МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

Учебное пособие для студентов направления

“Информатика и вычислительная техника”

Рецензенты:

Доктор технических наук, профессор

Смирнов Г.В. (ТУСУР)

Доктор технических наук, профессор

Ливенцов С. Н. (НИ ТПУ)

Замятин Н. В.

В пособие включены необходимые материалы для изучения дисциплины «Организация ЭВМ и систем». Рассматриваются основные принципы построения архитектур ЭВМ и систем на их основе. Приводятся основные определения и понятия организации ЭВМ, описываются операционные устройства и методы их синтеза. принципы организации шин, внутренней и внешней памяти, операционных устройств и устройств управления, систем ввода-вывода. Изложены основные тенденции в архитектуре построения и функционирования параллельных и нейрокомпьютерных систем разнообразных классов. Показаны перспективы развития вычислительных систем на новых принципах преобразования информации.

Для студентов факультета дистанционного образования, факультета систем управления и вычислительных систем.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компьютеры, как средство преобразования информации, находят применение практически во всех сферах человеческой деятельности. Эффективная организация работы уже невозможна без применения компьютеров в планировании и управлении производством, проектировании и разработке сложных технических устройств, издательской деятельности, образовании – везде, где возникает необходимость в обработке информации.

Любая форма человеческой деятельности, любой процесс функционирования технического и организационного объекта связаны с передачей и преобразованием информации. Поэтому важнейшее положение кибернетики заключается в том, что без информации и ее переработки невозможны организованные системы, представляющие живые организмы и созданные человеком технические системы.

Информация- это совокупность сведений, снимающих неопределенность. Неопределенность – понятие вероятностное и характеризуется понятием энтропия. Если событие точно произойдет или не произойдет, то неопределенность в обоих случаях равна нулю. Максимальная неопределенность, или максимальное количество полученной информации, снимающей неопределенность, соответствует вероятности событий «пятьдесят на пятьдесят». Процесс получения информации – это процесс снятия неопределенности, в результате чего из некоторой совокупности возможных явлений выделяется то, которое фактически имело место. Информация, воплощенная и зафиксированная в некоторой материальной форме, называется сообщением, которое в свою очередь может быть непрерывным или дискретным. В случае непрерывных сообщений для описания используется система величин, которые являются вещественными числами и могут изменяться непрерывно. Для дискретного же случая характерно то, что величины могут принимать лишь дискретные значения из некоторого ряда. Обработка информации связана с понятием информационная технология. Конкретные факты сообщений представляют данные, а правила перехода между данными - это знания.

В широком смысле понятие технология – это способ освоения человеком материального мира с помощью специально организованной деятельности, включающей три компоненты: информационную (научные принципы и обоснование), материальную (орудие работы) и социальную (специалисты, имеющие профессиональные навыки). Эта триада составляет сущность современного понимания понятия технологии. С одной стороны, технология связана с энергетикой, с другой стороны – с информатикой.

Понятие информационной технологии появилось с возникновением информационного общества, основой социальной динамики в котором яв-

ляются не традиционные материальные, а информационные ресурсы: знания, наука, организационные факторы, интеллектуальные способности.

Информационная система – взаимосвязанная программно-аппаратная совокупность элементов, используемая для хранения, обработки и выдачи информации с целью решения конкретных задач.

Современное понимание информационной системы предусматривает использование компьютера как основного технического средства обработки информации. Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом информационной системы.

В функционировании информационной системы можно выделить следующие этапы:

1. Получение данных и знаний – формирование первичных сообщений, фиксирующих результаты определенных операций, свойства объектов и субъектов управления, параметры процессов, содержание нормативных и юридических актов и т.п.

2. Накопление и систематизация данных и знаний – организация такого их размещения, которое обеспечивало бы быстрый поиск и отбор нужных сведений, методическое обновление данных, защита их от искажений, потери, деформирование целостности и др.

3. Обработка данных и знаний – процессы, вследствии которых на основании прежде накопленных данных формируются новые виды данных: обобщающие, аналитические, рекомендательные, прогнозные. Производные данные тоже можно обрабатывать, получая более обобщенные сведения (знания).

Отображение данных – представление их в форме, пригодной для восприятия человеком. Прежде всего – это вывод на печать, то есть создание документов на твердых (бумажных) носителях. Широко используют построение графических иллюстративных материалов (графиков, диаграмм) и формирование звуковых сигналов.

4. Передача информации – преобразования данных и знаний в пространстве на основе информационных каналов общества, не имеющих пространственных границ.

Передача и преобразования дискретной информации любой формы (обычного текста, содержащего буквы и цифры) могут быть сведены к эквивалентным передаче и преобразованиям сигналов в цифровой форме. Это позволяет с любой необходимой степенью точности непрерывные сообщения заменять цифровыми сигналами путем квантования и дискретизации непрерывного сообщения по уровню и времени.

В силу эффективности цифровой формы представления информации цифровые электронные вычислительные машины представляют собой наиболее универсальный тип устройства обработки информации (УПИ – универсальные преобразователи информации).

Свойства компьютера – автоматизация вычислительного процесса на основе программного управления, огромная скорость выполнения арифметических и логических операций, возможность хранения большого количества различных данных, возможность решения широкого круга задач обработки данных. И поэтому особое значение ЭВМ состоит в том, что впервые с их появлением стала возможной автоматизация процессов обработки информации.

Термин ЭВМ (электронные вычислительные машины) по функциональному назначению не совсем правильный, хотя и популярный, потому что основная задача – это выполнения процедур преобразования информации (данных и знаний). Поэтому лучше использовать термин «компьютер» или «универсальный преобразователь информации».

Компьютеры обеспечивают решение различных задач путем выполнения элементарных математических (арифметических и логических) операций над информацией, представленной исключительно в дискретной форме, – над числами, символами текста, точками графических изображений и т.п.

Технические средства автоматизированных систем обработки информации и управления (АСОИУ) реализуются на базе ЭВМ. Вместе с тем, в настоящее время термин ЭВМ (компьютер) трактуется широко – под ним понимается не только аппаратура, но и программное обеспечение (ПО, т.е. система в целом. Поэтому в АСОИУ совокупность ЭВМ и ПО обычно называют вычислительной системой (ВС).

Понятие организация вычислительных систем используется в отечественной литературе, тогда как за рубежом используется понятие архитектура. Следовательно, организация и архитектура применительно к вычислительным систем это синомимы.

Таким образом, предметом дисциплины «Организация ЭВМ и систем» являются различного рода ВС и принципы их организации (архитектуры).

Глава 1. ОСНОВЫ КОНЦЕПЦИИ КОМПЬЮТЕРОВ

1.1 Основные понятия и определения

Любой современный универсальный преобразователь информации – это сложная система, и поэтому целесообразно рассмотреть эти понятия.

Система – это совокупность элементов, объединенных для достижения какой-либо цели. Совокупность элементов при объединении (интегративности) приобретает новое свойство (обработка информации), что и является целью создания системы. Для системы характерны следующие основные понятия.

Структура системы – это совокупность элементов и связей между ними.

Функция системы – это представление системы, описывающее, как достигается поставленная перед системой цель, или функция системы – это правила получения результатов, вытекающих из назначения системы.

Организация – это способ соединения элементов, целью которого является получение требуемых функций в системах, состоящих из большого числа элементов.

Компьютер (электронно-вычислительная машина) представляется как сложная система. Поэтому существует иерархия понятий. Любая система состоит из элементов.

Элемент – это неделимая частица на данном уровне иерархии. На низких уровнях элемент рассматривается как система, структура которой, в свою очередь, строится на основе более простых элементов и связей между ними. Применительно к ЭВМ иерархия понятий представляется следующим образом: сначала простейшие кирпичики – логические элементы, затем логические узлы, потом операционные устройства, и наконец, микропроцессоры, далее сложные системы – преобразователи информации (компьютеры).

Электронная вычислительная машина (ЭВМ) обеспечивает преобразование информации. Термин **вычислительная** означает, что обработка информации осуществляется путем выполнения сравнительно простых математических (арифметических, логических и т.п.) операций, т.е. путем вычислений. Термин **электронная** означает, что машина построена на основе электронных элементов, электронной элементной базы.

Под вычислительной системой понимается система, состоящая из двух частей (элементов) – АО и ПО, находящихся во взаимодействии. Здесь: АО – аппаратное обеспечение. АО ВС – это технические средства ВС.

ПО ВС – это системное ПО (СПО) и прикладное ПО (ППО), Архитектура ВС включает общую логическую организацию ВС, режимы работы (т.е. взаимодействие АО и ПО), способы представления данных, способы адресации и т.д.

ЭВМ – это сложная система, предназначенная для автоматизации обработки информации на основе алгоритмов, состоящая из устройств: процессоров, запоминающих устройств (ЗУ), устройств ввода-вывода (УВВ).

Если основным признаком распределенной системы является наличие нескольких центров обработки данных, то к распределенным системам также можно отнести как многопроцессорные и многоядерные системы, а также многопроцессорные комплексы (кластеры). На рис. 1.1.1 приведена иерархическая схема структуры преобразователей информации.

В многопроцессорных системах имеется много процессоров, каждый из которых может выполнять свою работу, независимо от остальных, имея общую операционную систему и общую память.

В многопроцессорных комплексах (кластерах) имеется совокупность

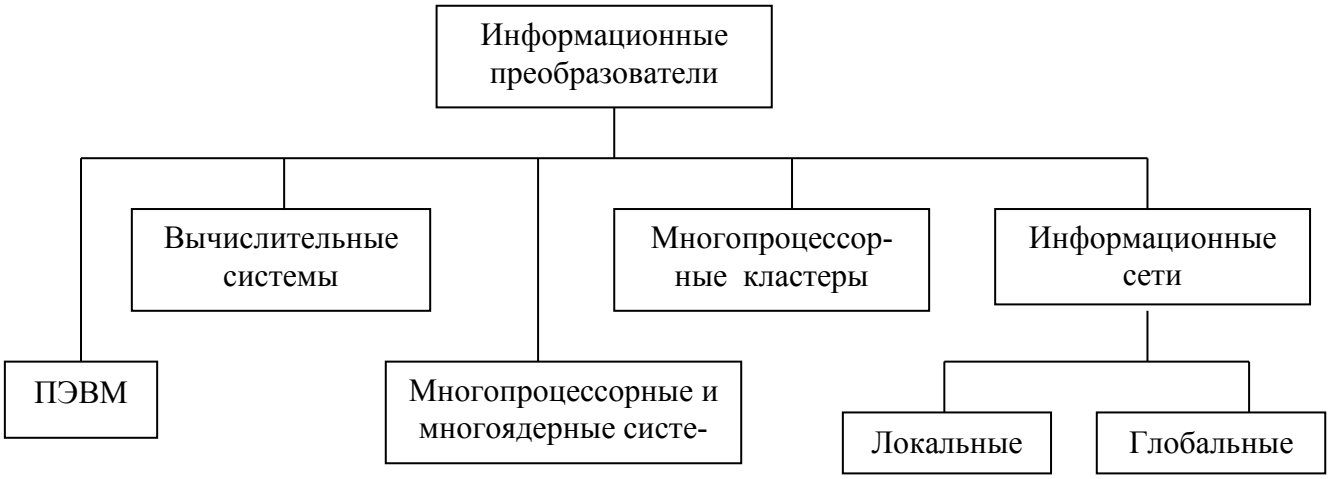


Рис. 1.1.1 – Структура преобразователей информации

процессоров, объединенных в кластеры, с общей памятью, но могут иметь и раздельные операционные системы. Связь между процессорами и кластерами осуществляется через программные и аппаратные средства связи.

1.2 Обобщенная структура компьютера

В цифровых компьютерах информация представляется и обрабатывается с помощью электронных логических схем. Логические схемы оперируют двоичными переменными, принимающими одно из двух значений (нуля и единицы).

Определим компьютер или вычислительную систему как комплекс устройств, выполняющий представление программы (алгоритма) в виде физических процессов, назначением которых является реализация арифметических и логических операций над информацией, представляемой в цифровой форме.

ЭВМ содержит следующие основные устройства: арифметическо-логическое устройство, внешняя и внутренняя память, устройство ввода и вывода данных и устройства управления. Обобщенная структурная схема ЭВМ приведена на рис. 1.2.1.

Арифметико-логическое устройство (АЛУ) производит арифметические и логические преобразования над поступающими в него машинными словами, т.е. кодами определенной длины.

Память хранит информацию, передаваемую из других устройств, в том числе поступающую в машину извне через устройство ввода, и выдает

во все другие устройства информацию, необходимую для выполнения вычислительного процесса. Память машины состоит из двух, отличающихся по своим характеристикам частей: быстродействующей оперативной (внутренней) памяти (на рис. блок ОЗУ – оперативное запоминающее устройство) и медленно действующую, но хранящую больший объем информации внешнюю память (блок ВЗУ – внешнее запоминающее устройство).

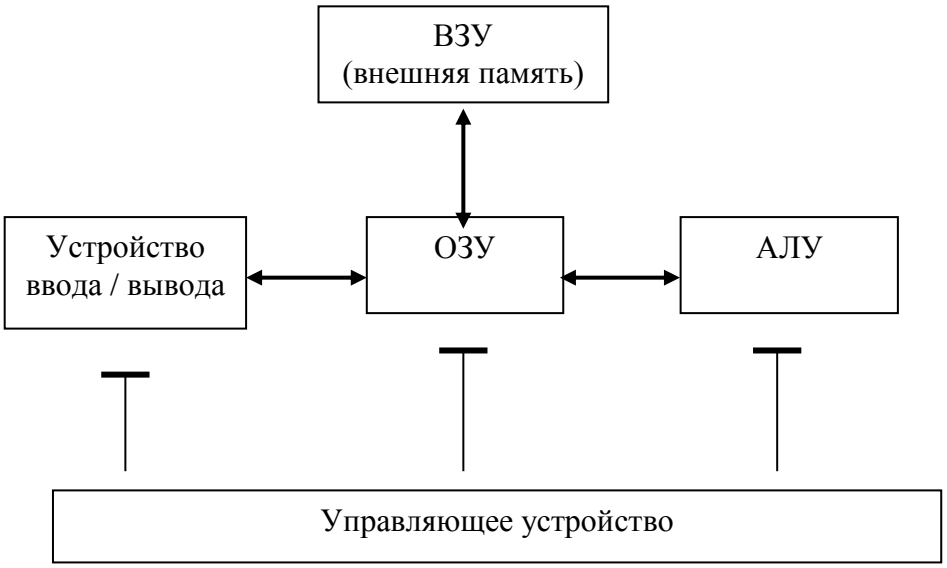


Рис.1.2.1 – Обобщенная структурная схема компьютера

Управляющее устройство (УУ) автоматически управляет вычислительным процессом, посылая всем другим устройствам управляющие сигналы, инициирующие выполнение соответствующей последовательности микроопераций, обеспечивающих реализацию текущей команды. Наиболее важными частями являются АЛУ и УУ. В них выполняются все основные процессы, связанные с переработкой информации, и они выделяются в группу оборудования, называемую процессором. В частности, УУ указывает ОЗУ, какие слова должны быть переданы в АЛУ и в другие устройства, включает АЛУ на выполнение нужной операции и помещает полученный результат в ОЗУ.

Устройство ввода используется для считывания программы и исходных данных из внешних устройств и переноса их в ОЗУ. Необходимая для решения задачи информация может также вводиться непосредственно с клавиатуры.

Устройство вывода служит для выдачи из машины результатов преобразования информации, например, путем вывода их на печатные устройства или отображения на экране монитора.

Принципы функционирования УПИ, во многом определяющие процессы обработки информации в компьютерах:

- принцип Фон Неймана, основанный на последовательной обработке информации;
- принцип Оккама (гарвардская модель), основанный на параллельном функционировании;
- нейросетевой подход.

Принцип Фон Неймана основан на следующих положениях:

- принцип условного перехода, согласно которому выполнение команд осуществляется при условии выполнения некоторого условия. Благодаря условному переходу компьютер автоматически изменяет ход вычислительного процесса, что позволяет решать сложные логические задачи;
- принцип программного управления, являющийся основной особенностью компьютера, посредством которого достигается автоматическое управление процессом решения задачи;
- принцип хранимой в памяти программы, согласно которому команды программы, закодированные в цифровом виде, хранятся в памяти наравне с числами. В команде указываются участвующие в операциях адреса ячеек ОП, в которых они находятся, и адрес ячейки, куда помещается результат операции;
- принцип двоичной арифметики (булевой алгебры), смысл которого состоит в использования двоичных слов в физическом смысле;
- принцип иерархической памяти, включающей пять «ступеней»: регистры, ОЗУ, кэш-память, ВЗУ, архивные запоминающие устройства (АрЗУ).

Фон-неймановская модель вычислительной машины проста и гибка при управлении вычислительным процессом. Недостаток ее заключается в общей памяти для данных и команд и общей шине (магистрали) для передачи команд и данных из памяти в другие устройства, ограничивающие скорость преобразования информации. Проблема решается использованием кэш-памяти и расслоения памяти, введением конвейера и очереди команд, пакетной передаче данных, переходом к параллельной обработке информации, многоядерным процессорам.

Принцип параллельного вычисления заключается в одновременном выполнении нескольких процессов обработки информации. Это реализуется построением компьютера с отдельными памятью и шинами для хранения и передачи команд и данных, допускающими параллельное во времени извлечение их из памяти и передачу по шинам. Такая модель получила название «гарвардской» и была реализована в 1944 г. в США.

Нейросетевой подход предусматривает параллельную обработку информации подобно человеческому мозгу. В его основе лежит использование простейших элементов – нейронов, объединенных сложными связями. Основной задачей является «обучение» нейронов. Применение технологии нейросетевой обработки позволяет проводить прогноз, зрительное распо-

знавание, управлять учреждениями и решать другие сложные трудноформализуемые задачи.

1.3 Организация вычислительных процессов

Организацию вычислительных процессов в компьютерах выполняют управляющие программы операционных систем (ОС), обеспечивающие взаимодействие аппаратуры ЭВМ и ПО и задающие соответствующие режимы работы.

Алгоритмические языки и трансляторы предназначены для снижения трудоемкости программирования в машинных кодах путем передачи этой рутинной работы трансляторам. Задача операционной системы заключается в автоматизации взаимодействия программ и аппаратуры ЭВМ для повышения эффективности их использования.

Функционирование ЭВМ заключается в организации процессов выполнения задач (заданий) на аппаратной части компьютера. Задача – это программа и данные, загруженные в ОЗУ, т.е. программа вместе с выделенными ей ресурсами. Мультизадачный (мультипрограммный) это основной режим работы современных вычислительных систем.

Недостаток однопрограммного режима – неэффективное использование оборудования, в частности процессора (временная диаграмма рис.1.3.2,), простаивающего большую часть времени. Быстродействие периферийных устройств (ПУ), посредством которых осуществляется ввод/вывод информации, ниже быстродействия процессора и памяти. В результате простоев процессора производительность ЭВМ минимальна. В силу этого недостатка однопрограммный режим практически не применяется.

В мультипрограммном режиме, кроме программ ОС, в ОЗУ компьютера располагаются и (по очереди) выполняются множество программ пользователей с целью увеличения загрузки ЦП и, как следствие, производительности ЭВМ. Работу ЭВМ в мультипрограммном режиме можно проиллюстрировать временной диаграммой (рисунок 1.3.2).

