество сигналов условий {X}, сигналы, определяющие код операции. Управляющий автомат выдаёт сигналы управления {Y}, которые поступают на вход операционного автомата.

Наибольшее распространение для построения управляющих автоматов получили два типа автоматов: цифровой автомат Мили и цифровой автомат Мура.

Особенностью автомата Мили является то, что функция выходов является двухаргументной и символ в выходном канале y(t) обнаруживается только при наличии символа во входном канале x(t). Функциональная схема не отличается от схемы абстрактного автомата. Автомат Мили рассмотрен в лекции 9.

Зависимость выходного сигнала только от состояния представлена в автоматах типа Мура(англ. Moore machine). В автомате Мура функция выходов определяет значение выходного символа только по одному аргументу — состоянию автомата. Эту функцию называют также функцией меток, так как она каждому состоянию автомата ставит метку на выходе.

Конечным детерминированным автоматом типа Мура называется совокупность пяти объектов:

где S, X, Y и δ — соответствуют определению автомата типа Мили, а μ является отображением вида: $\mu : S \rightarrow Y$,

с зависимостью состояний и выходных сигналов во времени уравнением:

$$y(t) = \mu(s(t)), t \in T$$

Особенностью автомата Мура является то, что символ y(t) в выходном канале существует все время, пока автомат находится в состоянии s(t).

Функциональную структуру автомата Мура поясняет рис. 10.2.

Символами б и μ обозначены входная и выходная комбинационные схемы, которые осуществляют логику определения кода, определяющего состояния памяти автомата, выходная комбинационная схема формирует соответствующий состоянию памяти один или несколько управляющих сигналов Y.

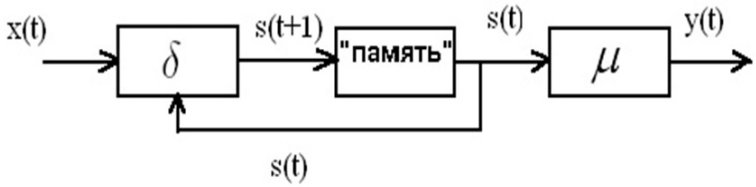


Рис. 10.2. Функциональная схема управляющего автомата по схеме Мура.

Исходные данные для автомата Мура:

$$Y = \{y1, y2, \ldots\ldots y n\}$$

$$X = \{x1, x2, \ldots\ldots x n\}$$

$$A = \{a1, a2, \ldots\ldots a_{n-1}\}$$

Это сигналы управления Y, сигналы условий X, коды номеров A вершин закодированного графа (состояния памяти управляющего автомата)

Все эти данные определены в функциональной микропрограмме (ФМП).

Следующим проектным этапом является формализация графа микроопераций с помощью закодированного графа, в котором каждая микрооперация (МО)

заполняется соответственно управляющим сигналом Y, логические условия - осведомительным сигналом X.

Пример перехода от содержательного графа к закодированному графу иллюстрирует рис.10.3

Для этого надо произвести:

- осуществить построение таблицы микроопераций и логических условий и построение графа функциональной микропрограммы;

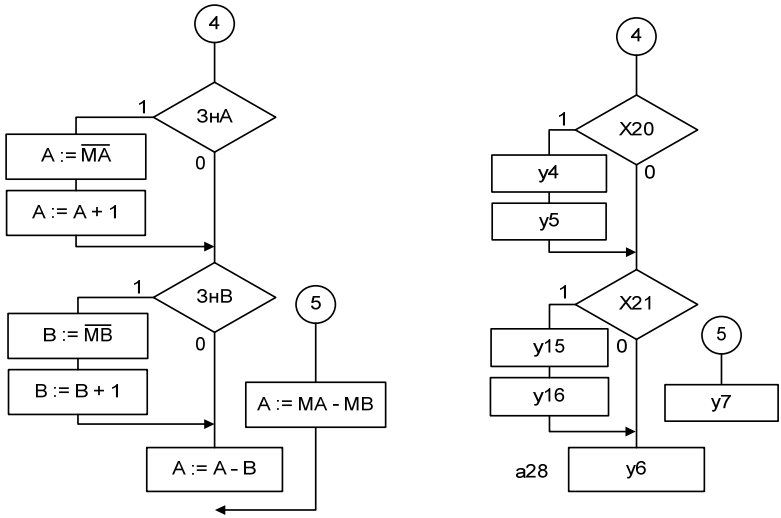


Рис. 10.3. Переход от содержательного графа функциональной микропрограммы к закодированному графу.

Переход осуществляется в несколько этапов:

- построение закодированного графа ФМП путём замены каждой МО управляющим сигналом Y, каждого ЛУ осведомительным сигналом X;

- на закодированном графе пометить только функциональные вершины индексами a0,a1…an-1.

- завершить построение закодированного графа для синтеза управляющего автомата Мура:

каждому состоянию поставить в соответствие вершину графа;

каждому переходу поставить в соответствие дугу графа.

Последующий этап проектирования управляющего автомата состоит в построение списка переходов в табличной форме. В таблице каждая дуга графа соответствует строке таблицы с указанием условия X, сигнала управления Y, и перехода от начального состояния в конечное (а нач. и а конеч.)