Мультипрограммный режим минимизирует соотношение цена/производительностm I: I=S/Λ, где S – стоимость ЭВМ, Λ – производительность ВК (количество задач в единицу времени). Впервые это понятие

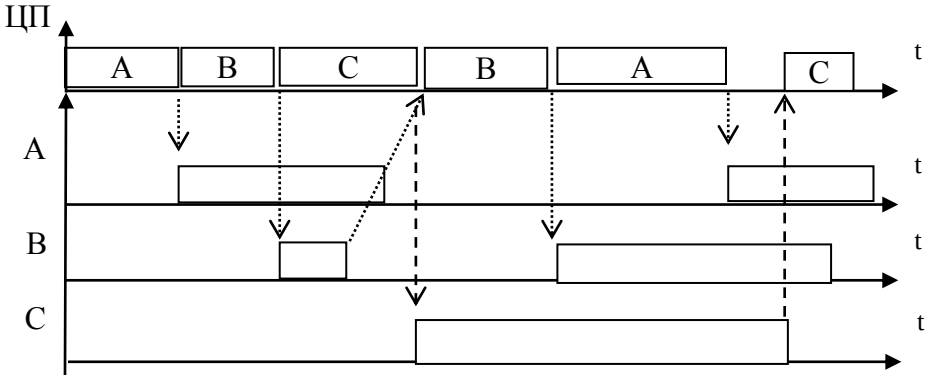


Рис. 1.3.2 – Мультипрограммный режим

в вычислительной технике использовал В.М. Глушков [ ].

Зависимости I, S, Λ от уровня мультипрограммирования M представлены на рисунке 1.3.3., здесь $M_{опт}$ – оптимальный уровень мультипрограммирования зависит от многих факторов: класса решаемых задач, быстродействия устройств вычислительной системы (ВС), емкости памяти, структуры ВС и др.

Определение значения $M_{опт,}$ при котором цена производительности

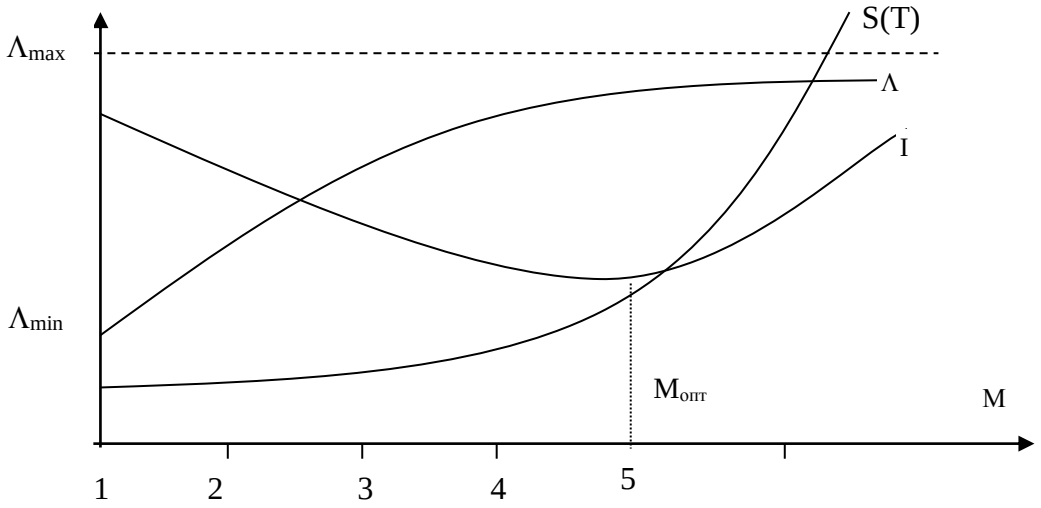


Рис. 1.3.3 – Зависимости I, S, Λ от уровня мультипрограммирования

принимает минимальное значение, выполняется на моделях. Экспериментально установлено, что для средних по вычислительной мощности компьютерах оптимальный уровень мультипрограммирования $M_{опт}$ лежит в пределах от 4 до 8, для больших – в пределах от 8 до 16.

Основное достоинство мультизадачного режима – минимизация соотношения цена/производительность. Основной недостаток – увеличение времени решения (ответа) из-за ожидания в очередях к ресурсам. С увеличением быстродействия процессоров этот недостаток практически не играет роли.

Мультипрограммный режим значительно сложнее, поэтому к ВС предъявляют специальные требования, для реализации которых в состав компьютеров вводятся специальные средства мультипрограммирования.

1. Средства, обеспечивающие заданный режим мультипрограммирования (управляющие программы ОС), обеспечивающие определенный порядок прохождения задач через компьютер. Именно они образуют ядро ОС и обеспечивают выделение требуемые программам ресурсы ВС: объемы памяти, различных устройств и т.п., а также слежение за границами областей памяти, выделенных задаче. В общем случае управляющие программы ОС (супервизор, монитор, диспетчер, планировщик и др) обеспечивают взаимодействие между аппаратурой ВС и многочисленными программами пользователей.

2. ОЗУ увеличенной емкости, т.к. для размещения М задач требуется больше места в ОП (в М раз больше).

3. Средства, обеспечивающие одновременную (параллельную) работу ЦП и средств ввода/вывода информации в ОП. Система прерываний ЭВМ предназначена для быстрой реакции на различные ситуации (события), возникающие внутри и вне компьютера. Быстрая реакция на события обеспечивается путем прерывания процесса выполнения текущей программы и перехода к выполнению программы, обслуживающей данную причину прерывания. По завершению обслуживания, естественно, осуществляется возврат к прерванной программе и ее продолжению с того места, в котором она была прервана.

Система программного (математического) обеспечения – это комплекс программных средств, в котором можно выделить операционную систему, комплект программ технического обслуживания и пакеты прикладных программ. На рис. 1.3.4 изображена структура вычислительной системы как совокупности аппаратных и программных средств.

Операционная система (ОС) – это центральная и важнейшая часть программного обеспечения ЭВМ, предназначенная для эффективного управления вычислительным процессами облегчения общения оператора с машиной, планирующая работу и распределение ресурсов ЭВМ, автоматизации процесса подготовки программ и организации их выполнения при различных режимах работы машины.

1. Управляющие программы – осуществляют управление работой устройств ЭВМ, т.е. координируют работу устройств в процессе ввода, подготовку и выполнение других программ.

2. Обрабатывающие программы – осуществляют работу по подготовке новых программ для ЭВМ и исходных данных для них, например сборку отдельно транслируемых модулей в одну или несколько исполняемых программ, работу с библиотеками программ, перезаписи массивов информации между ВП и т.д. ОС в большинстве случаев являются универсальными и не учитывают особенности конкретных аппаратных средств. В современных компьютерах для адаптации ОС к конкретным аппаратным средствам используют аппаратно-ориентированную часть операционной системы, которая называется BIOS (Basic Input / Output System – базовая система ввода/вывода).

Оператор и пользователь не имеют прямого доступа к аппаратным средствам ЭВМ. Все связи осуществляются только через ОС, обеспечивающую определенный уровень общения человека и машины, который определяется в первую очередь уровнем языка общения.

Проблемно-ориентированный язык – это язык, ориентированный на какую-либо проблему (задачу моделирования сложной системы, задачу САПР и т.д.).

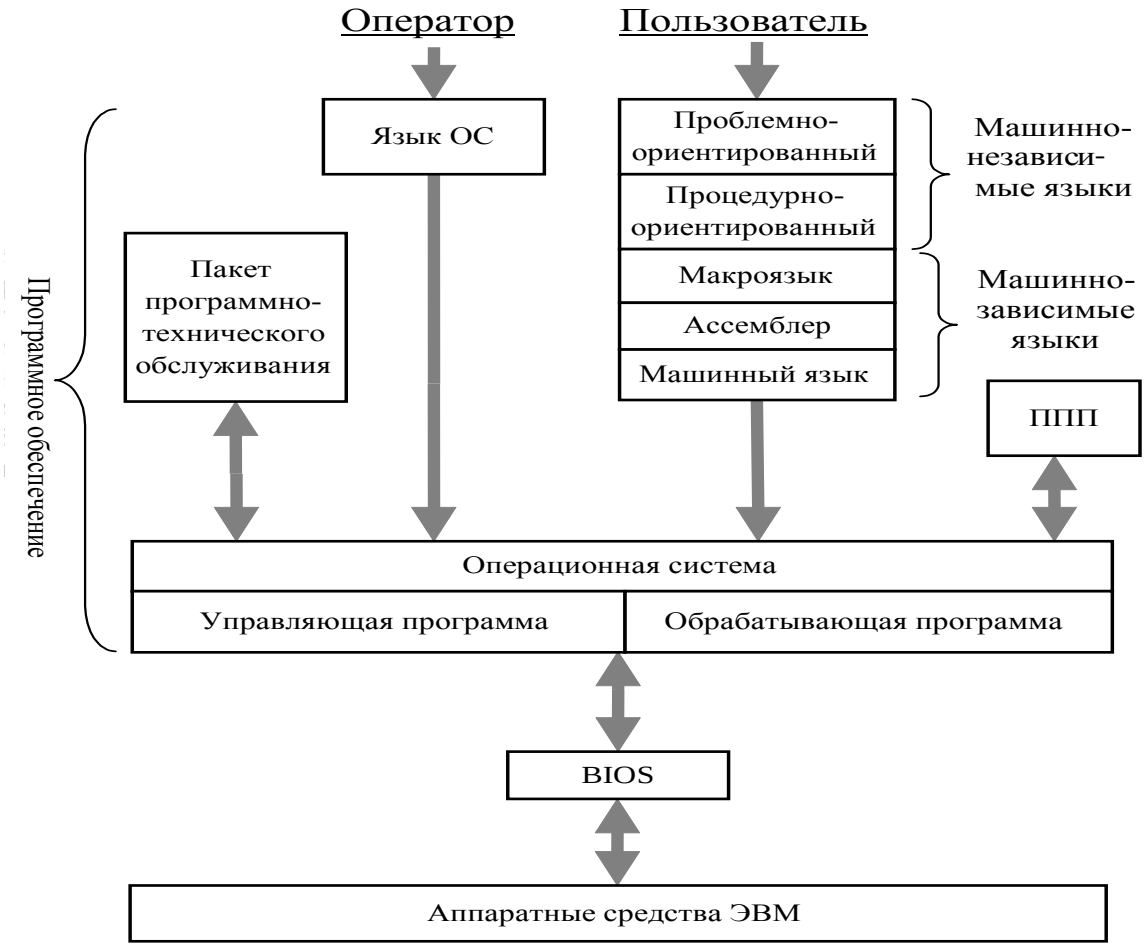


Рис. 1.3.4 – Вычислительная система, как совокупность программных и аппаратных средств копьютера

Процедурно-ориентированный язык – это язык, ориентированный на выполнение общих процедур обработки данных (Фортран, Си, Руби, Питон и т.д.).

Машинный язык – это самый нижний уровень языка. Команды записываются в виде двоичных кодов. Адреса ячеек памяти – абсолютны. Программирование очень трудоемко.

Ассемблер – это язык более высокого уровня, использует мнемокоды (т.е. команды обозначаются буквенными сочетаниями). Запись программы ведется с использованием символических адресов, т.е. вместо численных значений адреса используются имена. За исключением первого оператора программы, который должен быть жестко привязан к физическому адресу. Макроязык – в первом приближении можно определить как язык процедур, написанных на языке ассемблера, т.е. когда вместо целого комплекса команд (которые часто встречаются) используется только имя (название) этого комплекса.

Язык ОС – это язык, на котором оператор может выдавать директивы ОС, вмешиваться в ход вычислительного процесса.

Пакет программно-технического обслуживания предназначен для уменьшения трудоемкости эксплуатации компьютера. Эти программы позволяют провести тестирование работоспособности ЭВМ и ее отдельных устройств, определять места неисправностей.

Пакеты прикладных программ представляют комплексы программ для решения достаточно широких классов задач (научно-технических, планово-экономических), а также для расширения функций ОС (управление базами данных, реализация режимов телеобработки данных, реального времени и др.).

Сложность современной вычислительной системы привела к возникновению понятия архитектуры вычислительных машин. Это понятие охватывает комплекс общих вопросов построения ВС, существенных в первую очередь для пользователя, интересующегося главным образом возможностями компьютера, а не деталями его технического исполнения. К числу таких вопросов относятся вопросы общей структуры, организации вычислительного процесса и общения пользователя ссистемой, вопросы логической организации представления, хранения и преобразования информации и вопросы логической организации совместной работы различных устройств, а также аппаратных и программных средств машины.

1.4 История развития и поколения ЭВМ

Этапы развития ЭВМ составляют основу классификации ЭВМ по поколениям.

Первые вычислительные машины в современном смысле появились в конце 30-х – начале 40-х годов. В 1941 году была создана машина на электромагнитных реле и с программным управлением на перфоленте. Перфолента двигалась в одну сторону, и все циклы записывались в развернутом виде, т.е. в виде последовательности групп команд.

В 1944 году появились машины, построенные на электронных лампах (Марк). В Англии была введена в эксплуатацию первая в мире ЭВМ с хранимой в памяти программой – «ЭДСАК».

Первое поколение (1955–1960 гг.)

ЭВМ этого поколения строились на дискретных элементах и вакуумных лампах, имели большие габариты, вес, мощность, обладая при этом малой надежностью. Они использовались в основном для решения научно-технических задач атомной промышленности, реактивной авиации и ракетостроения.

Машины этого поколения имели быстродействие порядка 10–20 тысяч операций в секунду и ОЗУ порядка 1К (1024 слова). В этот же период

появились первые простые языки для автоматизированного программирования.

Второе поколение (1960–1965 гг.)

В качестве элементной базы использовались дискретные полупроводниковые приборы и миниатюрные дискретные детали, а в качестве технологической – печатный монтаж. Уменьшились габариты и энергозатраты, возросла надежность. Возросли также быстродействие (приблизительно 500 тысяч оп/сек) и объем оперативной памяти (16–32К слов). Появились языки высокого уровня (Фортран, Алгол, Кобол) и соответствующие им трансляторы. Служебные программы оформились в ОС, которая автоматизировала работу оператора: ввод текста программы, вызов нужного транслятора, вызов необходимых библиотечных программ, размещение программ в основной памяти. Совершенствование аппаратного обеспечения привело к тому, что появилась возможность строить в ЭВМ помимо центрального (основного) процессора еще ряд вспомогательных процессоров.

Третье поколение (1965–1970 гг.)

В качестве элементной базы использовались интегральные схемы малой интеграции с десятками активных элементов на кристалл, что позволило сократить габариты и мощность, повысить быстродействие. Появилась возможность создания малогабаритных, надежных, дешевых машин – мини-ЭВМ,

Четвертое поколение (с 1970 г.)

Успехи микроэлектроники позволили создать БИС и СБИС, содержащие десятки тысяч активных элементов. Это позволило разработать более дешевые ЭВМ с большой ОП. Стоимость одного байта памяти и одной машинной операции резко снизилась. Но затраты на программирование почти не сократились. Поэтому на первый план вышла задача экономии человеческих, а не машинных ресурсов.

Совершенствование БИС и СБИС привело в начале 70-х гг. к появлению новых типов микросхем – микропроцессоров (в 1968 г. фирма Intel разработала и изготовила первые БИС микропроцессоров, которые первоначально предполагалось использовать как составные части больших процессоров).

Первоначально под микропроцессором понималась БИС, в которой полностью размещен процессор простой архитектуры, т.е. АЛУ и УУ. В результате были созданы дешевые микрокалькуляторы и микроконтроллеры – управляющие устройства, построенные на одной или нескольких БИС, содержащие процессор, память и устройства сопряжения с датчиками и исполнительными механизмами. В 70-е же годы появились первые микро-ЭВМ – универсальные вычислительные системы, состоящие из процессора, памяти, схем сопряжения с устройствами ввода/вывода и тактового генератора, размещенные в одной БИС (однокристальная ЭВМ) или

в нескольких БИС, установленных на одной печатной плате (одноплатные ЭВМ).

Использование в больших ЭВМ микропроцессоров и СБИС позволило резко увеличить объем памяти и реализовать некоторые функции программ ОС аппаратными методами.

Характерным для крупных ЭВМ 4-го поколения является наличие нескольких процессоров, ориентированных на выполнение определенных операций, процедур или решение определенных классов задач.

Пятое поколение

Характерной особенностью пятого поколения ЭВМ является то, что основные концепции этого поколения были заранее формулированы в явном виде. Задача разработки принципиально новых компьютеров впервые поставлена в 1979 году японскими специалистами, объединившими свои усилия под эгидой научно-исследовательского центра по обработке информации – JIPDEC. В 1981 г. JIPDEC опубликовал предварительный отчет, содержащий детальный многостадийный план развертывания научно-исследовательских и опытно-конструкторских работ с целью создания к 1991 г. прототипа ЭВМ нового поколения.

Разработаны концепции создания не только поколения ЭВМ в целом, но и вопросы архитектуры основных типов ЭВМ этого поколения, структуры программных средств и языков программирования, разработки наиболее перспективной элементной базы и способов хранения информации.

Прогнозы японских специалистов не сбылись. До сих пор не создан компьютер в полной мере их удовлетворяющий. Основное внимание уделяется компьютерам, использующим СБИС и многопроцессорные схемы. Происходит слияние микроЭВМ и суперкомпьютеров. Переход на высокие степени интеграции и, соответственно, к радикальным изменениям в организации вычислительных процессов. В первую очередь это оптические, квантовые и молекулярные компьютеры.

Контрольные вопросы

1. Почему не сбылись планы по построению компьютеров пятого поколения?
2. Почему память на обобщенной структурной схеме компьютера находится между устройствами ввода и вывода и процессором?
3. В чем принципиальное отличие принципов фон Неймана и Оккама?
4. В чем заключается закон Мура?
5. Что такое данные и знания?
6. Какая связь между информацией и энтропией?

Глава 2. ЛОГИЧЕСКИЕ ОСНОВЫ ПРЕОБРАЗОВАТЕЛЕЙ ИНФОРАЦИИ

2.1 Булевы (переключательные) функции

Функционирование логических схем основано на математическом аппарате переключательных (булевых) функций. Переменные $x_1, x_2, \ldots, x_n$ называются двоичными, если они могут принимать только два значения: 0 и 1. Функция от двоичных переменных $f(x_1, x_2, \ldots, x_n)$ называется булевой, если она, как и ее аргументы, принимает только два значения: 0 и 1.

Входной набор обычно нумеруют десятичным числом, поэтому количество k входных наборов для булевой функции n переменных равно $k = 2^n$. Всего различных функций n переменных будет равно 2 в степени 2^n.

Булева функция может быть задана таблицей ее значений, называемой *таблицей истинности*, в зависимости от значений аргументов. Виды булевых функций одной и двух переменных, их условные обозначения и наименования приведены в табл. 2.1 и 2.2 соответственно.

Таблица 2.1 – Булевы функции одной переменной

Булева функция	Имя булевой функции	x	
		0	1
$f_1=0$	Константа 0	0	0
$f_2= x$	Повторение	0	1
$f_3= \neg x$	Инверсия (функция НЕ)	1	0
$f_4=1$	Константа 1	1	1

Таблица 2.2 – Булевы функции двух переменных

Булева функция	Имя булевой функции	x_1			
		0	0	1	1
		x_2			
		0	1	0	1
$f_1= 0$	Константа 0	0	0	0	0
$f_2= x_1 x_2$	Конъюнкция (логическое И)	0	0	0	1
$f_3= \neg(x_1 \rightarrow x_2)$	Запрет x_1	0	0	1	0
$f_4= x_1$	Повторение x_1	0	0	1	1
$f_5= \neg(x_2 \rightarrow x_1)$	Запрет x_2	0	1	0	0
$f_6= x_2$	Повторение x_2	0	1	0	1

Булева функция	Имя булевой функции	x_1			
		0	0	1	1
		x_2			
		0	1	0	1
$f_7 = x_1 \oplus x_2$	Сложение по модулю 2	0	1	1	0
$f_8 = x_1 \vee x_2$	Дизъюнкция (логическое ИЛИ)	0	1	1	1
$f_9 = x_1 \downarrow x_2$	Стрелка Пирса	1	0	0	0
$f_{10} = x_1 \sim x_2$	Эквивалентность	1	0	0	0
$f_{11} = \neg x_2$	Инверсия x_2	1	0	1	0
$f_{12} = x_2 \rightarrow x_1$	Ипликация x_2 в x_1	1	0	1	1
$f_{13} = \neg x_1$	Инверсия x_1	1	1	0	0
$f_{14} = x_1 \rightarrow x_2$	Ипликация x_1 в x_2	1	1	0	1
$f_{15} = x_1 \mid x_2$	Штрих Шеффера	1	1	1	0
$f_{16} = 1$	Константа 1	1	1	1	1

Используя принцип суперпозиции из нескольких простых булевых функций формируется сложная функция, в частности булева функциа от большего числа переменных. При этом запись логических формул упрощают, опуская некоторые скобки и считая, что наиболее приоритетна функция отрицания, затем идет конъюнкция, после нее – дизъюнкция. Все остальные функции имеют равный приоритет, меньший, чем у дизъюнкции.

2.2 Понятие о функциональной полноте

При синтезе ЭВМ существует важное понятие как функциональная полнота булевой функции или системы булевых функций. Это набор функций, из которого методом суперпозиции можно получить булеву функцию, посредством которой можно преобразовать любой сколь угодно сложный алгоритм. В математической логике доказывается, что множество булевых функций {И, ИЛИ, НЕ} является базисом, называемым базисом Буля. Следующие утверждения доказывают возможность такого представления в виде формулы, содержащей только функции дизъюнкции, конъюнкции, отрицания.

Представление булевой функции в форме дизъюнкции:

$$f(x_1, x_2, \ldots, x_n) = K_1 \vee K_2 \vee \ldots \vee K_m),\ m \geq 1,$$

где каждый терм K_i (или конъюнкт) представляет собой конъюнкцию взятых с отрицаниями или без них двоичных переменных функции, называется дизъюнктивной нормальной формой этой функции (ДНФ). Если каждый

конъюнкт содержит в точности по одной все (взятые с отрицаниями или без них) двоичные переменные функции, ДНФ называется совершенной дизъюнктивной нормальной формой этой функции (СДНФ).

Представление булевой функции в форме конъюнкции:

$$f(x_1, x_2, \ldots, x_n) = D_1 \wedge D_2 \wedge \ldots \wedge D_m),\ m \geq 1,$$

где каждый терм D_i (или дизъюнкт) представляет собой дизъюнкцию взятых с отрицаниями или без них двоичных переменных функции, называется конъюнктивной нормальной формой этой функции (КНФ). Если каждый конъюнкт содержит в точности по одной все (взятые с отрицаниями или без них) двоичные переменные функции, КНФ называется совершенной конъюнктивной нормальной формой этой функции (СКНФ).

Любая булева функция может быть представлена в СДНФ (кроме тождественного ноля) или в СКНФ (кроме тождественной единицы), причем представление функции в СДНФ или СКНФ единственно.

Функции базиса – это полный набор строительных блоков, из которых можно строить все другие двоичные функции от любого числа переменных, а следовательно, реализовать любые конечные функциональные преобразователи.

Определение функциональной полноты или неполноты булевых функций выполняется на основе пяти замечательных свойств булевых функций:

1. Функция $f(x_1, x_2, \ldots, x_n)$ называется сохраняющей ноль, если на наборе из нулей принимает значение ноль, т.е. $f(0, 0, \ldots, 0) = 0$.

2. Функция $f(x_1, x_2, \ldots, x_n)$ называется сохраняющей единицу, если на наборе из единиц принимает значение единица, т.е. $f(1, 1, \ldots, 1) = 1$.

3. Функция $f(x_1, x_2, \ldots, x_n)$ называется самодвойственной, если для любого набора $<x_1, x_2, \ldots, x_n>$ $f(x_1, x_2, \ldots, x_n) = \neg f(\neg x_1, \neg x_2, \ldots, \neg x_n)$. Набор $x = <x_1, x_2, \ldots, x_n>$ предшествует набору $y = <y_1, y_2, \ldots, y_n>$, если $x_n \leq y_n$ $(i = 1, 2, \ldots, n)$. Этот факт обозначается $x << y$.

4. Функция $f(x_1, x_2, \ldots, x_n)$ называется монотонной, если для любой пары наборов $x = <x_1, x_2, \ldots, x_n>$ и $y = <y_1, y_2, \ldots, y_n>$ таких, что $x << y$, $f(x) \leq f(y)$.

5. Функция $f(x_1, x_2, \ldots, x_n)$ называется линейной, если ее полином имеет вид $f(x_1, x_2, \ldots, x_n) = a_0 \oplus a_1 \cdot x_1 \oplus \ldots \oplus a_n \cdot x_n$, где $a_i \in \{0, 1\}$ $(i = 0, 1, \ldots, n)$.

Критерий полноты системы булевых функций: необходимо и достаточно, чтобы эта система содержала функцию, не сохраняющую ноль; функцию, не сохраняющую единицу; несамодвойственную функцию, немонотонную функцию, нелинейную функцию. Таблица 2.4

Если проанализировать перечисленные булевы функции, то окажется, что только две функции являются функционально полными. Это – восьмая функция, реализующая логическую операцию И-НЕ, и четырнадцатая функция, реализующая ИЛИ-НЕ.

Таблица 2.4 – Свойства функций двух переменных

x_1				Функция	Свойства функций				
0	0	1	1		Сохраняющая 0	Сохраняющая 1	Самодвойственность	Монотонность	Линейность
x_2									
0	1	0	1						
0	0	0	0	$f_1 = 0$	+	–	–	+	+
0	0	0	1	$f_2 = x_1 x_2$	+	+	–	+	–
0	0	1	0	$f_3 = \neg(x_1 \rightarrow x_2)$	+	–	–	–	–
0	0	1	1	$f_4 = x_1$	+	+	+	+	+
0	1	0	0	$f_5 = \neg(x_2 \rightarrow x_1)$	+	–	–	–	–
0	1	0	1	$f_6 = x_2$	+	+	+	+	+
0	1	1	0	$f_7 = x_1 \oplus x_2$	+	–	–	–	+
0	1	1	1	$f_8 = x_1 \vee x_2$	+	+	–	+	–
1	0	0	0	$f_9 = x_1 \downarrow x_2$	–	–	–	–	–
1	0	0	0	$f_{10} = x_1 \sim x_2$	–	+	–	–	+
1	0	1	0	$f_{11} = \neg x_2$	–	–	+	–	+
1	0	1	1	$f_{12} = x_2 \rightarrow x_1$	–	+	–	–	–
1	1	0	0	$f_{13} = \neg x_1$	–	–	+	–	+
1	1	0	1	$f_{14} = x_1 \rightarrow x_2$	–	+	–	–	–
1	1	1	0	$f_{15} = x_1 \mid x_2$	–	–	–	–	–
1	1	1	1	$f_{16} = 1$	–	+	–	+	+

2.3 Минимизация булевых функций

Сложность схемы, реализующей булеву функцию, определяется сложностью ее аналитической записи. Поэтому булеву функцию упрощают (минимизируют) в целях экономии материалов, объема, веса, и энергопотребления логической схемы.

Для упрощения логических выражений на первом этапе выполняется ряд алгебраических операций, основанных на логических законах: дистрибутивном

$$x(y + z) = xy + xz,$$

законе исключенного третьего:

$$x + \neg x = 1,$$ где знак $\neg$ это отрицание

и дистрибутивном законе:

$$x + yz = (x + y)(x + z).$$

Простые логические схемы, содержащие меньше электронных схем и входных значений, легче и дешевле реализовать на практике. Так что стремление минимизировать логические выражения имеет под собой чисто практическое основание. Законы, которые используются для манипулирования логическими выражениями, объединены в табл. 2.5.

Таблица 2.5 – Законы двоичной логики

Название закона	Алгебраическое тождество	
Коммутативный	$x + y = y + x$	$x\,y = y\,x$
Ассоциативный	$(x + y) + z = x + (y + z)$	$(x\,y)\,z = x\,(y\,z)$
Дистрибутивный	$x + yz = (x + y)(x + z)$	$(x + y)\,z = x\,z + yz$
Идемпотентности	$x + x = x$	$x\,x = x$
Возведение в степень	$\neg\neg x = x$	
Дополнения (закон исключенного третьего)	$\neg x + x = 1$	$\neg x\,x = 0$
Закон де Моргана	$\neg(x + y) = \neg$x $\neg$y	$\neg(x\,y) = \neg$x + $\neg$y
	$1 + x = 1$	$1\,x = x$
	$0 + x = x$	$0\,x = 0$
Закон поглощения	$(x\,y) + x = x$	$(x + y)\,x = x$

Минимизация функций с использованием карты Карно

Для быстрого получения минимального выражения, представляющего логическую функцию нескольких переменных, используют графическое представление таблицы истинности, называемым картой Карно. Для функции трех переменных карта Карно представляет собой прямоугольник, составленный из восьми квадратов, расположенных в два ряда по четыре в каждом. Каждый квадрат соответствует конкретному набору значений входных переменных. Например, третий квадрат в верхнем ряду представляет значения (x_1, x_2, x_3) = 1, 1, 0). Поскольку в таблице истинности функции трех переменных содержится восемь строк, карта должна состоять из восьми квадратов. Значения внутри квадратов – это значения функции при соответствующих значениях переменных.

Главная идея карты Карно заключается в том, что расположенные рядом по горизонтали и по вертикали квадраты отличаются значениями только одной переменной. Если два смежных квадрата содержат единицы, это означает возможность алгебраического упрощения соответствующей пары термов.

Минимизированное произведение, соответствующее группе квадратов, – это произведение входных переменных, значения которых одинаковы для всех квадратов этой группы. Если значение входной переменной x_i равно нулю для всех квадратов группы, тогда переменная x_i входит в результирующее произведение. Квадраты с левого края карты считаются смежными с квадратами с ее правого края.

Общая процедура формирования на карте Карно групп из двух, четырех, восьми и т.д. квадратов определяется просто. Две смежные пары квадратов, содержащих единицы, можно объединить в группу из четырех квадратов. Две смежные группы по четыре квадрата можно объединить в группу из восьми квадратов. В общем случае количество квадратов в группе должно быть равным 2^k, где k – целое число.

На рис 2.3.1 представлена карта Карно для трех аргументов с указанием номеров минтермов. Доказательство положений в карте минтермов на основе теорем склеивания поглощения.

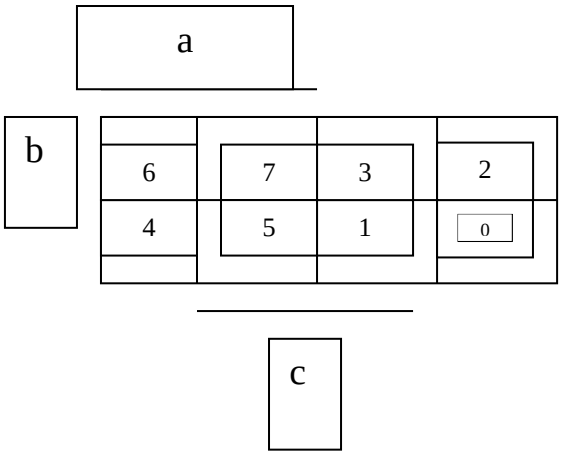


Рис. 2.3.1 – Карта Карно для трех аргументов a, b, c

На рис 2.3.2. представлена карта Карно для четырех аргументов.

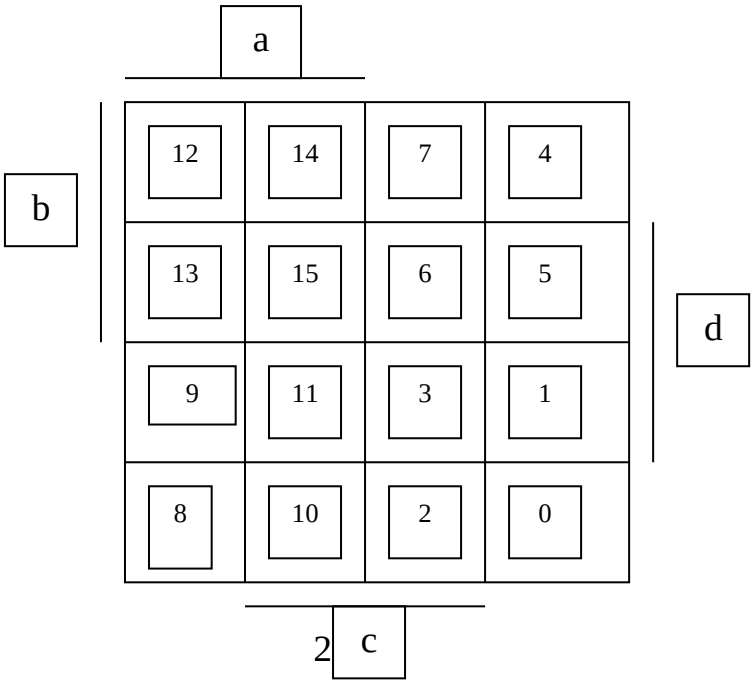


Рис. 2.3.2 – Карта Карно для четырех аргументов a, b, c, d

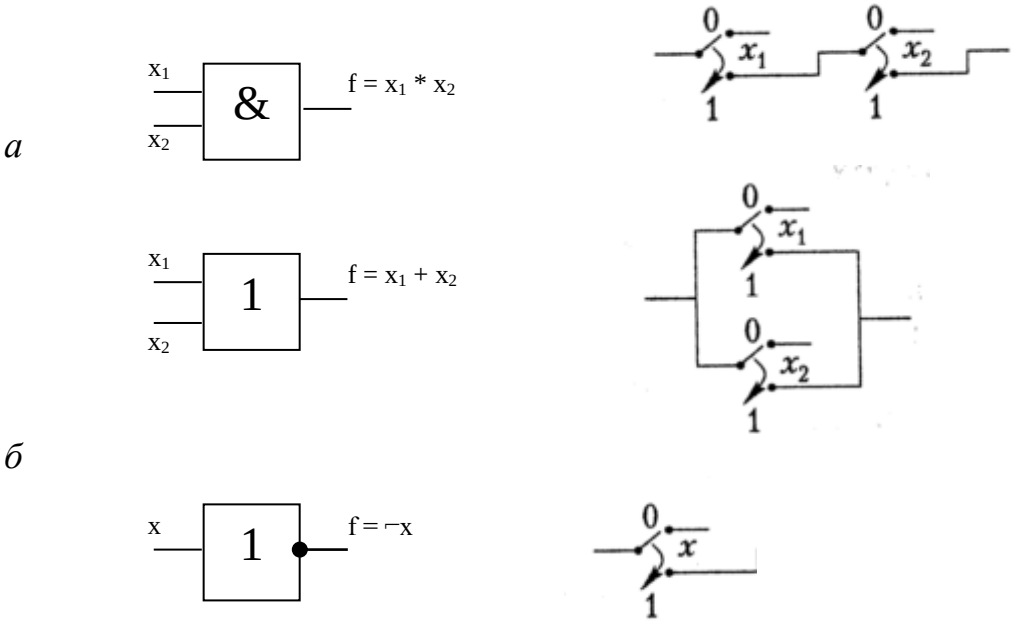
2.4 Логические элементы

2.4.1 Логические элементы без памяти

Техническим аналогом булевой функции является комбинационная схема, выполняющая соответствующее этой функции преобразование информации.

Уровни напряжения шин, соответствующие принятому в схеме представлению сигналов 0 и 1, могут рассматриваться как технические аналоги функции константы 0 и константы 1.

Рассмотрим принципиальные электрические схемы некоторых логических элементов. С применением транзисторов можно сконструировать простые электронные схемы, которые будут выполнять логические операции И, ИЛИ, НЕ. Эти базовые схемы традиционно называют вентилями. Стандартные обозначения вентилей всех трех типов и соединения, реализующие соответствующие типы управлений, приведены на рис. 2.4.1. Если операция НЕ применяется к входному или выходному значению логического вентиля, для нее используется упрощенное обозначение – просто маленький кружок.



а

б

в

Рис. 2.4.1 – Вентиль И и схема параллельного соединения И (*а*); вентиль ИЛИ и схема последовательного соединения ИЛИ (*б*); вентиль НЕ и схема НЕ (*в*)

Существуют еще два базовых логических вентиля, называемых И-НЕ и ИЛИ-НЕ. Эти вентили очень широко применяются в логических схемах,

что объясняется простотой их реализации. Таблицы истинности вентилей И-НЕ и ИЛИ-НЕ приведены на рис. 2.4.2. Они представляют собой штрих Шеффера и стрелку Пирса, т.е. функции И и ИЛИ, к результату которых применена функция НЕ. Используя закон де Моргана (см. табл. 2.5), сможем представить их следующим образом:

$$x_1 \mid x_2 = \neg(x_1 x_2) = \neg x_1 + \neg x_2$$
$$x_1 \downarrow x_2 = \neg(x_1 + x_2) = \neg x_1 \neg x_2$$

x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	0

$x_1 \mid x_2 = \neg(x_1 x_2) = \neg x_1 + \neg x_2$ $x_1 \downarrow x_2 = \neg(x_1 + x_2) = \neg x_1 \neg x_2$

а *б*

Рис. 2.4.2 – Вентили И-НЕ (*а*); ИЛИ-НЕ (*б*)

Для каждой логической функции существует множество вариантов реализации, из которых выбирают вариант с наименьшей стоимостью или с минимальной задержкой распространения. С этой целью можно применить карты Карно, которые подскажут, какие алгебраические операции приведут к оптимальному решению. Оптимизация схем не обязательно выполняется вручную, имеется программное обеспечение для автоматизированного проектирования (Computer-Aided Design, CAD). Пользуясь такой программой, разработчик задает исходную функцию, и программа сама генерирует наиболее эффективную и простую схему для ее реализации.

2.4.2 Логические элементы с памятью

Большинству устройств, в которых задействована цифровая логика, требуются элементы для хранения информации. Логический элемент, используемый для хранения информации, называется триггером.

Триггер является элементом, который может находиться в одном из двух устойчивых состояний. Одному из этих состояний приписывается значение 1, а другому 0. Состояние триггера распознается по его выходному сигналу. Под влиянием входного сигнала триггер может скачкообразно переходить из одного устойчивого состояния в другое. При этом скачкообразно изменяется уровень напряжения его входного сигнала.

Для удобства использования в схемах вычислительных устройств триггеры обычно имеют два выхода – прямой *Q* (называется также «выход

1») и инверсный ⌐*Q* («выход 0»). В единичном состоянии триггера на выходе *Q* высокий уровень сигнала, а в нулевом – низкий. На выходе ⌐*Q* наоборот.

В зависимости от количества входов и особенностей схемы, управляющей ими, меняется логика работы триггера. Схемы триггера можно разделить на несколько типов: с установочными входами – RS-триггер, со счетным входом – Т-триггер, а также D-триггер и специфический для интегральных схем, универсальный JK-триггер.

Если хотя бы с одного входа информация в триггер заносится принудительно под воздействием синхронизирующего сигнала, то триггер называется синхронным. Если занесение информации в триггер с любого входа производится без синхронизирующего сигнала, то триггер называется асинхронным.