Рассмотрим алгоритм операции умножения. На рис. 10.4 и рис.10.5 представлен граф ФМП и соответствующей ей закодированный граф.

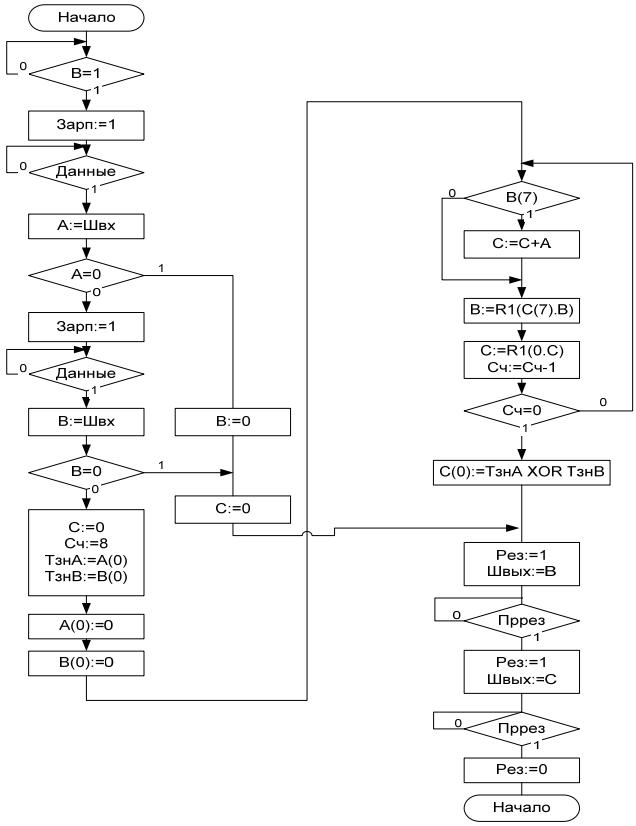


Рис. 10.4. Граф формализованной микропрограммы для операции умножения.

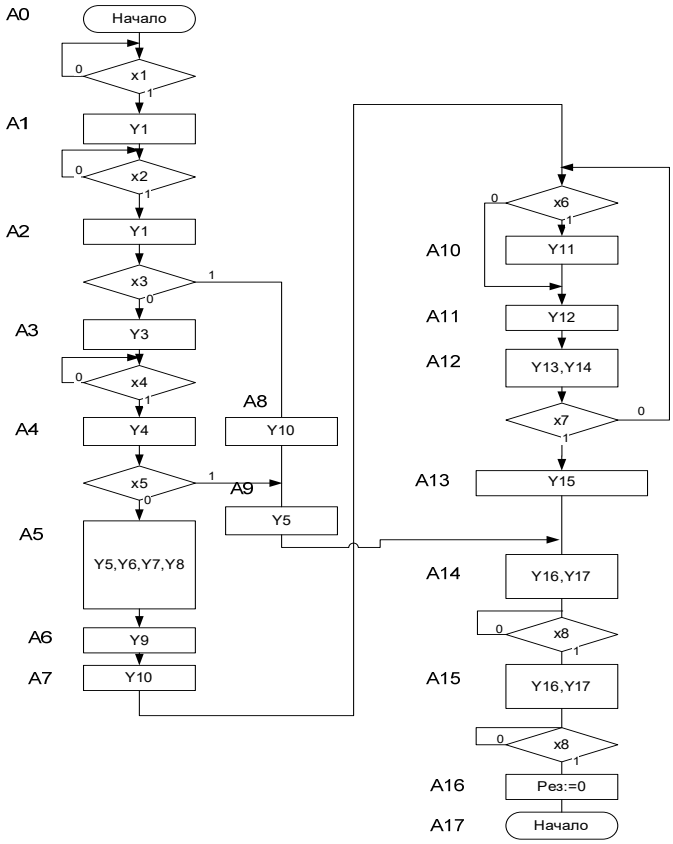


Рис. 10.5. Закодированный граф для операции умножения.

Согласно данным закодированного графа, строится таблица переходов, в которой отображается вся возможная последовательность переходов в графе, каждая строка содержит нумерацию исходящей вершины и вершины перехода, при этом учитывается значение одной или нескольких условных переменных. Если имеется одно ветвление из одной вершине в две другие то отмечается, какое значение должно быть у определяющей этот переход условной переменной: если 1, то пишется прямое значение условной переменной, если 0, то инверсное обозначение переменной (обозначено! X).

Соответствующие исходящей вершине сигналы условий обозначаются идентификатором Y с порядковым номером. Эти сигналы при данном значении кода исходящей вершины ложны быть равны 1. Логика получения сигналов управления Y также закладывается в специализированные таблицы для прошивки ПЛМ.

Текущее значение кода следующей вершины передается с выходов КС1 в составе ПЛМ на многоразрядный регистр параллельного занесения, тактируемый сигналом синхроимпульса. Данный код вершины не меняется в регистре до следующего такта. Выход данного регистра подключается на вход комбинационной схемы КС1 в составе ПЛМ для получения нового кода следующей вершины, зависящего и от значения условного сигнала, поступившего из операционного автомата (ОА).