Состояние триггера определяется сигналом Q на прямом выходе триггера (или сигналом ⌐Q на его инверсном выходе).

Законы функционирования триггеров задаются таблицей истинности, при которой в столбце состояний может быть указано, что новое состояние совпадает с предыдущим или является его отрицанием. Ниже рассматриваются логические принципы построения наиболее распространенных триггерных схем для потенциальных элементов.

Асинхронный RS-триггер

Асинхронный RS-триггер на интегральных элементах ИЛИ-НЕ показан на рис. 2.4.3,а. Триггер образован из двух комбинационных схем ИЛИ-НЕ, соединенных таким образом, что возникают положительные обратные связи, благодаря которым в устойчивом состоянии входной транзистор одной схемы ИЛИ-НЕ закрыт, а у другой открыт. Таблица на рис. 2.4.3,б, называемая таблицей истинности, определяет закон функционирования этого триггера. В таблице истинности приводятся выходные значения схемы, соответствующие каждой комбинации входных значений. Логические схемы, выходные значения которых однозначно определены для каждого входного значения, называются комбинационными. Схемы, содержащие запоминающие элементы, относятся к другому классу и называются последовательными (рис. 2.4.3). Выходное значение любой такой схемы является функцией значений входных переменных и предшествующего состояния.

В приведенной на рис. 2.4.3,*б* таблице истинности, которая описывает поведение асинхронного RS-триггера, предыдущее состояние схемы обозначено как Q(t). Переход в следующее состояние, обозначаемое как Q(t + 1), происходит после поступления тактового импульса. При R = 1, S = 0 триггер устанавливается в нулевое состояние Q = 0; при R = 0, S = 1 триггер устанавливается в единичное состояние Q = 1; при R = S = 0 триггер сохраняет состояние, в котором он находился до момента поступления на его входы нулевых сигналов. При R = S = 1 на прямом и инверсном входах

устанавливается нулевой сигнал. Триггерное кольцо превращается в два независимых инвертора, и при переходе к *хранению (R = S = 0)* триггер может устанавливаться в любое состояние. Поэтому такая комбинация входных сигналов запрещена.

Условное обозначение RS-триггера показано на рис. 2.4.3,в. Символ Т в основном поле обозначает однотактный триггер. В соответствии с принципом действия данной электронной схемы ее контакты S и R называют входами установки и сброса (set и reset соответственно, откуда и их обозначения). На рис. 2.4.3,г представлены временные диаграммы прохождения сигналов через триггер. Стрелки показывают причинно-следственные отношения между сигналами. Когда значения на входах R и S одновременно изменяются с 1 на 0, результирующее состояние не определено. На практике триггер перейдет в одно из двух своих стабильных состояний, но в какое именно — предсказать невозможно.

Аналитическое функционирование RS-триггера можно получить, используя карты Карно (рис 2.4.4).

Уравнение функционирования имеет вид:

$$Q(t+1) = S(t) \vee Q(t) \neg R(t),$$

причем $S(t)\,R(t) = 0$.

Выберем базис на элементах ИЛИ-НЕ:

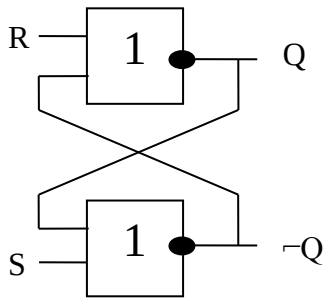
$$Q(t+1) = S(t) + Q(t)\neg R(t) = S(t) + \neg\neg(Q(t)\neg R(t)) = S(t) + \neg(\neg Q(t) + \neg\neg R(t)) = S(t) + \neg(\neg Q(t) + R(t)).$$

Логическая схема на элементах ИЛИ-НЕ уже приводилась на рис. 2.4.3,а.

Если выбрать базис на элементах И-НЕ, то булева функция примет вид:

$$Q(t+1) = S(t) + Q(t)\neg R(t) = \neg\neg(S(t) + Q(t)\neg R(t)) = \neg(\neg S(t)\neg(Q(t)\neg R(t))).$$

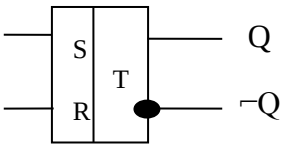
Логическая схема на этих элементах приведена на рис. 2.4.5. Как видно, выгоднее реализовывать схему триггера на элементах ИЛИ-НЕ.



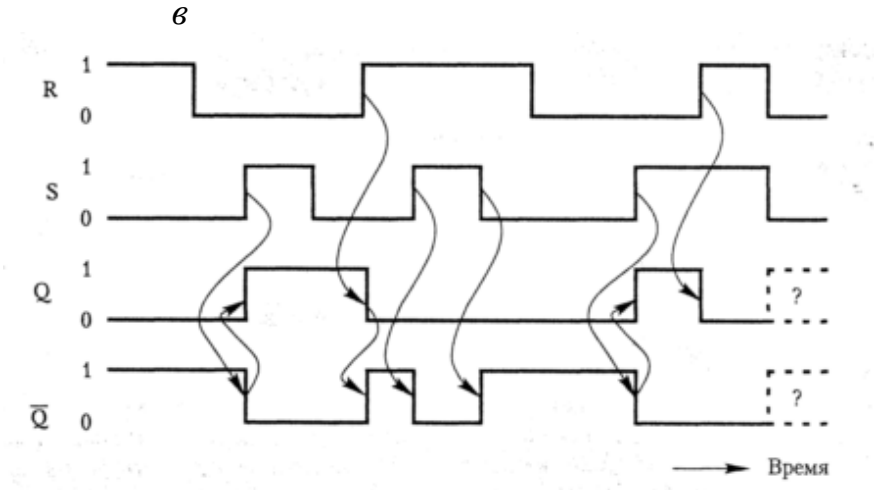
а

t		t+1	Примечание
R	S	Q(t+1)	
0	0	Q(t)	Хранение
0	1	1	Установка 1
1	0	0	Установка 0
1	1	-	Запрещено

б



в



г

Рисунок 2.4.3 – Асинхронный RS-триггер на элементах ИЛИ-НЕ: логическая схема (*а*); таблица истинности (*б*); условное обозначение (*в*); временная диаграмма (*г*)

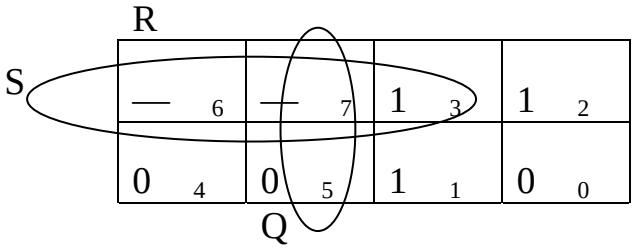


Рисунок 2.4.4 – Карта Карно логической функции RS-триггера

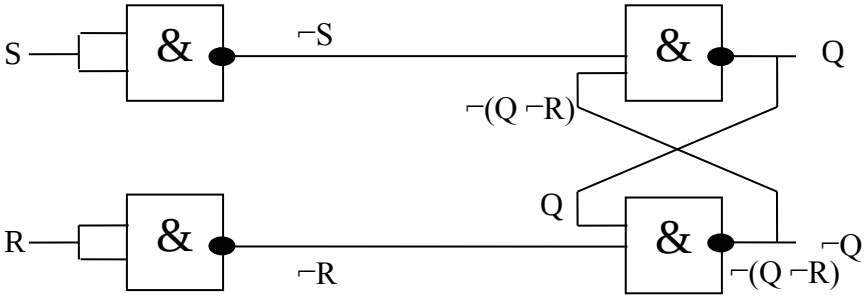


Рисунок 2.4.5 – Логическая схема асинхронного триггера на элементах И-НЕ

Оператор и пользователь не имеют прямого доступа к аппаратным средствам ЭВМ. Все связи осуществляются только через ОС, обеспечивающую определенный уровень общения человека и машины. А уровень общения определяется в первую очередь уровнем языка, на котором оно происходит.

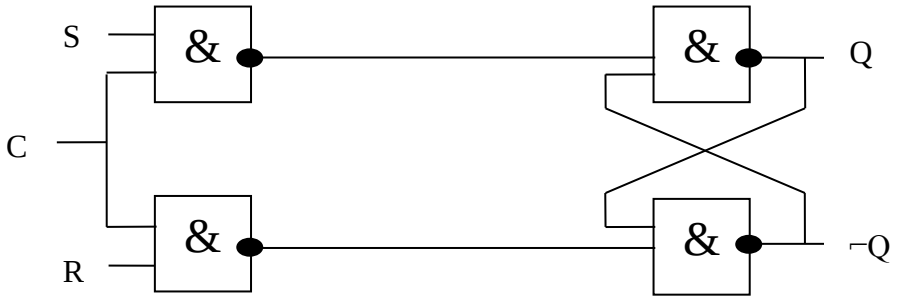
Проблемно-ориентированный язык – это язык, строго ориентированный на какую-либо проблему (задачу моделирования сложной системы, задачу САПР и т.д.).

Синхронный однотактный RS-триггер

Часто необходимо управлять временем сброса и установки триггера посредством входного сигнала, отличного от сигналов R и S, который называется синхронизирующим или тактовым входом. Логический элемент с таким управлением получил название синхронный RS-триггер. Логическая схема, таблица истинности, временная диаграмма и графическое обозначение такого триггера представлены на рис. 2.4.6.6. Здесь элементы 1 и 2 образуют схему входной логики асинхронного RS-триггера, построенного на элементах 3 и 4.

Когда сигнал на синхронизирующем входе C равен 1, сигналы в точках S' и R' равны входным сигналам S и R соответственно. Когда вход C равен 0, сигналы в точках S' и R' тоже равны 0 и изменить состояние триггера невозможно.

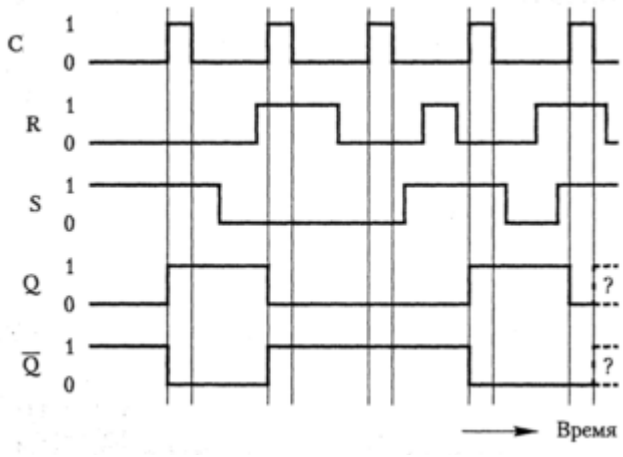
В приведенной на рис. 2.4.6,*б* таблице истинности, описывающей поведение синхронного SR-триггера. Для набора входных значений S = R = 1 значение $Q(t + 1)$ также запрещено.



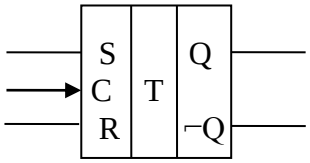
а

R	S	C	Q(t+1)	
x	x	0	Q(t)	(режим хранения)
0	0	1	Q(t)	(режим
0	1	1	1	асинхронного
1	0	1	0	триггера)
1	1	1	-	

б



в



г

Рисунок 2.4.6 – Синхронный RS – триггер: логическая схема (*а*); таблица истинности (*б*); временная диаграмма (в); графическое обозначение (*г*)

Аналитическое функционирование синхронного RS-триггера можно получить, используя карты Карно (рис. 2.4.9).

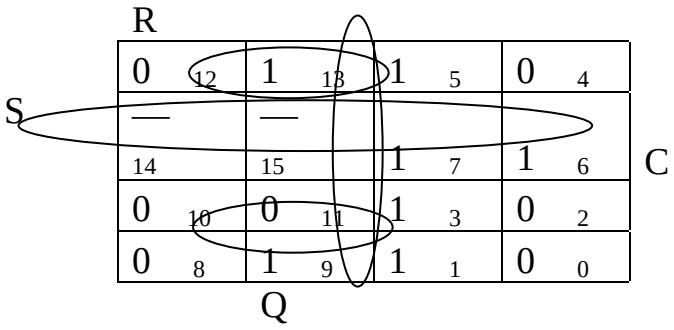


Рисунок 2.4.7 – Карта Карно логической функции синхронного RS-триггера

Уравнение функционирования имеет вид:

$Q(t+1) = S(t)\,C(t) + Q(t)\,\neg R(t) + Q(t)\,\neg C(t)$,

причем $S(t)\,R(t) = 0$.

Выберем базис на элементах И-НЕ:

$$Q(t+1) = S(t)\,C(t) + Q(t)\,\neg R(t) + Q(t)\,\neg C(t) =$$
$$= \neg\neg(S(t)\,C(t) + Q(t)\,(\neg R(t) + \neg C(t))) =$$
$$= \neg(\neg(S(t)\,C(t))\,\neg(Q(t)\,\neg(R(t)\,C(t)))).$$

Графическое изображение триггера показано на рис. 2.4.8,*г*. Его тактовый вход обозначается стрелкой →. Это стандартное обозначение, указываю-

щее, что изменение состояния триггера происходит на положительном фронте тактового сигнала. Если представляется ситуация, когда изменение происходит на отрицательном фронте, на тактовом входе стрелка отображается в обратную сторону.

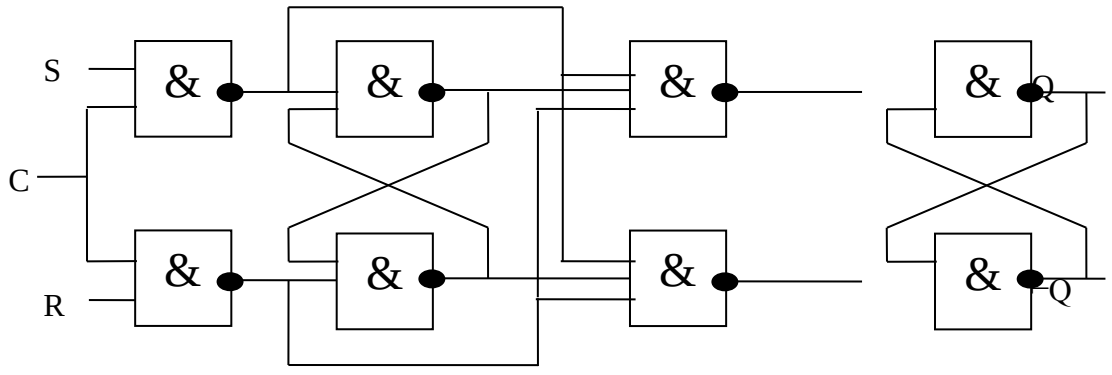
Двухтактный RS-триггер

Устойчивая работа однотактных RS-триггеров в произвольной схеме возможна только в случае, если занесение в триггер информации осуществляется после завершения передачи информации о прежнем его состоянии в другой триггер.

Такой принцип обмена информацией реализован в двухтактных RS-триггерах.

Простейшая схема двухтактного RS-триггера показана на рис. 2.4.8,*а*, состоящей из двух однотактных RS-триггеров и инвертора в цепи синхронизации. При поступлении на вход RS-триггера сигнала *C*=1 входная информация заносится в первый однотактный RS-триггер, а второй при этом будет хранить информацию, относящуюся при этом к предыдущему периоду представления. По окончании действия сигнала синхронизации, когда *C*=0, первый RS-триггер перейдет в режим хранения, а второй перепишет из него новое значение выходного сигнала. Таблица на рис. 2.4.8,*б* задает закон функционирования такого двухтактного триггера.

Двухтактный триггер изменяет свои состояния только после действия сигнала синхронизации *C*=1 (переход в режим хранения информации). Поэтому из двухтактных триггеров можно строить произвольные схемы, в том числе подавать сигналы с выхода триггера на его вход. Символ ТТ в основном поле условного обозначения указывает на двухтактный триггер (рис. 2.4.8,*в*).

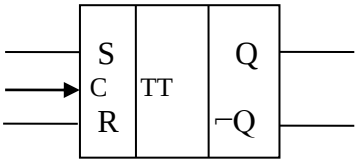


а

R	S	Q(t+1)	
0	0	Q(t)	(режим хранения)
0	1	1	(установка 1)
1	0	0	(установка 0)

1	1	- (запрещено)

б



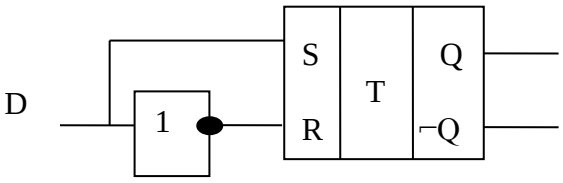
в

Рисунок 2.4.8 – Двухступенчатый RS-триггер: логическая схема (*а*); таблица истинности (*б*); графическое обозначение (*г*)

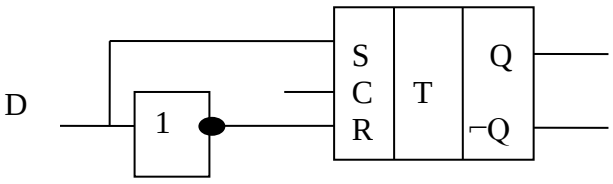
Схема RS-триггеров составляет основу для построения других триггерных схем, таких как *T*-, *D*- и *JK*-триггеров.

D-триггер

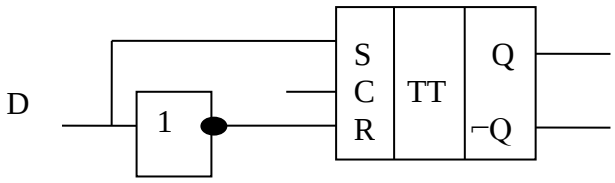
На рис. 2.4.9 показан широко используемый D-триггер, реализующий функцию временной задержки (delay). D-триггер имеет только режимы установки 1 и установки 0. D-триггер соответствует RS-триггеру, работающему только в режимах установки, т.е. с R = 1 и S = 0 либо с R = 0 и S = 1. В связи с этим несинхронизируемый D-триггер (рис. 2.4.9,*а*) не применяется, т.к. его выход будет просто повторять входной сигнал. Синхронизируемый однотактный D-триггер (рис. 2.4.9,*б*) задерживает распространение входного сигнала на полпериода. Для задержки на период (на один такт) используется двухтактный D-триггер.



а



б



в

Рисунок 2.4.9 – D-триггер: несинхронный триггер (*а*); синхронный однотактный D-триггер (*б*); двухтактный D-триггер (*в*)

Вариант построения двухтактного D-триггера показан на рис. 2.4.9,*в*. Под действием синхросигнала информация, поступающая на вход *D*, принимается в RS-триггер, но на выходе *Q* появляется с задержкой в момент времени $t + 1$:

$Q(t + 1) = D(t)$.

В схеме однотактного D-триггера сигналы S и R поступают из одного входа, обозначенного как D. В ответ на поступление тактового импульса значение на выходе *Q* становится равным 1, если $D = 1$, или сбрасывается в 0, если $D = 0$. Это означает, что при поступлении каждого нового тактового импульса сигнал на входе *D* передается на выход *Q* D-триггера и сохраняется неизменным до прихода следующего тактового импульса.

Аналитическое функционирование синхронного D-триггера можно получить из таблицы истинности (рис. 2.4.10,*а*), используя карты Карно (рис 2.4.10,*б*).

Уравнение функционирования имеет вид:

$Q(t+1) = D(t)\,C(t) + \neg C(t)\,Q(t)$.

Выберем базис на элементах И-НЕ:

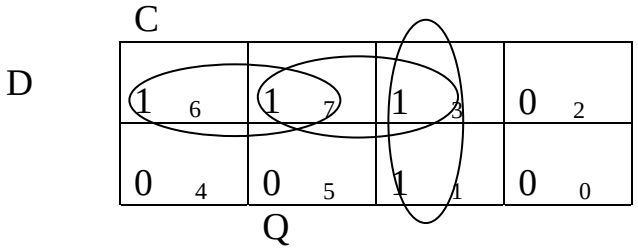
$Q(t+1) = D(t)\,C(t) + \neg C(t)\,Q(t) = \neg\neg(D(t)\,C(t) + \neg C(t)\,Q(t)) =$

$= \neg(\neg(D(t)\,C(t))\,\neg(\neg C(t)\,Q(t)))$

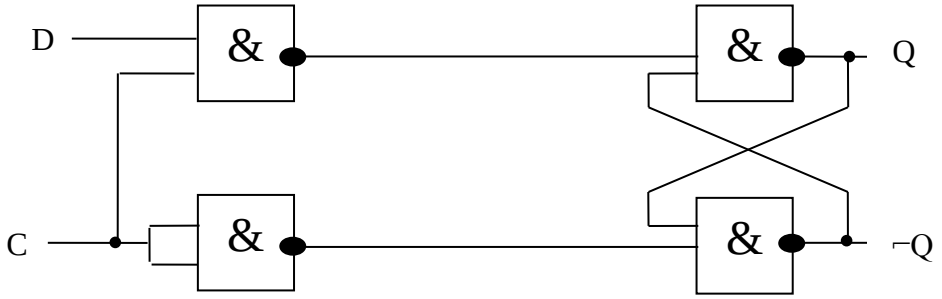
Логическая схема D-триггера показана на рис. 2.4.10,*в*.

D	C	Q(t+1)	
x	0	Q(t)	(режим хранения)
0	1	0	(установка 0)
1	1	1	(установка 1)

а



б



в

Рисунок 2.4.10 – Синхронный D-триггер: таблица истинности (*а*); карта Карно (*б*); логическая схема (*в*)

Т-триггеры

Триггер со счетным входом, Т-триггер, в простейшем случае может быть реализован с использованием двухтактного синхронного RS-триггера. Состояние Т-триггера изменяется на каждом такте, если на его вход *Т* подается значение 1. Говорят, что такой триггер «переключает» свое состояние.

Схема Т-триггера, а также его таблица истинности, графическое обозначение представлены на рис. 2.4.11.

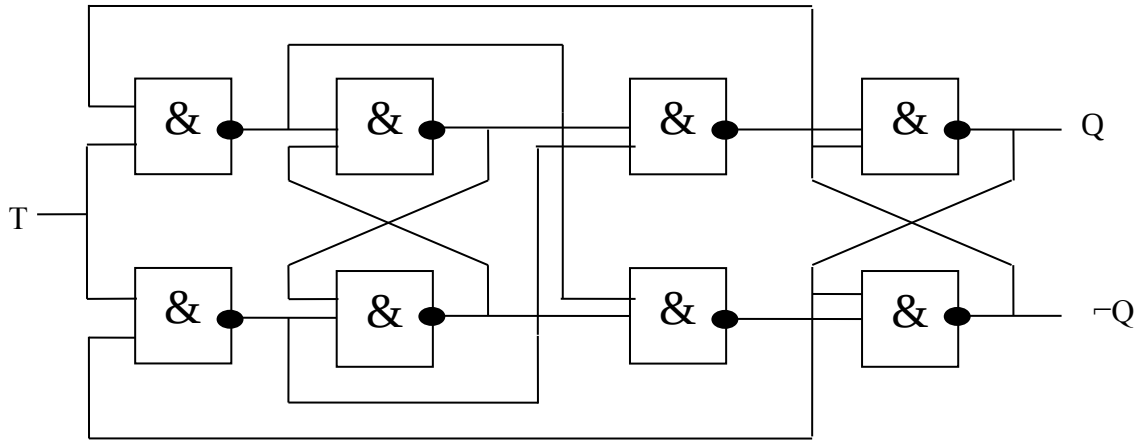
Т-триггер должен реализовать функцию вида:

$Q(t+1) = Q(t)\,\neg T(t) + \neg Q(t)\,T(t)$,

т.е. осуществлять операцию суммирования по модулю 2 сигнала состояния триггера *Q* и входного сигнала *Т*.

В схеме Т-триггера поступление сигнала $T = 1$ приводит к записи в двухступенчатый RS-триггер состояния, противоположного ранее хранимому. При этом, так как триггер двухступенчатый, на его выходе сигнал изменится только по завершению действия сигнала $T = 1$, что исключает возникновение генерации в схеме с обратной связью. Можно считать, что в данной схеме единичный входной сигнал представляется спадом сигнала $T = 1$, т.к. при лю-

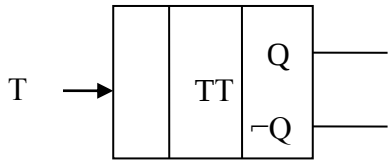
бой продолжительности сигнала $T = 1$ изменение состояния Т-триггера происходит только один раз – при снятии сигнала $T = 1$ (рис. 2.4.11,*г*).



а

T	Q(t)	Q(t+1)	Примечание
0	0	0	Постоянный импульс
0	1	1	Меняюшийся
1	0	1	импульс
1	1	0	Постоянный импульс

б



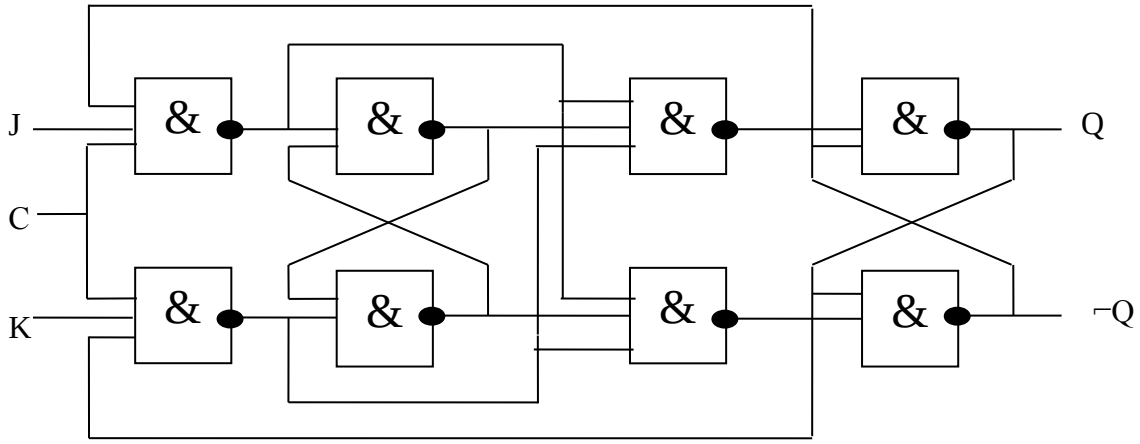
в

Рисунок 2.4.11 – Т-триггер: логическая схема (*а*); таблица истинности (*б*); графическое обозначение (*в*)

JK-триггеры

Распространенным типом триггера в системах интегральных логических элементов является универсальный двухтактный JK-триггер. JK-триггер обладает чертами триггеров SR и T. Его схема, таблица истинности и обозначение приведены на рис. 2.4.12. Первые три строки в таблице истинности JK-триггера определяют его поведение, аналогичное поведению RS-триггера при $C = 1$, так что входы J и K соответствуют входам S и R соответственно. При входном сигнале $J = K = 1$ следующее состояние триггера определяется как

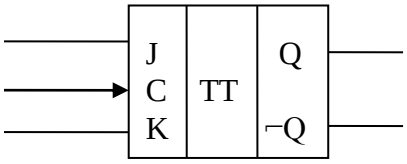
дополнение его текущего состояния. Это означает, что когда $J = K = 1$, триггер действует как переключатель, изменяя свое текущее состояние на противоположное. При $C = 0$ триггер находится в режиме хранения. Триггеры JK могут применяться для разных целей. В частности, их подобно D-триггерам можно использовать для хранения данных. На их основе также можно создавать счетчики, поскольку при соединении терминалов J и K они ведут себя как Т-триггеры.



а

K	J	C	Q(t+1)	
x	x	0	Q(t)	(режим хранения)
0	0	1	Q(t)	(режим
0	1	1	1	синхронного
1	0	1	0	триггера)
1	1	1	¬ Q(t)	(режим Т-триггера)

б



в

Рисунок 2.4.12 – JK-триггер: логическая схема (*а*); таблица истинности (*б*); графическое обозначение (*в*)

2.5 Логическое проектирование

Основные процедуры логического проектирования (синтеза) цифровых устройств (Рис. 2.5.1):

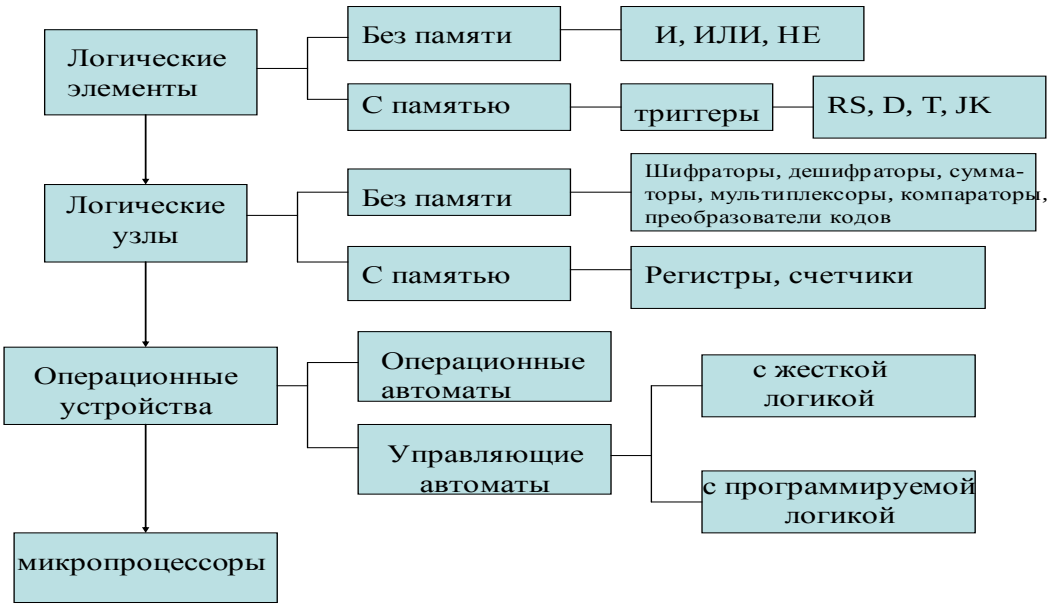


Рис. 2.5.1. Структура набора для логического проектирования

Логические элементы без памяти (ЛЭ без П): И, ИЛИ, НЕ, И- НЕ, ИЛИ – НЕ;

Логические элементы с памятью (ЛЭ с П): триггеры;

Логические узлы без памяти (ЛУ без П): сумматор, мультиплексор, демультиплексор, шифратор, дешифратор.

Логические узлы с памятью (ЛУ с П):

регистр, счётчик;

МП – микропроцессор;

ЖЛ – жёсткая логика;

ПЛ – программируемая логика.

Таким образом, перед тем как начать логическое проектирование, необходимо выбрать тип логики (жёсткая или программируемая).

Набор булевых функций: $N_н = 2^n$;

Общее количество функций: $N = 2^{2^n}$.

Задача синтеза:

1. Уяснить задачу.
2. Определить число аргументов и число функций.
3. Составить таблицу истинности.
4. Получить булеву функцию.
5. Произвести её минимизацию.

6. Выбрать логический базис.

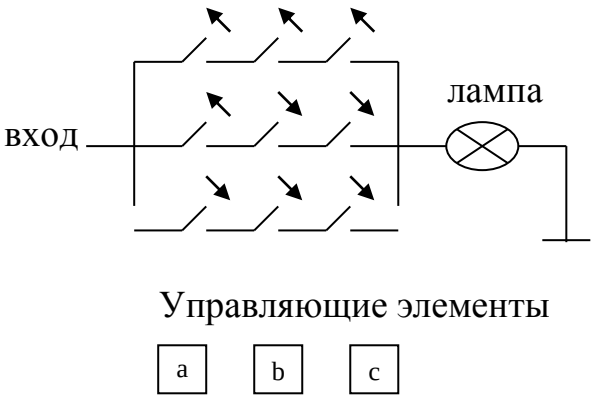
7. Синтезировать логическую схему.

8. Выполнить её проверку.

Пример: спроектировать устройство, определяющее число, которое делится на три.

Таблица истинности такого устройства (рис. 2.5.2).

n	a	b	c	Y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0



Транзистор – элемент НЕ.

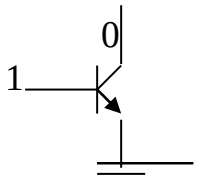
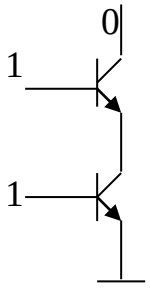
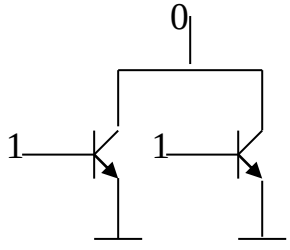


Рис. 2.5.2 – Таблица истинности и схема

Два транзистора, соединённых последовательно, – элемент И-НЕ.



Два транзистора, соединённых параллельно, – элемент ИЛИ- НЕ.



Получим булеву функцию: $Y = a*b*\overline{c} + \overline{a}*b*c + a*\overline{b}*c$.

Нанесем булеву функцию на карту Карно (рис. 2.5.3).

Произведем минимизацию, вариантов минимизации в данном пример нет.

P_i	a_i	b_i	S_i	P_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

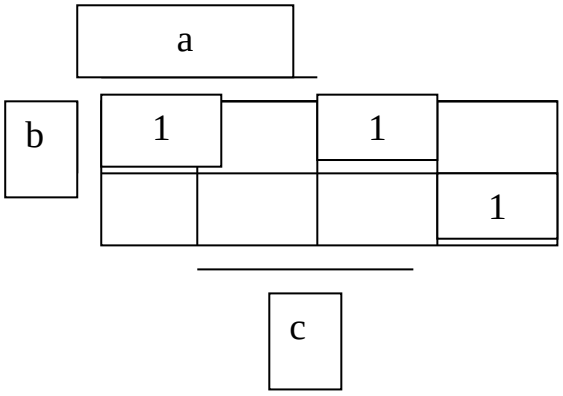


Рис. 2.5.3 – Карта Карно для булевой функции

Выберем базис на элементах И-НЕ, используя искусственный прием (двойная инверсия от аргумента равна аргументу) и теорему Де Моргана, получим.

$$Y = \overline{a*b*c} + a*\overline{b*c} + a*b*\overline{c} = \overline{\overline{a*b*c} * a*\overline{b*c} * a*b*\overline{c}}$$

Синтезируем схему на элементах И-НЕ, рис 2.5.4.

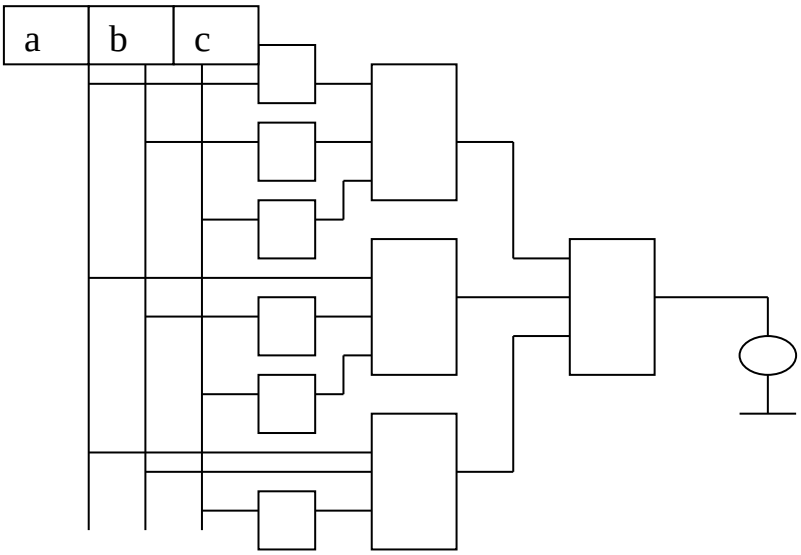


Рис. 2.5.4 – Логическая схема на элементах И-НЕ

2.6 Логические узлы

2.6.1 Логические узлы без памяти

Сумматор

Сумматором называется узел ЭВМ, выполняющий арифметическое суммирование кодов чисел. Обычно это комбинация одноразрядных суммирующих схем.

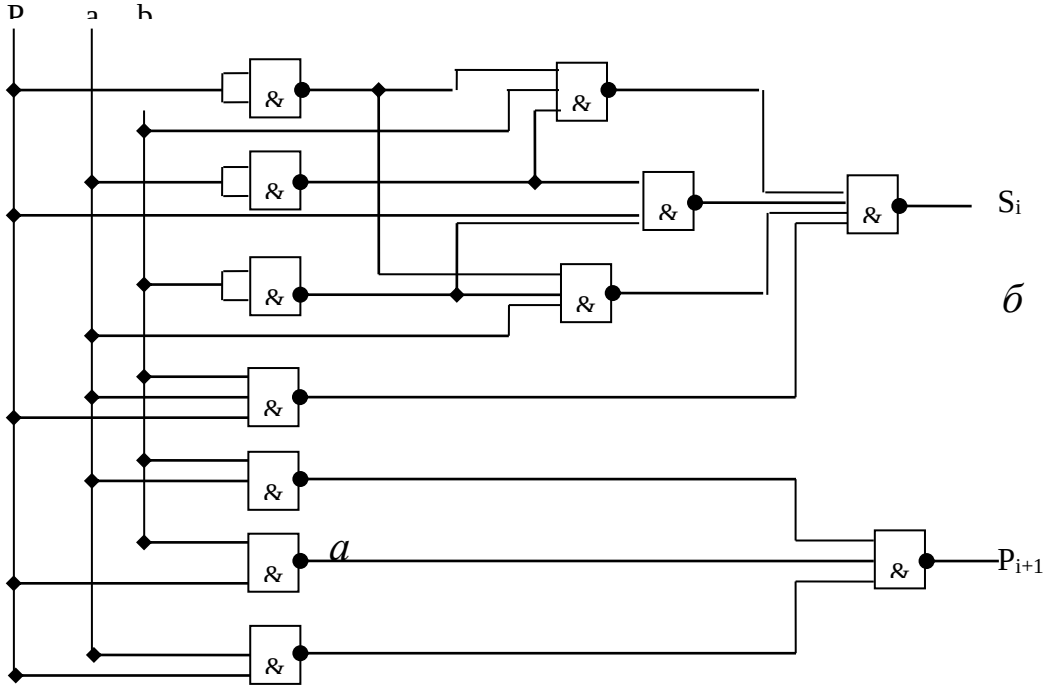


Рис. 2.6.1 – Функциональная схема (а); одноразрядный сумматор: таблица истинности (б)

При сложении двух чисел независимо от системы счисления в каждом разряде производится сложение трех цифр: двух цифр данного разряда слагаемых и цифры (1 или 0) переноса из соседнего младшего разряда. Полусумматор такого входа не имеет, поэтому суммирует одноразрядные числа

По таблице истинности получим булевы функции

$$S_i = a_i \neg b_i \neg P_i + \neg a_i\, b_i \neg P_i + \neg a_i \neg b_i\, P_i + a_i\, b_i\, P_i;$$

$$P_{i+1} = a_i\, b_i \neg P_i + a_i \neg b_i\, P_i + \neg a_i\, b_i\, P_i + a_i\, b_i\, P_i,$$

где P_i – цифра переноса из предыдущего (младшего) разряда; a_i, b_i – цифры слагаемых в данном разряде; S_i – сумма; P_{i+1} – цифра переноса в старший разряд.