Необходимая разрядность регистра памяти зависит от количества разрядов, необходимых для отображения самого старшего номера вершины в двоичном коде и определяется по формуле:

$$N =]\log P[$$

где: N- требуемая разрядность регистра, P-максимальный номер вершины в закодированном графе.

Пример составления таблицы переходов иллюстрирует табл. 10.1.

Таблица переходов Табл. 10.1

№	Исход. сост.	Код	Конечн. состояние	Код	Условие X	Сигнал управления Y	Осведоми-тельные сигналы
1	a0	00000	a0	00000	!x1	-	-
2	a0	00000	a1	00001	x1	-	D0
3	a1	00001	a1	00001	!x2	y1	D0
4	a1	00001	a2	00010	x2	y1	D1
5	a2	00010	a3	00011	!x3	y1	D1, D0
6	a2	00010	a8	01000	x3	y1	D3
7	a3	00011	a3	00011	!x4	y3	D1, D0
8	a3	00011	a4	00100	x4	y3	D2
9	a4	00100	a5	00101	!x5	y4	D2, D0

(Показаны только первые 9 строк таблицы)

Таблица переходов служит для проектирования логики двух комбинационных схем:

комбинационная схема КС1 строится на основании булевых функций для каждого разряда кода следующей вершины перехода.

Сигнал возбуждения – это обозначение того разряда в коде следующей вершины, который должен быть равен 1. Младший разряд кодируется D0, следующий D1 и т.д.

Из данной таблицы можно составить булевы выражения для каждого из сигналов возбуждения. Например:

$$D0 = a0 * x1 + a1 * \overline{x2} + a2 * \overline{x3} + + a4 * \overline{x5} + a4 * x5 +$$

$$a7 * \overline{x6} + a15 * \overline{x8}$$

В дальнейшем эти булевы выражения для сигналов возбуждения лягут в основу специализированных таблиц для прошивки программируемых логических матриц, которые позволяют воспроизводить СДНФ функций.

Лекция 11

Применение программируемых логический матриц (ПЛМ) в структуре управляющего автомата

В материалах лекции 10 обосновано наличие в составе управляющего автомата двух комбинационных схем различного назначения.

Одним из наиболее перспективных способов реализации комбинированных схем ЭВМ является использование технологии БИС, что существенно повышает надежность, а также уменьшает стоимость, габариты и массу. Для реализации различных систем булевых функций, т.е. для построения различных комбинационных схем на основе БИС постоянной структуры, разработаны и используются специальные регулярные структуры – программируемые логические матрицы (ПЛМ).

Рис. 11.1. Структура УА на основе ПЛМ и регистра.

В основе построения ПЛМ лежит принцип вычисления дизъюнктивной формы булевой функции по таблице истинности. Для их синтеза можно использовать исходные данные таблицы переходов в качестве таблице истинности для получения былевых выражений для сигналов возбуждения и управляющих выходных сигналов управляющего автомата.

В самом общем случае упрощённую структуру ПЛМ можно представить из состава двух матриц: матрицы «И» и матрицы «ИЛИ» рис. 11.2.

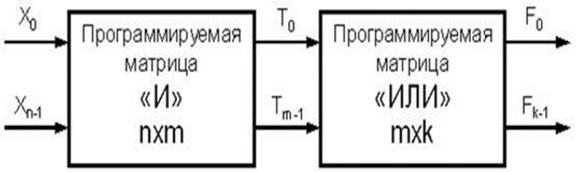


Рис. 11.2. Упрощённая структура ПЛМ.

На рис. 11.3 представлена структура ПЛМ с обозначением плавких перемычек.

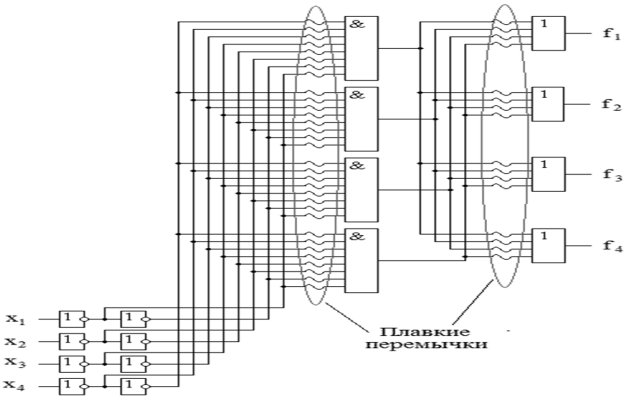


Рис. 11.3. ПЛМ с плавкими перемычками, используемыми для прошивки булевых функций.

Для того чтобы отличать ПЛМ от ПЗУ при изображении принципиальных электрических схем, в среднем поле условного графического обозначения пишется PLM рис. 11.3.