Выражение для цифры переноса в следующий разряд может быть приведено к более простому виду:

$$P_{i+1} = a_i\, b_i + a_i\, P_i + b_i\, P_i\,.$$

Параллельный (многоразрядный) сумматор может быть составлен из одноразрядных сумматоров, число которых равно числу разрядов слагаемых, путем соединения выхода, где формируется сигнал переноса данного разряда, с входом для сигнала переноса для соседнего старшего разряда (рис. 2.6.2).

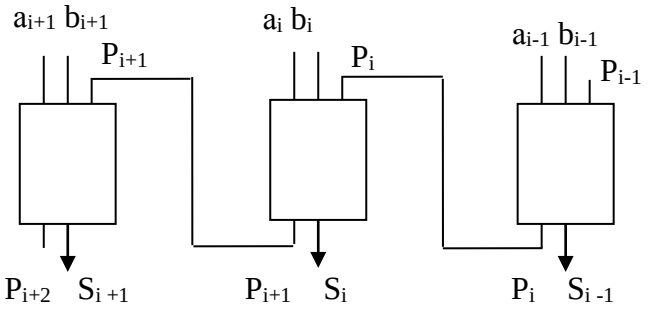


Рис. 2.6.2 – Параллельный сумматор с последовательным переносом

Компаратор

Компаратором называется узел ЭВМ, выполняющий поразрядное сравнение кодов чисел. При сравнении двух чисел независимо от системы счисления сравнение ведется со старшего разряда. Если во всех разрядах результат сравнения оказался положительным (числа не отличаются), то на выходе имеем 1, в противном случае (хотя бы в одном разряде числа различны) – 0. В связи с этим можно составить таблицу истинности для сравнения одноразрядных чисел (рис. 2.6.3).

a_1	a_2	f
0	0	1
0	1	0
1	0	0
1	1	1

Рис. 2.6.3 –Таблица истинности компаратора

Дешифратор

Дешифратором называется комбинационная схема с несколькими входами и выходами, преобразующая код, подаваемый на входы, в сигнал на одном из выходов. Если на входы дешифратора подаются двоичные переменные, каждая из которых может принимать значения 0 или 1, то на одном из выходов дешифратора вырабатывается сигнал 1, а на остальных выходах сохраняются сигналы 0.

В общем случае дешифратор с n входами имеет 2^n выходов, т.к. n-разрядный код входного слова может принимать 2^n различных значений и каждому из этих значений соответствует сигнал 1 на одном из выходов дешифратора.

На выходах дешифратора с номерами 0, 1, 2, …, 2^n –1 вырабатываются значения булевых функций соответственно:

$$f_0 = \overline{x_0}\,\overline{x_1}\ldots\overline{x_{n-1}}\,\overline{x_n},$$

$$f_1 = \overline{x_0}\,\overline{x_1}\ldots\overline{x_{n-1}}x_n\ ,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

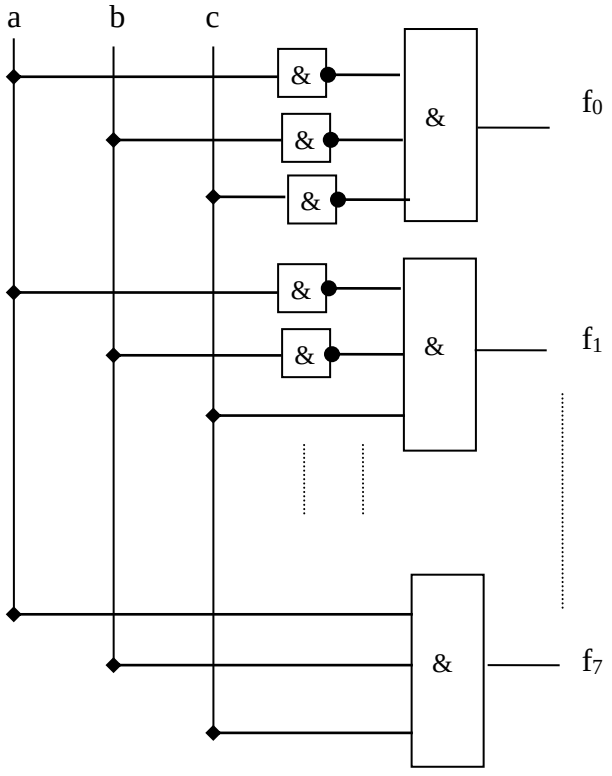
$$f_{2^n-1} = x_0\,x_1\ldots x_{n-1}x_n.$$

Дешифраторы устанавливаются в схемах ЭВМ на выходах регистров или счетчиков и служат для преобразования кода слова, находящегося в регистре (в счетчике), в управляющий сигнал на одном из выходов дешифратора.

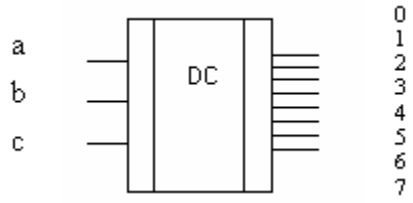
На рис. 2.6.4 показаны таблица истинности, функциональная схема и условное обозначение дешифратора.

a b	c	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	0 0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

а



б



в

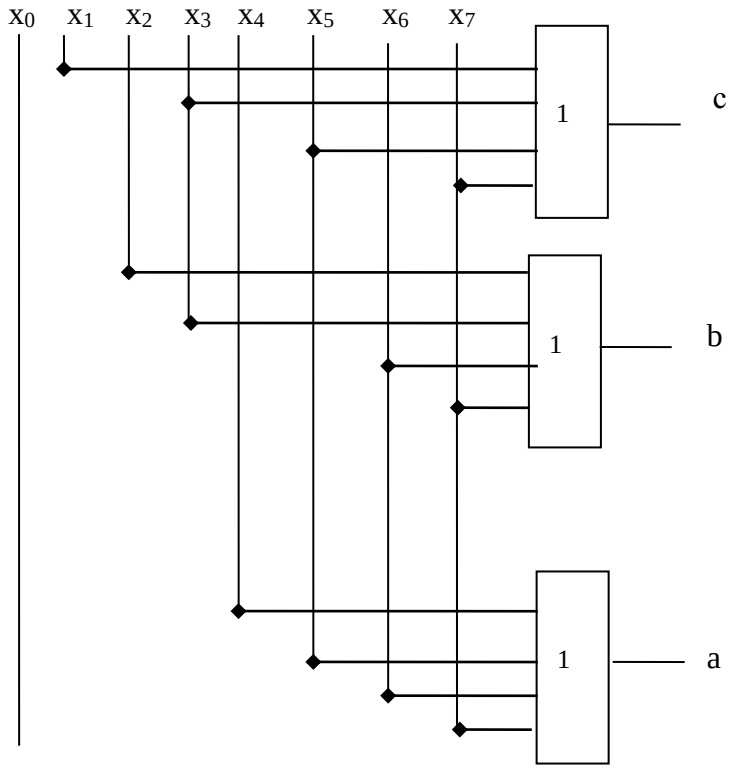
Рис. 2.6.4 – Дешифратор: таблица истинности (а); функциональная схема (б); условное обозначение (в)

Шифратор

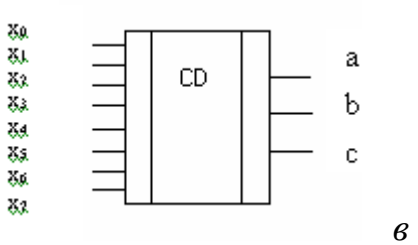
Шифратором называется комбинационная схема с несколькими входами и выходами, преобразующая сигнал, подаваемый на

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	a	b	c
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

а



б



в

Рис. 2.6.5 – Шифратор: таблица истинности (*а*); функциональная схема (*б*); условное обозначение (*в*)

входы, в код на одном из выходов. Действие шифратора обратно действию дешифратора. Если на входы шифратора подаются двоичные переменные, каждая из которых может принимать значения 0 или 1, то на одном из выходов дешифратора вырабатывается сигнал 1, а на остальных выходах сохраняются сигналы 0.

В общем случае шифратор с 2^n входами имеет n выходов.

На выходах шифратора с номерами 0, 1, 2, …, 7 вырабатываются значения булевых функций соответственно:

$$f_0 = x_1 + x_3 + x_5 + x_7,$$
$$f_1 = x_2 + x_3 + x_6 + x_7 \ ,$$
$$f_3 = x_4 + x_5 + x_6 + x_7.$$

На рис. 2.6.5 показаны таблица истинности, функциональная схема и условное обозначение шифратора.

Мультиплексор

Существует класс логических узлов, предназначенный для выбора одного из n входов данных, значение которого передается на выход схемы. Выбор осуществляется на основе значений, поданных на так называемые входы выбора. Такие схемы называются мультиплексорами, рис. 2.6.6.

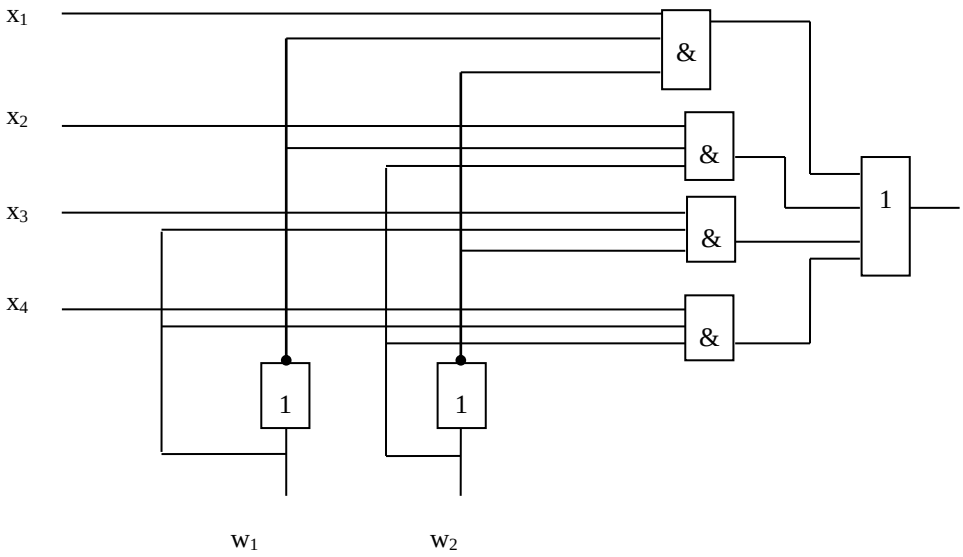
В зависимости от цифровой комбинации, поступающей на управляющие входы, мультиплексор подключает ту или иную входную цепь к выходу.

У данной схемы два входных сигнала выбора. Четыре возможные комбинации их значений используются для выбора одного из входов данных, значение которого передается на выход. Такую же структуру будут иметь и большие мультиплексоры, в которых k входных сигналов выбора используются для соединения одного из 2^k входов данных с выходом. Типичной областью применения мультиплексоров является фильтрация данных, поступающих из множества разных источников. В частности, с помощью шестнадцати четырехвходовых мультиплексоров можно реализовать загрузку 16-разрядного регистра данных из одного четырех источников.

Демультиплексор

Демультиплексор – противоположный по действию мультиплексору логический узел, предназначенный для выбора одного из *n* выходов, на который передается значение входного сигнала. Условное обозначение демультиплексора приведено на рис. 2.6.7.

В зависимости от цифровой комбинации, поступающей на управляющие входы, мультиплексор подключает ту или иную выходную цепь ко входу



а

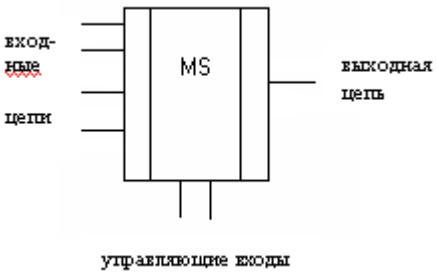


Рис. 2.6.6 – Мультиплексор: функциональная схема (*а*); условное обозначение (*б*)

.

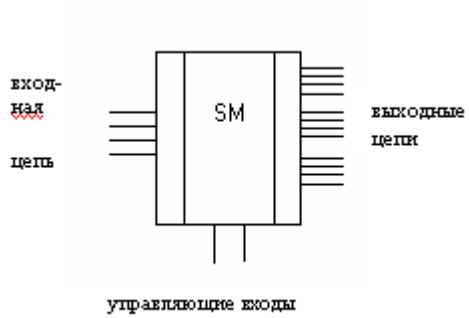


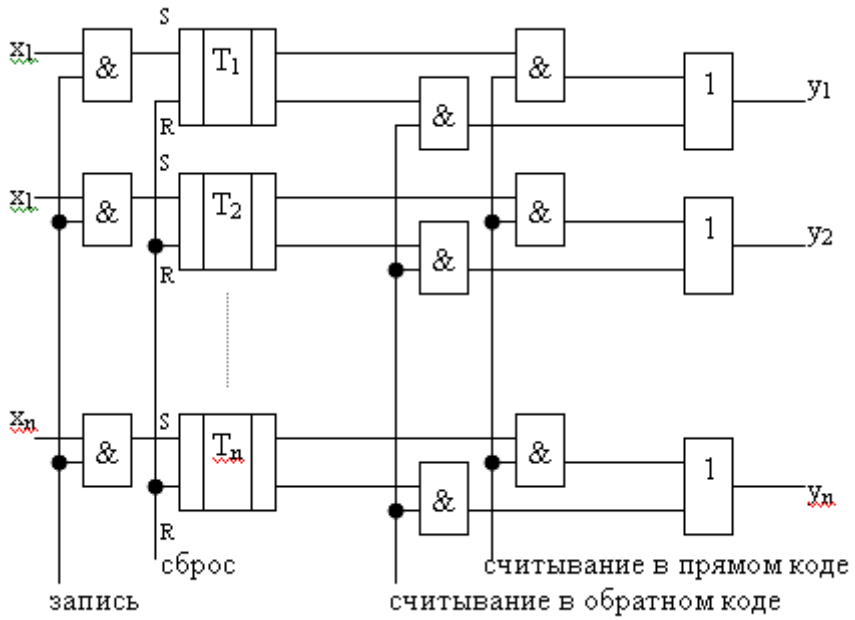
Рис. 2.6.7 – Демультиплексор: условное обозначение

2.6.2 Логические узлы с памятью

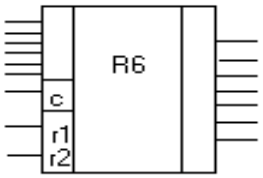
Отдельный триггер может использоваться для хранения одного бита информации. Однако для преобразователей, обрабатывающих слова данных, состоящих из множества битов (обычно 64), удобнее объединить группу триггеров в стандартную структуру, называемую регистром. Регистры предназначены для запоминания слова и выполнения над ним некоторых логических преобразований. Число триггеров в регистре соответствует количеству разрядов в слове.

Параллельные регистры осуществляют запись и считывание в параллельном коде (рис. 2.6.8).

В ходе обработки цифровых данных часто требуется сдвинуть или циклически прокрутить значения группы битов данных. Реализуются эти операции аппаратно. Простейшим механизмом для их выполнения является параллельный регистр, содержимое которого легко может быть сдвинуто вправо или влево на одну позицию за один раз. В качестве примера рассмотрим 3-разрядный сдвиговый регистр, показанный на рис. 2.6.9. Он состоит из трех D-триггеров, соединенных таким образом, что каждый тактовый импульс вызывает перемещение содержимого триггера в следующий за ним триггер, в результате чего получается «сдвиг вправо», обозначаемый → (1), сдвиг влево обозначается ← (0). Данные последовательно «вдвигаются» в регистр и «выдвигаются» из него. Для выполнения циклического смещения данных достаточно соединить выход и вход.



а



б

Рис. 2.6.1 – Параллельный регистр: логическая схема (*а*);
условное обозначение (*б*)

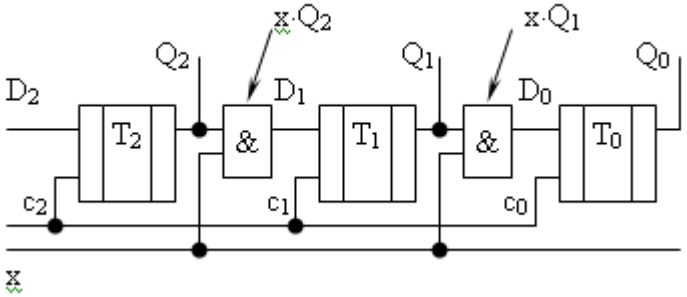
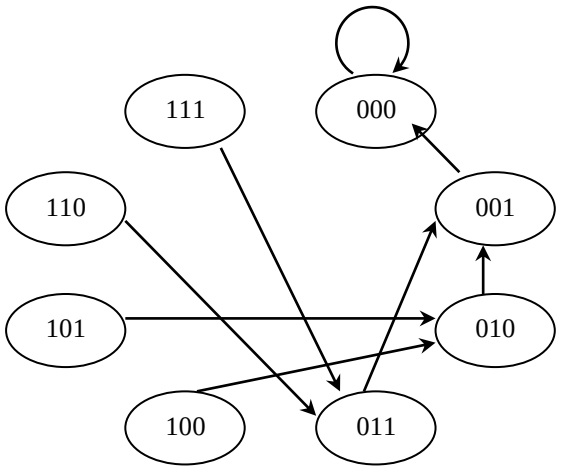


Рис. 2.6.9 – Последовательный регистр: логическая схема

Граф переходов при сдвиге вправо, таблица истинности и карта Карно показаны на рис. 2.6.10.



а

№	$Q_2(t)$	$Q_1(t)$	$Q_0(t)$	$Q_2(t+1)$	$Q_1(t+1)$	$Q_0(t+1)$	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	1
3	0	1	1	0	0	1	0	0	1
4	1	0	0	0	1	0	0	1	0
5	1	0	1	0	1	0	0	1	0
6	1	1	0	0	1	1	0	1	1
7	1	1	1	0	1	1	0	1	1

б

x

Q_2					Q_0
	1 12	1 13	0 5	0 4	
	1 14	1 15	0 7	0 6	
	0 10	0 11	0 3	0 2	
	0 8	0 9	0 1	0 0	

Q_1

x

Q_2					Q_0
	0 12	0 13	1 5	0 4	
	0 14	0 15	1 7	0 6	
	0 10	1 11	0 3	0 2	
	0 8	1 9	0 1	0 0	

Q_1

$D_2 = 0$
$D_1 = x \cdot Q_2$
$D_0 = x \cdot Q_1$

в

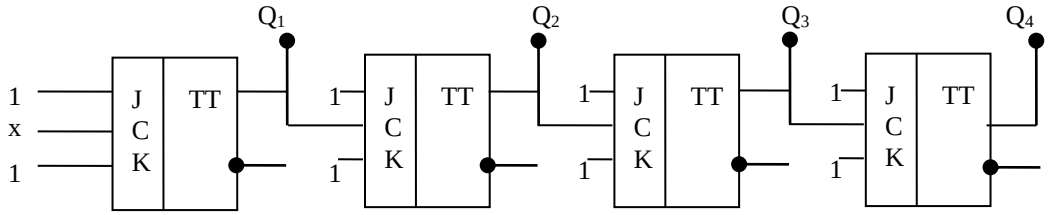
Рис. 2.6.10 – Последовательный регистр: граф переходов (*а*);

таблица истинности (*б*); карты Карно (*в*)

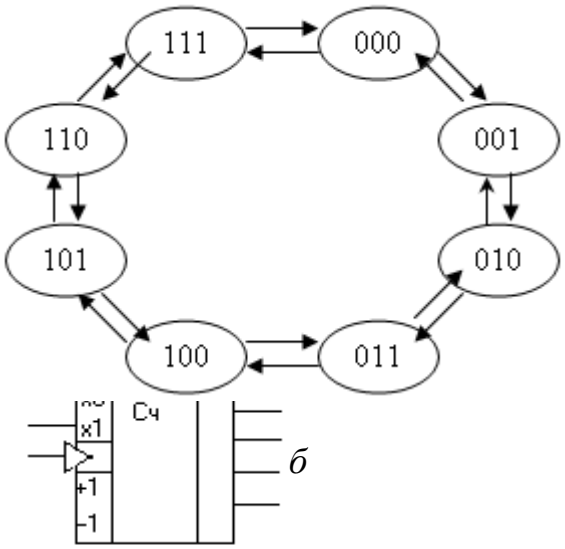
Счетчиком называется типовой узел ЭВМ, предназначенный для подсчета числа входных сигналов. Счетчики используются в ЭВМ для образования последовательностей адресов команд, для счета количества циклов выполнения операций и т.п.

Счетчики принято подразделять на суммирующие, вычитающие и реверсивные.

На рис. 2.6.11 показана схема несинхронного четырехразрядного двоичного суммирующего счетчика, предназначенного для



а



б

x	Q4	Q3	Q2	Q1	x	Q4	Q3	Q2	Q1
0	0	0	0	0	8	1	0	0	0
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	1

в

г

Рис. 2.6.11 – Несинхронный двоичный счетчик: функциональная схема (*а*); граф переходов (*б*); таблица истинности (*в*); условное обозначение (*г*)

подсчета импульсов, поступающих на его входы. Результатом является кодовая комбинация, соответствующая числу импульсов. Таблица истинности показывает состояния, в которых находятся триггеры счетчика при воздействии серии входных сигналов x (при $x = 1$ счетчик считает в прямом направлении, при $x = 0$ – в обратном).

На входы JK-триггеров подаются сигналы 1. Выход каждого предыдущего триггера соединен с входом синхронизации *C* последующего триггера. По спаду единичного входного сигнала изменяется состояние триггера младшего разряда счетчика на противоположное (т.е. реализуется сложение по модулю 2 в этом разряде). В последующих разрядах аналогичное действие производит сигнал переноса.

Контрольные вопросы

1. Какая связь между понятием функциональной полноты и элементами И-НЕ и ИЛИ-НЕ?
2. Какие существуют критерии минимизации булевых функций?
3. В чем отличие синтеза логических схем в базисе И-НЕ и ИЛИ-НЕ?
4. Что такое запрещенные состояния для RS-триггера и можно ли их использовать при минимизации?
5. Почему JK-триггер является универсальным?
6. Чем мультиплексор отличается от коммутатора?

Глава 3. АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭВМ

3.1 Представление информации в ЭВМ

В ЭВМ на машинном уровне обеспечивается обработка данных трех типов: чисел, символов, логических значений. Адреса представляются целыми числами. Данные других типов на уровне аппаратуры не обрабатываются а обрабатываются на программном уровне путем выполнения команд, возбуждающих в АЛУ обработку данных указанных трех типов.

Число A в позиционной системе счисления можно представить:

$$A = a_0 \sum_{k=1}^{m} a_k q^{n-k},$$

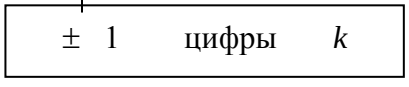
где $a_k \in \{0,1,\ldots.q-1\}$, q – основание системы счисления, обычно принимает значения q=2(8,10,16), a_0 – знак числа, a_k – цифры числа, q^{n-k} – вес цифры в зависимости от ее позиции, n – количество цифр в целой части числа, m – общее количество цифр.

Пример: A=217,37 – десятичное число, q=10, n=3, m=5.

Множество чисел можно разделить на два подмножества – целые и вещественные. Основной формой представления чисел в ЭВМ являются двоичные числа.

Целые двоичные числа в ЭВМ представлены в следующем формате:

Веса: 2^k 2^{k-1} … 2^1 2^0.



Диапазон представления: $\pm(2^k-1)$. Основная проблема заключается в переполнении разрядной сетки при выполнении арифметических операций, если результат операции C удовлетворяет условию $|C|\geq 2^k$. Переполнение при сложении возможно обнаружить, если знаки слагаемых различны. В результате переполнения знак суммы изменяется и не совпадает со знаком слагаемых. Переполнение в ЭВМ также обнаруживается путем анализа битов переноса в знаковый разряд сумматора p_1 и из знакового разряда p_0. Условие переполнения: $p_0 \neq p_1$.

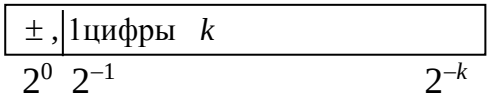
p_0	p_1	$\oplus$
0	0	0
0	1	1
1	0	1
1	1	0

1 – переполнение

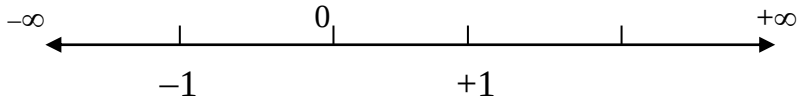
Сигнал переполнения формируется схемой сложения $\oplus$ по mod 2.

Для представления вещественных чисел используются два способа: правильная дробь и плавающая запятая (формат чисел с плавающей запятой).

Двоичная правильная дробь имеет формат:



Диапазон представления: $\pm(1-2^{-k})$. Наименьшее число, отличное от нуля, – 2^{-k}=0,0…01. Числа, меньшие чем 2^{-k}, рассматриваются как машинный ноль. Таким образом, если целые числа в ЭВМ представлены точно (без погрешности), то действительные числа – с дискретностью $\Delta=2^{-k}$. То есть разрядность числа k определяет точность представления действительных чисел в k-разрядной сетке. $\Delta=2^{-k}$ – это дискретность, с которой идет «прореживание» множества действительных чисел в k-разрядной сетке.



Пример: k=15, $\Delta=2^{-15}\approx 0{,}3\cdot 10^{-4}$, т.е. точность представления приблизительно три десятичных знака.

Особенности операций над правильными дробями: переполнение разрядной сетки при выполнении операций сложения, вычитания и деления. Условие переполнения $|C|\geq 1=2^0$, переполнение разрядной сетки при выполнении операции умножения не происходит. Общая для двух форматов с фиксированной запятой – и для целых чисел, и для правильной дроби – особенность: представление отрицательных чисел не в прямом коде (как для положительных чисел), а в дополнительном (или обратном) коде.

Дополнительный код для представления отрицательных чисел используется для того, чтобы с помощью двоичного комбинационного сумматора стало возможным складывать не только положительные числа и числа без знаков, но и числа с отрицательными знаками.

Двоичные числа с плавающей запятой – универсальный формат, который можно использовать для представления любых чисел – целых, правильных дробей и неправильных дробей в широком диапазоне. Для представления двоичного числа с плавающей запятой используется формула:

$$A=\pm M\cdot q^{\pm P},$$

где $q=2^h$, (h=1или 4) – основание системы счисления с плавающей запятой, M – **мантисса (**формат представления – правильная дробь), P – **порядок числа (**формат представления – целое число). Формат числа с плавающей запятой:

m_0	m_1 M m_n	p_0	$1Pk$

Здесь: M – мантисса числа, P – порядок числа, m_0 – знак числа, p_0 – знак порядка. Диапазон представления (если мантисса нормализована):

$d^{-1}\ d^{-P\max} \le |A| \le (1-2^{-n})d^{+P\max}$.

Здесь: $P_{\max} = (2^k-1)$ – максимальный порядок числа.

Диапазон представления на числовой оси:

Двоично-десятичные числа (BCD) – можно использовать для представления десятичных чисел. Суть: каждой десятичной цифре 0,1, …, 9 ставится в соответствие двоичный код 0000, …, 1001.

Для представления двоично-десятичного числа используются упакованные и не упакованные форматы произвольной длины (машинный элемент – строка байтов). Упакованный формат: один байт содержит две десятичных цифры (по 4 бита каждая). Не упакованный формат: один байт – одна десятичная цифра (4 бита из 8 не используются). Количество байтов зависит от количества десятичных цифр и формата числа.

Особенности BCD-чисел: операции над ними выполняются обычно при помощи двоичного сумматора, поэтому их результат требует коррекции. Коррекция осуществляется путем сложения каждой цифры результата с цифрой 6. Для этого в систему команд ЭВМ вводятся специальные команды коррекции. Другой вариант решения этой проблемы – использова-

ние корректирующих кодов, например с избытком три, для представления BCD-чисел.

Для представления символов текста используется формат данных типа строка байтов. В этом формате каждому символу ставится в соответствие байт строки байтов. Первый символ в виде ASCII-кода располагается в первом (младшем) байте, второй символ – во втором байте, адрес которого на 1 больше, и т.д.

Для представления логических значений истина – 1, ложь – 0 используется строка битов переменной длины или битовое поле фиксированной длины (например, байт).

Представление чисел в прямом, обратном и дополнительном кодах

В ЭВМ для выполнения арифметических операций применяют специальные коды для представления чисел. При помощи этих кодов упрощается определение знака результата операции. Операция вычитания (или алгебраического сложения) чисел сводится к арифметическому сложению кодов, облегчается выработка признаков переполнения разрядной сетки. В результате упрощаются устройства ЭВМ, выполняющие арифметические операции.

Для представления чисел со знаком в ЭВМ применяют прямой, обратный и дополнительный коды.

Код представляется как число без знака, а диапазон чисел разбивается на два поддиапазона. Один из них представляет положительные числа, другой – отрицательные. Разбиение выполняется таким образом, чтобы принадлежность к поддиапазону определялась максимально просто.

При формировании кодов значение старшего разряда указывает на знак представляемых чисел, т.е. при таком кодировании старший разряд – знаковый (бит знака), остальные разряды – информационные.

Прямой код

Это обычный двоичный код. Если двоичное число является положительным, то бит знака равен 0, если двоичное число – отрицательное, то бит знака равен 1. Цифровые разряды прямого кода содержат модуль представляемого числа, что обеспечивает наглядность представления чисел в прямом коде (ПК).

Однобайтовое представление двоичного числа. Пусть это будет $28_{(10)}$. В двоичном формате – $0011100_{(2)}$ (при однобайтовом формате под величину числа отведено 7 разрядов). Двоичное число со знаком будет выглядеть так

$+28_{(10)}=00011100_{(2)}$ $-28_{(10)}=10011100_{(2)}$

а *б*

Рис. 3.1.1 – Формат двоичного числа:
а – положительное число; *б* – отрицательное

Сложение в прямом коде чисел, имеющих одинаковые знаки: числа складываются, и сумме присваивается знак слагаемых. Значительно более сложным является алгебраическое сложение в прямом коде чисел с разными знаками. В этом случае приходится определять большее по модулю число, производить вычитание модулей и присваивать разности знак большего по модулю числа. Такую операцию значительно проще выполнять, используя обратный и дополнительный коды.

Обратный код

В обратном коде (ОК), так же как и в прямом коде, для обозначения знака положительного числа используется бит, равный нулю, и знака отрицательного числа – единице. Обратный код отрицательного двоичного числа формируется дополнением модуля исходного числа нулями до самого старшего разряда модуля, а затем поразрядной заменой всех нулей числа на единицу и всех единиц на нули (инвертирование). В знаковом разряде обратного кода у положительных чисел будет 0, а у отрицательных – 1.

На рис. 3.1.2 приведен формат однобайтового двоичного числа в обратном коде.

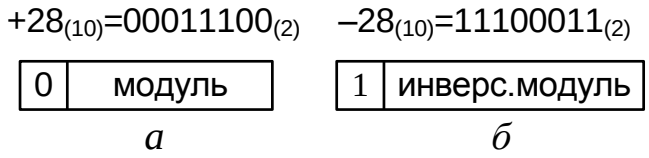


Рис. 3.1.2 – Формат двоичного числа со знаком в обратном коде:
а – положительное число; *б* – отрицательное

В общем случае ОК является дополнением модуля исходного числа до наибольшего числа без знака, помещенного в разрядную сетку.

Алгоритм формирования ОК позволяет унифицировать операции сложения и вычитания в АЛУ, которые в прямом коде выполняются поразному. Работа с ОК имеет ряд проблем:

- при выполнении операции возникают два нуля: +0 и –0, т.е. в прямом коде (в котором представлены положительные числа) имеет место (+0) = 000...0, а в обратном коде (в котором представлены отрицательные числа) – (–0) = 111...1;

- в операциях сложения и вычитания требуется дополнительная операция по прибавлению бита переноса в младший разряд суммы.

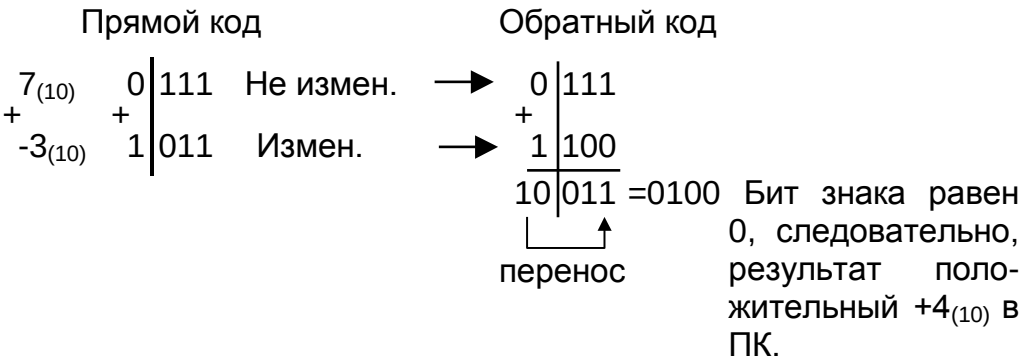
Алгоритм сложения в ОК включает в себя:

- сложение кодов, включая знаковый разряд;

- прибавление переноса к МЗР (младшему значащему разряду) суммы.

Пример:

Вычислить $7_{(10)} - 3_{(10)}$.



Дополнительный код. В современных ЭВМ большинство операций выполняется в дополнительном коде. Дополнительный код (ДК) строится следующим образом. Сначала формируется обратный код (ОК), а затем к младшему разряду (МЗР) добавляют 1. При выполнении арифметических операций положительные числа представляются в прямом коде (ПК), а отрицательные числа – в ДК, причем обратный перевод ДК в ПК осуществляется аналогичными операциями в той же последовательности. На рис. 3.1.3 рассмотрена цепь преобразований числа из ПК в ДК и обратно в двух вариантах.

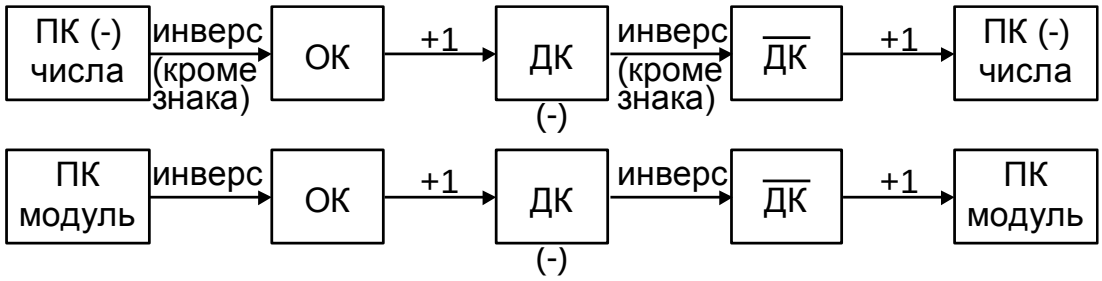


Рис. 3.1.3 – Два варианта преобразования чисел из ПК в ДК и обратно

Пример:
Число $-5_{(10)}$ перевести в ДК и обратно (первый вариант).

1	101	ПК (-)	1	011	ДК (-)
1	010	ОК (-)	1	100	$\overline{ДК}$ (-)
+	1		+	1	
1	011	ДК (-)	1	101	ПК (-)

Пример:
Число $-5_{(10)}$ перевести в ДК и обратно (второй вариант).

0	101	мод. ПК		1	011	ДК (-)	
1	010	ОК (-)		0	100	$\overline{\text{ДК}}$	
+	1			+	1		
1	011	ДК (-)		0	101	мод. ПК	

ДК для записи отрицательных чисел перекрывает диапазон десятичных чисел от 2^{k-1} до $+2^{k-1}-1$, где k – число используемых двоичных разрядов, включая знаковый.

В ЭВМ используется быстрый способ формирования ДК. Его суть заключается в следующем. Двоичное число в ПК просматривается от МЗР к СЗР. Пока встречаются нули, их копируют в одноименные разряды результата. Первая встретившаяся единица также копируется в соответствующий разряд, а каждый последующий бит исходного числа заменяется на противоположный (0 – на 1, 1 – на 0).

Пример:

Число $-44_{(10)}$ ($10101100_{(2)}$) перевести в ДК и обратно.

Проверка:

1	0101	100	ПК	1	0101100	ПК	
	инвертируется	сохраняется			инверсия		
	⇩	⇩		1	1010011	ОК	
1	1010	100	ДК	+	1		
	инвертируется	сохраняется		1	1010100	ДК	
	⇩	⇩					
1	0101	100	ПК				

3.2 Алгоритмы сложения и вычитания

При выполнении арифметических операций в современных ЭВМ используется представление положительных чисел в прямом коде (ПК), а отрицательных – в обратном (ОК) или в дополнительном (ДК) кодах. Это можно проиллюстрировать схемой на рис. 3.3.1.

ПК	0	Мод.дв.числа
ОК	0	Мод.дв.числа
ДК	0	Мод.дв.числа

а

1	Мод.дв.числа
1	$\overline{\text{Мод}}$.дв.числа
1	ОК+$1_{\text{МЗР}}$

б

Рис. 3.2.1 – Представление чисел в ЭВМ:
а – положительное число; *б* – отрицательное число

При алгебраическом сложении двух двоичных чисел, представленных обратным (или дополнительным) кодом, производится арифметическое суммирование этих кодов, включая разряды знаков. При возникновении пе-

реноса из разряда знака единица переноса прибавляется к МЗР суммы кодов при использовании ОК и отбрасывается при использовании ДК. В результате получается алгебраическая сумма в обратном (или дополнительном) коде.

При алгебраическом сложении чисел со знаком, результатом также является число со знаком. Суммирование происходит по всем разрядам, включая знаковые, которые при этом рассматриваются как старшие. При возникновении переноса из старшего разряда единица переноса отбрасывается и возможны два варианта результата:

- знаковый разряд равен нулю: результат – положительное число в ПК;
- знаковый разряд равен единице: результат – отрицательное число в ДК.

Для определения абсолютного значения результата его необходимо инвертировать, затем прибавить единицу.

Пример.

Вычислить алгебраическую сумму 58 – 23.