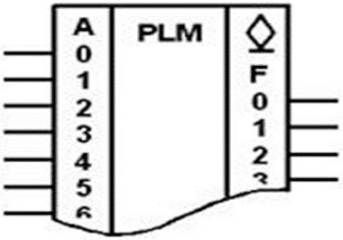


Рис. 11.3. Условно-графическое обозначение ПЛМ.

ПЛМ имеют 16 входов,8 выходов и от 48 до 68 конъюнкций.

Для определения необходимого количества (Q) ПЛМ, для реализации управляющего автомата, имеющего К разрядов регистра и m – число управляющих сигналов y, формируемых в автомате используется формула:

Необходимое количество ПЛМ (Q) для УА определяется формулой:

$$Q =](K + m)/8[$$

где К – количество разрядов регистра памяти УА, m – количество управляющих сигналов Y.

ПЛМ изготавливаются с полным набором электрических элементов, обеспечивающих однонаправленную передачу сигналов: в матрице «И» из любой вертикальной цепи в любую горизонтальную, а в матрице «ИЛИ» - из любой горизонтальной – в вертикальную. Такого рода соединения обеспечиваются за счет диодов или транзисторов.

Информация в ПЛМ заносится путем установления соединений между горизонтальными и вертикальными цепями матриц. Этот процесс называется программированием матриц. По способу программирования различают ПЛМ с масочным и электрическим программированием, а также перепрограммируемыми ПЛМ.

В ПЛМ первого типа информация заносится путем подключения диодов или транзисторов к цепям матриц, для чего производится металлизация соответствующих участков матриц, выполняемая через маску (шаблон). В ПЛМ с электрическим программированием необходимая информация записывается путем выжигания перемычки или установления соединения путем подачи токового импульса большой амплитуды (20 – 200 мА) в соответствующую цепь матрицы. Такое программирование осуществляется самим пользователем на специальном оборудовании.

Репрограммируемые ПЛМ позволяют многократно перезаписывать информацию. При этом стирание информации производится обычно ультрафиолетовым облучением, а запись – электрическим током.

Формирование данных для проектирования управляющего автомата.

При проектировании УА решается задача определения разрядности регистра хранения (формула была дана в предыдущих лекциях) и после определения нужного количества ПЛМ, создание таблиц прошивки ПЛМ.

Порядок подготовки таблиц для программирования ПЛМ.

Распределить управляющие сигналы и сигналы возбуждения между всеми ПЛМ (сигналы возбуждения D1-Dn подавать на входы регистра состояний. Каждый из этих сигналов может быть закреплён только за одной ПЛМ.

Выполнить (виртуальное) программирование ПЛМ. Для каждой ПЛМ составить таблицу соединений.

В каждой таблице указать входы, выходы и строки. На входы F1 – Fk всех ПЛМ подключить выходы регистра состояний (старшие слева- младшие справа). На остальные входы ПЛМ подключить осведомительные сигналы (условий), используемые в данной ПЛМ. Неиспользуемые входы ПЛМ не указывать. Число входов не должно превышать 16. Число выходов в каждой ПЛМ не должно превышать 8.

В каждой строке входов прямое значение переменной кодировать единицей, инверсное – нулём, а безразличное звёздочкой.

На выходах единицей обозначать необходимость использовать данную конъюнкцию (строку) в булевской функции, описывающей соответствующую выходную переменную (управляющий сигнал Y или сигнал возбуждения D).

Число строк в каждой матрице не должно превышать 68.

Для упрощения понимания создания таблиц прошивки ПЛМ, будем делить ПЛМ на два множества: с одних ПЛМ будем получать сигналы возбуждения, определяющие код вершины перехода, с других ПЛМ будем получать управляющие сигналы Y, зависящие от кода исходящей вершины. Для удобства составления таблиц, код вершины перехода обозначаем D0, D1…Dn, а для кода исходящей вершины используем, например, обозначения F0, F1…Fn. Ниже приводятся (начальные фрагменты) этих двух таблиц для

программирования ПЛМ. Ранее было принято, что разрядность регистра хранения К=5.

Таблица для ПЛМ сигналов возбуждения. Табл.11.1

Входные данные														Выходные данные				
№	F4	F3	F2	F1	F0	X1	X2	X3	X4	X5	X6	X7	X8	D4	D3	D2	D1	D0
1	0	0	0	0	0	1	*	*	*	*	*	*	*	0	0	0	0	1
2	0	0	0	0	1	*	0	*	*	*	*	*	*	0	0	0	0	1
3	0	0	0	0	1	*	1	*	*	*	*	*	*	0	0	0	1	0
4	0	0	0	1	0	*	*	0	*	*	*	*	*	0	0	0	1	1
5	0	0	0	1	0	*	*	1	*	*	*	*	*	0	1	0	0	0
6	0	0	0	1	1	*	*	*	0	*	*	*	*	0	0	0	1	1

Таблица для ПЛМ управляющих сигналов. Табл.11.2.

Входные данные						Выходные данные							
№	F4	F3	F2	F1	F0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8
1	0	0	0	0	1	1							
2	0	0	0	1	0	1							
3	0	0	0	1	1			1					
4	0	0	1	0	0				1				
5	0	0	1	0	1					1	1	1	1
6	0	0	1	1	0								

Таким образом, ПЛМ представляет собой универсальную структуру, которая путем программирования может быть приспособлена для реализации широкого класса булевых функций.

ПЛМ с К входами, Р меж матричными связями и n выходами может реализовать любую систему, содержащую не более n функций, не более чем от К переменных, если система функций может быть определена не более чем через Р различных конъюнктивных термов.

Пример микросхемы ПЛМ: КР556РТ1 – ПЛМ (16 входов, 8 выходов, 48 термов).

Лекция 12

Микропрограммные устройства управления (МПУУ)

Классификация МПУУ:

1. МПУУ схемотехнического типа.

2. Микропрограммные МПУУ.

Первые обладают жёсткой логикой, быстродействием, но их трудно модернизировать. Пример первого был рассмотрен в лекциях 10 и 11. Вторые – строятся на основе микрокоманд, но легко модернизируются.

Микропрограммные устройства управления с хранимой в памяти логикой

Принцип микропрограммного управления ходом вычислительного процесса были разработаны Уилксом еще в 1951 году.

Функционирование управляющего автомата определяется:

1.Множеством входных осведомительных сигналов

X = {X1, …,Xm},

отражающих состояние операционного автомата.

2. Множеством выходных (управляющих) сигналов

Y = {y1, …., yк} инициирующих микрооперации, реализуемых операционным автоматом.

3. Графом микропрограммы, задающим порядок следования управляющих сигналов Y в зависимости от значений осведомительных сигналов X.

Функционирование управляющего автомата при этом сводится к генерированию последовательности управляющих сигналов Y, предписанной микропрограммой и соответствующей последовательности осведомительных сигналов X.

Рассмотрим построение управляющего автомата на основе принципа программного управления, использующего операционно-адресную структуру управляющих слов.

Управляющее слово определяет порядок функционирования МПУУ в течение одного такта и называется микрокомандой.

Совокупность микрокоманд (МК) образует массив микрокоманд МК[О:Р] отдельные МК в котором выделяются посредством адреса, равного номеру 0,1, …., Р элемента массива МК.

Микрокоманда содержит информацию о микрооперациях, которые должны выполняться в данном такте, и информацию об адресе следующей МК.

Определим простейшую структуру управляющих слов, достаточную для представления МК.

Пусть множество Y содержит М – микроопераций, которым присвоим номера 1,2,…,М. Это множество дополняется обычно еще одной микрооперацией – (М+1)-й, которая используется для завершения микропрограммы и перехода к следующей микропрограмме.

Типовая функциональная структура МПУУ

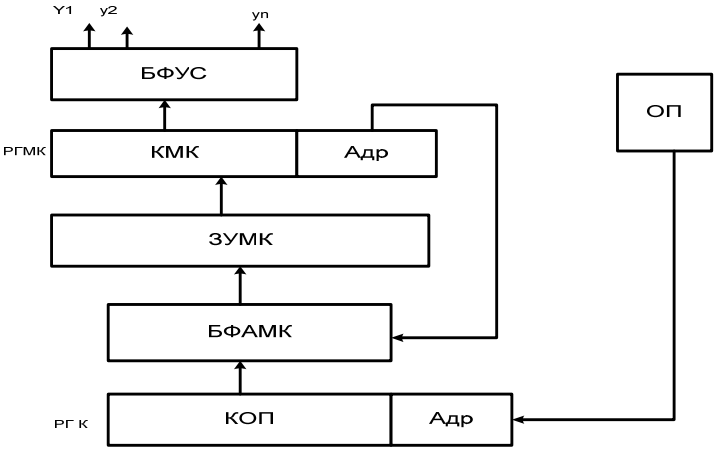


Рис.12.1 Обобщённая типовая функциональная структура МПУУ.

Использованы следующие обозначения:

БФУС – блок формирования управляющих сигналов;

КМК – код микрокоманды;

Адр – адрес перехода для извлечения следующей микрокоманды;

ЗУМК – запоминающее устройство микрокоманд;

БФАМК – блок формирования адреса микрокоманды;

КОП – код операции;

ОП – оперативная память.

По способу кодирования управляющих сигналов МПУУ подразделяются на:

- МПУУ с горизонтальным микропрограммированием;
- МПУУ с вертикальным микропрограммированием;
- МПУУ со смешанным микропрограммированием.

При горизонтальном способе микропрограммирования - каждый бит содержит управляющий сигнал.

Появление управляющего сигнала в соответствующем такте будем обозначать кодом «1», отсутствие – «0».

При таком способе операционная часть МК (КМК) содержит М разрядов,

где, М – общее число микроопераций, а сама МК состоит из операционной части и адресной.

Достоинства горизонтального микропрограммирования:

-возможность одновременного выполнения в одном такте любого набора из М микроопераций;

- простота формирования управляющих сигналов.

Недостаток – требуется большая длина МК, а следовательно, и большой объем ЗУМК.

При вертикальном способе микропрограммирования в поле КМК содержится только один управляющий сигнал.

При вертикальном микропрограммировании микрооперация определяется не состоянием одного из разрядов МК, а двоичным кодом, содержащимся в операционной части МК.

При этом количество разрядов операционной части МК для кодирования М МО определяется из соотношения: R=]log2 M[

Достоинством вертикального микропрограммирования является:

- минимальная длина МК;
- малая разрядность МК;
- простота дешифрации МК;

- малое количество оборудования (ЗУМК).

Недостатки:

- по каждой МК может быть выполнена только одна микрооперация;
- малое быстродействие;
- большое количество МК в микропрограмме.

В связи с этим недостатком в настоящее время наибольшее распространение получило смешанное микропрограммирование, которое сочетает принципы горизонтального и вертикального микропрограммирования.

При смешанном микропрограммировании множество всех микроопераций Y={y1,….,ym} разбивается на К подмножеств: $Y=\bigcup Y_i$

При этом микрооперации внутри каждого из подмножеств кодируются вертикальным методом.

Разрядность поля КМК = количеству микроопераций.

Каждый разряд закрепляется за своей микрооперацией. Блок БФУС вырождается, увеличивая быстродействие. За один такт формируется один управляющий сигнал. Есть ещё некоторые классификационные отличия:

По способу использования машинного такта:

- однофазные 1 МО- 1 такт
- многофазные: Такт разбивается на микротакты : 1МО – 1 микротакт.

По способу кодирования микроопераций:

- прямое кодирование. Поле микрокоманды несёт фиксированные функции;
- косвенное кодирование. Характеризуется наличием дополнительных полей, которые меняют смысл основных полей.

По способу адресации микрокоманд:

- с принудительной адресацией;
- с естественной адресацией.

Особенность реализации технологии принудительной адресации.

В общем случае принудительная адресация состоит в том, что в каждой МК указываются всевозможные адреса следующих МК. Структура микрокоманды выглядит: рис. 12.2.

A=A0, если x=0, A=A1, если x=1.

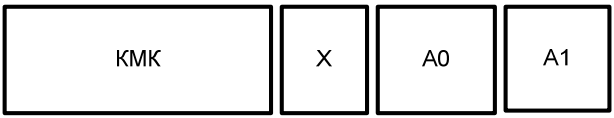


Рис. 12.2. Структура полей микрокоманды при принудительной адресации.

Адрес следующей МК может задаваться, безусловно, т.е. независимо от значений осведомительных сигналов, или выбираться по условию, определяемому текущими значениями осведомительных сигналов.

Для МК можно использовать один адрес, вместо мультиплексора – инкрементор, поля микрокоманды имеют вид рис. 12.3.

A=A0, если x=0, A=A0+1, если x=1.

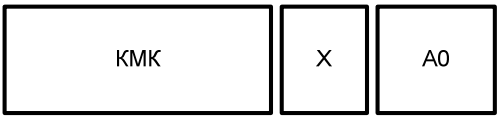


Рис. 12.3. Структура полей микрокоманды при принудительной адресации с одним адресным полем.

Особенность реализации технологии естественной адресации.

При естественной адресации адрес, следующей МК принимается равным увеличенному на единицу адресу предыдущей МК.

Если A адрес исполняемой МК, то следующая МК выбирается по адресу (A+1).

При этом методе процесс адресации реализуется счетчиком адреса МК, состояние которого увеличивается на 1 после чтения очередной МК.

В этом случае функциональные МК могут содержать только операционную часть, представленную полями Y1, … ,Yk.

После выполнения МК с адресом A может потребоваться переход к МК с адресом B¹A+1. Переход может быть безусловным и условным по условию X:

Xx= 0, переход по адресу (A+1)

Xx = 1 , переход по адресу B

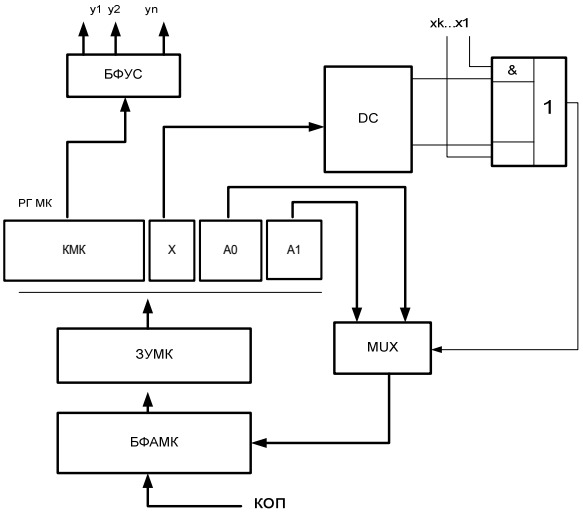


Рис. 12.4. Структура МПУУ с принудительной адресацией.

Для реализации условных переходов в МК вводится адресная часть, состоящая из полей Х и В. Рис.12.5.

Для этой цели используются МК двух типов: операционные и управляющие**.**

Управляющие МК используются для изменения естественного порядка следования МК при выполнении условных и безусловных переходов. Управляющая МК содержит поле Х, определяющее номер условия, и поле В, определяющее адрес следующей МК.

Если Xx≠0, то адрес следующей МК всегда равен В.

Если Xx=0, то переход по адресу (А+1). Рис. 12.5.

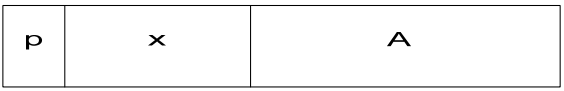


Рис.12.5 Структура управляющей части микрокоманды.

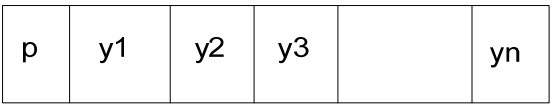


Рис. 12.6. Операционная часть микрокоманды

Для определения типа МК используют одноразрядное поле признака Р.
(Р=0 операционная МК, Р=1 управляющая МК):

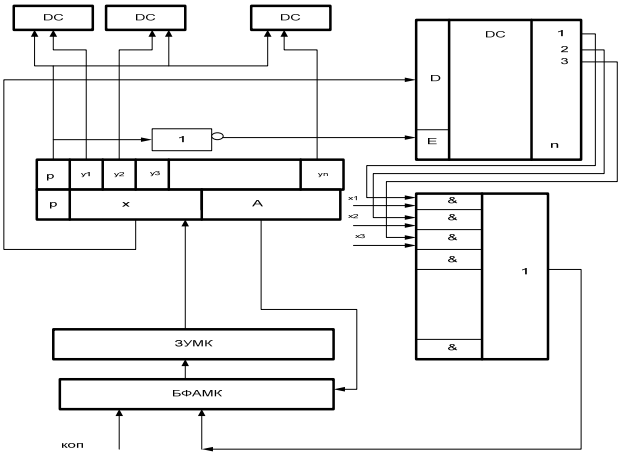


Рис. 12.6. Структура МПУУ с естественной адресацией.

Методы повышения быстродействия МПУУ

Время формирования управляющих сигналов складывается из:

- времени формирования адреса следующей МК;
- времени обращения к ЗУМК;
- времени дешифрирования операционной части МК.

Основная доля времени приходится на чтение из ЗУМК.

Увеличение быстродействия можно получить за счёт сокращения этого времени, либо за счёт уменьшения количеств обращений к ЗУМК.

На практике используют два основных метода увеличения быстродействия МПУУ:

- параллельная выборка МК;
- опережающая выборка МК.

При параллельной выборке МК длина слова, хранимого в ЗУМК, берётся равной длине К микрокоманд. За одно обращение к ЗУМК считывается

одновременно К микрокоманд, которые затем обрабатываются последовательно.

Рассмотрим структуру МПУУ с параллельной выборкой четырёх МК и естественной адресацией.

Адресация МК осуществляется счётчиком адреса, р старших разрядов которого определяет адрес УС в ЗУМК, а два младших разряда адрес МК в данном слове.

При опережающей выборке такт работы вычислительного устройства это сумма времён: $$T_{оу} = T_{уа} + T_{оа}$$

Для уменьшения длительности такта можно начинать выборку следующей МК до момента окончания микрооперации, т.е. выбирать МК с опережением времени.

В результате этого процесс выборки следующей МК совмещается с процессом реализации предыдущей МК.

Такт работы МПУУ будет: $$T = \max(T_{уа}, T_{оа}),$$

где: $$T < T_{уа} + T_{оа}$$

При этом методе адрес следующей МК назначается только предположительно, так как он может зависеть от значений логических условий, вычисляемых выполняемой МК.

Глоссарий. Список сокращений и обозначений

АВС – ассоциативная вычислительная система

АЗУ – ассоциативное запоминающее устройство

АЛУ – арифметико–логическое устройство

АП – ассоциативный процессор

АСОИУ – автоматизированные системы обработки информации управления

АЦП – аналогово-цифровой преобразователь

БИС – большая интегральная схема

БФ – булева функция

БФУС – блок формирования управляющих сигналов

ВС – вычислительные средства

ДНФ – дизъюнктивно нормальная форма

ЗУ – запоминающее устройство

ЗУМК – запоминающее устройство микрокоманд

ЗЭ – запоминающий элемент

ИС – интегральная схема

КНФ – конъюнктивно – нормальная форма

КОП – код операции

КП – кэш - память

КС – комбинационная схема

ЛЗС – линия записи - считывания

МДП – металл – диэлектрик – полупроводник

МКМД - множественный поток команд множественный поток данных

МКОД - множественный поток команд одиночный поток данных

МН – магнитный носитель

МО - микрооперация

МОП – металл – окисел – полупроводник

МП – микропрограмма

МФУ – многофункциональное устройство

НЖМД – накопитель на жёстком магнитном диске

ОА – операционный автомат

ОЗУ – оперативное запоминающее устройство

ОКМД - одиночный поток команд множественный поток данных

ОКОД - одиночный поток команд одиночный поток данных

ОП – оперативная память

ОУ – операционное устройство

ПАВ – поверхностно активная волна

ПЗС – прибор с зарядовой связью

ПЗУ – постоянное запоминающее устройство

ПЛМ – программируемая логическая матрица

ППУ – принцип программного управления

ПЭ – процессорный элемент

ПЭП – пьеза электрический преобразователь

РОН – регистр общего назначения

СДНФ – совершенная дизъюнктивно – нормальная форма

СКНФ – совершенная конъюнктивно – нормальная форма

ТЭЗ – типовой элемент замены

УА – управляющий автомат

УБ – управляющий блок

УУ – устройство управления

ФМП – функциональная микропрограмма

ФЭУ – фотоэлектронный умножитель

ЦА – цифровой автомат

ЦАП – цифро-аналоговый преобразователь

ЦВТ – цифровая вычислительная техника

ЦП – центральный процессор

ЭДС – электродвижущая сила

ЯФМП – язык формализованного микропрограммирования

FIFO – первым пришёл, первым обслужился

Литература

1. Максимов Н.В., Попов И.И., Партыка Т.П. Архитектура ЭВМ и вычислительных систем. Издательство Форум, Инфра-М, 2016, 512 с.
2. Майоров С.А., Новиков Г.И., Структура электронных вычислительных машин. Л.: Машиностроение, 1979. 384 с.
3. Новожилов, О.П. Архитектура ЭВМ и систем. В 2 частях. Ч. 1: учебное пособие для вузов / О.П. Новожилов. – Москва : Издательство Юрайт, 2023. - 276 с. –(Высшее образование).
4. Новожилов, О.П. Архитектура ЭВМ и систем. В 2 частях. Ч. 2: учебное пособие для вузов / О.П. Новожилов. – Москва : Издательство Юрайт, 2023. - 246 с. –(Высшее образование).
5. Орлов С.А., Цилькер Б.Я. Организация ЭВМ и систем: Учебник для вузов. 3-е изд. Стандарт третьего поколения. – СПб.: Питер, 2015. -688 с.: ил. – (Серия «Учебник для вузов»).
6. Сенкевич А.В. Архитектура ЭВМ и вычислительные системы. Издательство Academia, 2014, 240 с.
7. Таненбаум Э. Архитектура компьютера, 5-е изд. СПб.: Питер, 2007, 843 с.
8. Хорошевский В.Г. Архитектура вычислительных систем. М.: Изд-во МГТУ им. Н.Э. Баумана, 2005. 512 с.

Приложение

Перечень контрольных вопросов по лекциям модуля 1.

1. Какие основные структурные блоки ЭВМ обоснованы в трудах Фон-Неймана?
2. Перечислите основные признаки, по которым ЭВМ соотносят к тому или иному поколению?
3. В каких единицах измеряется ёмкость памяти ЭВМ?
4. Перечислите известные вам виды оценки производительности ЭВМ?
5. В чём различие четырёх структур вычислительных систем, предложенных в классификации Флина?
6. Где могут находится операнды при выполнении процессором безадресной команды?
7. Откуда извлекается операнд в командах с непосредственной адресацией?
8. Приведите аппаратные примеры устройств из множества операционных устройств ЭВМ.
9. Сформулируйте основное назначение операционного и управляющего автоматов.
10. Какими сигналами обменивается операционный автомат?
11. Какими сигналами обменивается управляющий автомат?
12. Поясните, чем отличаются совместимые и несовместимые микрооперации.
13. Может ли функциональная вершина графа функциональной микропрограммы содержать несколько микроопераций и каких?
14. Какое действие вызывает управляющий сигнал у, поступающий из управляющего автомата в операционный автомат?
15. Все ли микрооперации надо учитывать при описании конкретного операционного элемента, в которых он задействован?
16. В чём принципиальное отличие комбинационной схемы от цифрового автомата?
17. В чём состоит различие автомата Мили и автомата Мура?
18. Какое устройство может являться памятью автомата Мура?

19. Напишите выражение для определения разрядности регистра в составе автомата Мура.

20. Поясните терминологию «сигнал возбуждения», используемую в составлении таблиц переходов в закодированном графе.

21. Какие три функциональные подсистемы содержатся в структуре программируемых логических матриц (ПЛМ)?

22. Какая наиболее распространённая технология программирования ПЛМ?

23. Какие булевы функции позволяют воспроизвести ПЛМ: СДНФ или СКНФ?

24. Какое назначение блока формирования управляющих сигналов в архитектуре МПУУ.

25. Есть ли различия в размере ячейки памяти в МПУУ при принудительной и естественной адресации?

Учебное издание

Спиридонов Сергей Борисович

ВЫЧИСЛИТЕЛЬНЫЕ СРЕДСТВА АВТОМАТИЗИРОВАННЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

Краткий курс лекций

Учебно-методическое пособие

Часть 1

Издательство «Спутник +»
109052, Москва, Смирновская улица, д. 4, стр. 2.
Тел.: (495) 730-47-74, 778-45-60 (с 9.00 до 18.00)
http://www.sputnikplus.ru E-mail: print@sputnikplus.ru
Подписано в печать 27.04.2024. Формат 60×90/16.
Бумага офсетная. Усл. печ. л. 7,19. Тираж 80 экз. Заказ 86.
Отпечатано в ООО «Издательство «Спутник +»