$58_{(10)} \rightarrow 0011\ 1010_{(2)}$ - ПК

$-28_{(10)} \rightarrow 1001\ 0111_{(2)}$ - ПК

$1110\ 1001_{(2)}$ - ДК

Число отрицательное – необходимо перевести в ДК (быстрый перевод)

```
  0011 1010
+
  1110 1001
-------------
1 0010 0011(2)   (ПК) = 35(10)
```

Перенос из знакового разряда отбрасываем. Число является положительным в ПК.

перенос

3.3. **Алгоритмы умножения и деления**

Операции перемножения двоичных многоразрядных чисел производятся путем образования частичных произведений и последующего их суммирования. Частичные произведения формируются в результате умножения множимого на каждый разряд множителя, начиная с МЗР. Каждое частичное произведение смещено относительно предыдущего на один разряд. Поскольку умножение идет в двоичной системе счисления, каждое частичное произведение равно либо 0 (если в соответствующем разряде множителя стоит 0), либо является копией множимого, смещенного на соответствующее число разрядов влево (если в разряде множителя стоит 1). Поэтому умножение двоичных чисел выполняется путем сдвига и сложения. Таким образом, количество частичных произведений определяется количеством единиц в множителе, а их сдвиг – положением единиц (МЗР частичного произведения совпадает с положением соответствующей еди-

ницы в множителе). Положение точки в дробном числе определяется так же, как и при умножении десятичных чисел.

Вычислить произведение $17_{(10)} * 12_{(10)}$ в двоичной форме.

$17_{(10)}=00010001_{(2)}$; $12_{(10)}=00001100_{(2)}$

```
 00010001
*00001100
 10001      - сдвинуто на 2 разряда
+10001      - сдвинуто на 3 разряда
 11001100 = 204(10)
```

При сложении частичных произведений в общем случае возникают переносы.

Машинный вариант операции перемножения. Общий алгоритм перемножения имеет вид:

$$Z = X * Y = sign(Z) * |X| * |Y|$$

$$sign = \begin{cases} +, sign(X) = sign(Y) \\ -, sign(X) \neq sign(Y) \end{cases}$$

Операция перемножения состоит в формировании суммы частичных произведений, которые суммируются с соответствующими сдвигами друг относительно друга. Процесс суммирования начинают либо с младшего, либо со старшего частичного произведения. В компьютере формируют одно частичное произведение, к нему с соответствующим сдвигом прибавляют следующее и т.д. (т.е. не формируют все частичные произведения, а потом их складывают). В зависимости от того, с какого частичного произведения начинается суммирование (старшего или младшего), сдвиг текущей суммы осуществляется влево или вправо. Компьютерный вариант выполнения умножения целых чисел: начиная Умножение младшими разрядами вперед.

```
Y=  1 0 1 1
          1101      P1
         01101      сдвиг на 1 разряд вправо
      +   1101      P2
         100111     сумма P1 + P2
        100111      сдвиг на 1 разряд вправо
    +    0000      P3
        100111      сумма P1+P2+P3
       0100111      сдвиг на 1 разряд вправо
  +     1101      P4
       10001111   сумма P1+P2+P3+P4 (результат)=143(10)
```

Деление

Деление – операция, обратная умножению, поэтому при делении двоичных чисел, так же как и в десятичной системе счисления, операция вычитания повторяется до тех пор, пока уменьшаемое не станет меньше вычитаемого. Число этих повторений показывает, сколько раз вычитаемое укладывается в уменьшаемом.

3.4 Сложение (вычитание) чисел с плавающей запятой (ЧПЗ)

Требуется вычислить Z=X±Y, при условии, что |X|≥|Y|. Формальное выражение для выполнения этой операции можно записать следующим образом:

$$Z = X \pm Y = q_X \cdot S^{P_X} \pm q_y \cdot S^{P_y} = S^{P_X}(q_x \pm \frac{q_y}{S^{(P_x - P_y)}}) = q_Z \cdot S^{P_Z}.$$

Алгоритм выполнения операции состоит в следующем:

1. Производится выравнивание порядков. Порядок меньшего по модулю числа принимается равным порядку большего, а мантисса меньшего числа сдвигается вправо на число разрядов S, равное разности (P_x–P_y), т.е. происходит денормализация.

2. Производится сложение (вычитание) мантисс, в результате чего получается мантисса суммы (разности).

3. Порядок результата равен порядку большего числа.

4. Полученный результат нормализуется.

В общем случае сложение и вычитание q производится по правилам сложения и вычитания чисел с фиксированной точкой, т.е. с использованием прямого, обратного и дополнительного кодов.

Операции сложения и вычитания чисел с плавающей запятой, в отличие от операций с фиксированной запятой, выполняются приближенно, т.к. при выравнивании порядков происходит потеря младших разрядов одного из слагаемых (меньшего) в результате его сдвига вправо (погрешность всегда отрицательна).

Сложить два числа (ЧПЗ) $Z=X+Y$ для $S = 2$.

P_X q_X

$X \rightarrow 010$ 0.11 $2^2*0.75=3_{(10)}$

P_Y q_Y

$Y \rightarrow 001$ 0.10 $2^1*0.5=1_{(10)}$

1. 010 0.01 - выравнивание порядка Y
2. $+$ 0.11 - q_x

   1.00
3. 010 1.00 - ненормализованное значение Z

   P_Z q_Z
4. 011 0.10 - нормализованное значение Z

Z=011 $0.10 = 2^3 * 0.5 = 4_{(10)}$

Умножение ЧПЗ

Требуется вычислить $Z = X \cdot Y$. Формальное выражение для выполнения этой операции можно записать следующим образом:

$$Z=X*Y=q_xS^{Px}*q_yS^{Py}= q_xq_yS^{(Px+Py)}=q_zS^{Pz}.$$

Алгоритм выполнения операции состоит в следующем:

1. Мантиссы сомножителей перемножаются.
2. Порядки сомножителей складываются.
3. Произведение нормализуется.
4. Произведению присваивается знак, в соответствии с алгоритмом, приведенным для ЧФЗ, а именно:

$$X = X*Y = sign(X)*|X|*|Y|;$$

$$sign(Z) = \begin{cases} +, sign(X) = sign(Y) \\ -, sign(X) \neq sign(Y) \end{cases}$$

- В данном случае имеется в виду способ умножения, предполагающий отделение от сомножителей их знаковых разрядов и раздельное выполнение действий над знаками и модулями чисел.

Контрольные вопросы

1. В чем недостатки обратного кода при выполнении операции вычитания?
2. Какие существуют способы выполнения операций умножения?
3. В чем особенности дополнительного кода?

4. В чем заключается процедура модификации при выполнении операций с плавающей запятой?

5. Как представляется знак числа в компьютерах?

6. В чем заключается позиционность системы счисления?

Глава 4. ОПЕРАЦИОННЫЕ УСТРОЙСТВА

4.1 Принцип микропрограммного управления

Обработка информации в ЭВМ осуществляется в АЛУ (основная обработка), а также в контроллерах ПУ (вспомогательная, предварительная обработка). Все эти устройства – АЛУ, ПУ – по принципам организации, построения относятся к классу ОУ и предназначены для выполнения операций из списка функциональных операций по инициативе ЦП (рис. 4.1.1). Принципы построения всех этих устройств одинаковы.

Организация ОУ базируется на принципе микропрограммного управления, основные положения которого можно сформулировать в виде следующих 4 тезисов.

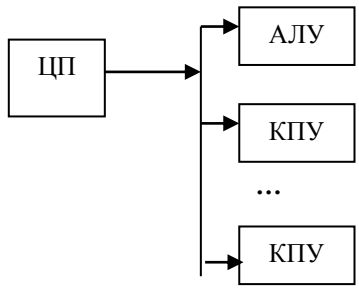


Рисунок 4.1.1

1. Любая операция $f_g \in F$ рассматривается как сложное действие и разделяется на совокупность элементарных действий, называемых микрооперациями (МО). Выполнение каждой МО осуществляется специальной комбинационной схемой (КС) за один такт машинного времени.

2. Порядок выполнения МО задается алгоритмом операции $f_g \in F$ и зависит от значений ЛУ (логических условий x). ЛУ принимают значения истина или ложь (0,1) в зависимости от значений операндов. ЛУ используются в качестве условий альтернативных переходов в алгоритмах операций.

3. Алгоритм, представленный, записанный в терминах МО и ЛУ, называется микропрограммой (МП). МП задает порядок выполнения МО и проверки ЛУ во времени.

4. Совокупность микропрограмм $МП_1$, …, $МП_g$ задает функцию ОУ.

Операция умножения $fg \in F$ для простого варианта (умножение в прямом коде): $C=A\times B$. Формат операндов A, B, разрядность k=15. Произведение C представляется в том же формате. Произведение двух n-разрядных чисел дает 2n-разрядный результат).

Алгоритм умножения:

1. Содержимым регистра A, регистра B и регистра C присвоить нулевое значение, значение счетчика обнулить.

2. Загрузить значение множимого в Rg A, множителя в Rg B, суммы частичных произведений (нулевое значение) в Rg C.

3. Содержимое регистра B сдвинуть на один разряд вправо

4. Проверка условия x1: младший разряд множителя равен 1?

5. Если «да», значение суммы частичных произведений увеличить на значение множимого, если нет, то переход на 6.

6. Сумму частичных произведений (произведение) C сдвинуть на один разряд вправо.

7. Значение счетчика уменьшить на 1.

8. Проверка условия x2 «Значение счетчика равно нулю»?

9. Если нет, то переход на п 3., если да, на выход.

Граф-схема алгоритма (ГСА) представлена на рис. 4.1.2. Сч – счетчик циклов. Операция умножения разделяется на 7 МО: сложение (реализуется комбинационным двоичным сумматором) и сдвиг (реализуется регистром сдвига). Порядок выполнения МО зависит от значений двух ЛУ : B(0), СЦ=0. ГСА умножения задает порядок выполнения МО и проверки ЛУ во времени. В зависимости от B(0) в следующем такте будет выполняться либо МО сложения C:=C+A (если B(0)=1), либо МО сдвига, если B(0)=0.

Обработка информации с помощью ОУ осуществляется выполнением операций из списка F в последовательности, которая задается алгоритмом решения задачи: ОА, выполняя программу, распределяет выполнение операций, предписанных командами программы, между различными ОУ – АЛУ, контроллерами ПУ.

Запуск (инициализация) операции $f_g \in F$ осуществляется путем подачи кода операции в ОУ из УУ. Реализация операции f_g осуществляется путем выполнения МО в порядке, заданном микропрограммой, хранимой внутри ОУ. Функционирование ОУ во времени осуществляется тактами. Реализация МПg занимает различное количество тактов n, и время выполнения операции $t_g = nT$, где T –продолжительность такта.

Принцип микропрограммного управления является основой организации ОУ. В основе управления лежит алгоритм в виде МП, находящийся в ПЗУ. При выполнении операции f_g ОУ генерирует последовательность МО, реализуемых комбинационными схемами КС.

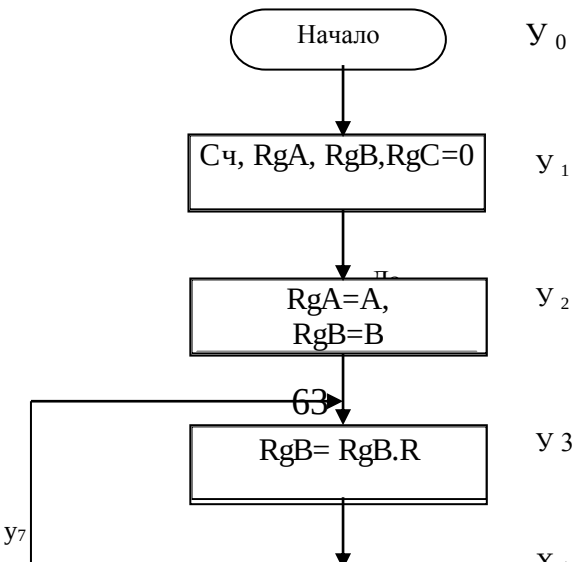


Рис. 4.12 – Граф-схема алгоритма умножения

1) операция – сложное действие, для реализации которого необходимо ОУ. МО – элементарное действие, для реализации которого достаточно КС;

2) операция выполняется за n тактов: $t_{опер}=nT$. МО выполняется за один такт (алгоритм – в структуре КС). КС управляется данными на ее входах.

Функция ОУ определяется совокупностью микропрограмм $МП_1$, …, $МП_G$, описывающих алгоритмы операций f_1, …, f_g. Для описания МП в языке используются различные средства, обеспечивающие описание слов, МО, ЛУ, а также средства, описывающие порядок их выполнения во времени.

Описание слов и массивов. Слово описывается своим именем и длиной: $C(n_1{:}n_2)$. Здесь С – имя слова, n_1, n_2 – номера старшего и младшего разрядов слова соответственно. Часть слова называется **полем** и описывается аналогично словам: A(0:7), A(0:15), B(15), A(0) и т.п.

Описание МО. Для описания МО используется оператор присваивания «:=« (или «←«). Слева от оператора указывается слово, поле, составное слово или элемент массива. Справа – двоичное выражение, которое описывает правило получения результата МО.

Описание ЛУ. Для описания ЛУ используются различного рода отношения: «<» – меньше, «>» – больше, «=0» – равно нулю, «≠0» – не равно нулю и т.п. Примеры: A<0, B≥0, C=0 и т.п. (смотри МП умножения).

Порядок выполнения МО. Порядок выполнения МО и проверки ЛУ задается в графической форме – в виде так называемой **граф-схемы алгоритма** (ГСА). ГСА строится с использованием вершин четырех типов: начальной, конечной, операторной и условной и дуг, связывающих эти вершины (рисунок 4.1.3).

Начальная вершина имеет одну выходящую дугу. Конечная вершина имеет одну входящую дугу. Операторная вершина имеет одну входящую и одну выходящую дугу. В ней записывается один или несколько операторов

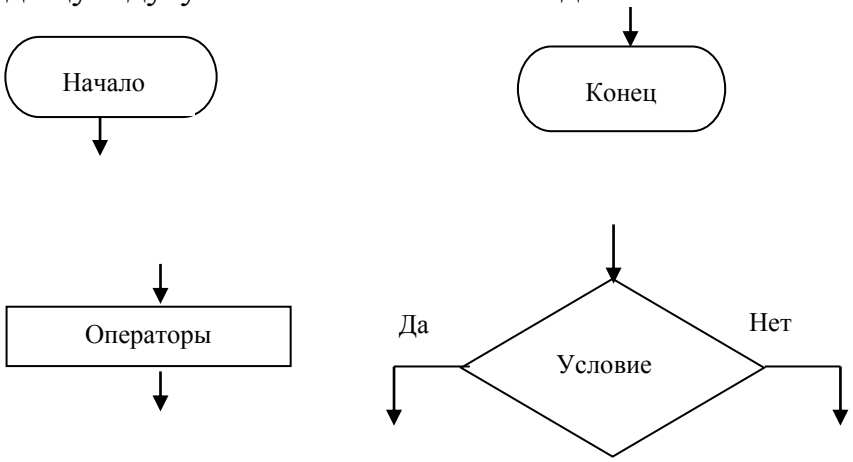


Рисунок 5.3 – Виды условных обозначений

присваивания, описывающих МО. Условная вершина имеет одну входящую и две исходящих, отмеченных символами «да» (1) и «нет» (0). Выход по дуге «да» осуществляется в случае, если ЛУ принимает истинное значение (1), и по дуге «нет» – если ложное значение (0).

ОУ состоит из двух двух частей – **операционного автомата** (ОА) и **управляющего автомата** (УА) (рис. 4.1.4).

Идея декомпозиции ОУ на ОА и УА принадлежит академику В. М. Глушкову. Конструктивность идеи заключается в следующем. Разделение ОУ (АЛУ прежде всего) на две части – пассивную исполнительную (ОА) и активную управляющую (УА) – это объективная предпосылка. В АЛУ часть узлов (сумматор и др. КС) являются исполнителями, а часть узлов (распределитель сигналов, например) являются управляющими. У этих узлов разные принципы построения, организации, причем принципы построения ОА сложнее, чем УА. Кроме того, поскольку принципы организации разные, то и их проектирование тоже раздельное.

Идея Глушкова позволила ввести очередной уровень иерархии для целой группы устройств (АЛУ, КПУ) и обобщить принципы их построения в виде абстрактного понятия – ОУ.

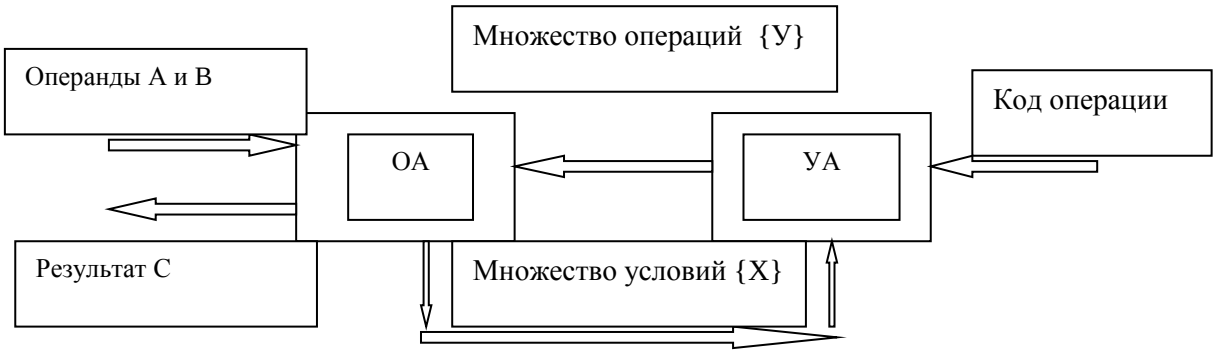


Рисунок 4.1.4 – Схема операционного устройства (ОУ)

Назначение ОА – выполнение микроопераций из списка $Y=\{y_1,\ldots,y_M\}$ под воздействием управляющих сигналов $y_m \in Y$ и формирование значений логических условий (осведомительных сигналов) $X=\{x_1, \ldots, x_L\}$.

С каждым сигналом $y_m \in Y$ в ОА отождествляется определенная МО. Например, МО сложения в МП умножения y_3: C:=C+A. Поступление сигнала y_3 в ОА приводит к выполнению этой МО и записи её результата в С.

С каждым логическим условием $x_l \in X$ в ОА отождествляется значение осведомительного сигнала, который принимает значение истина (единица) или ложь (ноль). Например, в МП умножения логическое условие (ЛУ) x_2: СЦ=0 – это булева функция, которая принимает значение «истина» (x_2=1), если СЦ=0, и ложь (x_2=0), если СЦ≠0.

Управляющий автомат (УА) предназначен для управления работой ОА. Он задает порядок выполнения МО в ОА путем выработки управляющих сигналов $y_m \in Y$ в той последовательности, которая задается микропрограммой операции $f_g \in F$ и значениями осведомительных сигналов $X=\{x_1,\ldots x_L\}$, поступающих из ОА. Например, если в ОУ (на вход «g» УА) подан код операции умножения, то управляющие сигналы вырабатываются в соответствии с алгоритмом (микропрограммой) умножения.

Таким образом, ОА является исполнительной (пассивной) частью ОУ, а УА – управляющей (активной) частью ОУ.

4.2 Операционный автомат

Функция ОА считается заданной, если определены три множества S,Y,X:

- $S=D \cup R \cup I$ – множество слов , где D – множество входных слов ОА (в МП умножения – это два слова А и В); R – множество выходных слов-результатов (примере это слово С); I – множество внутренних слов (в примере с умножением это слово СЦ);

- $Y=\{y_1,..., y_m\}$ – множество (список) микроопераций (в примере с умножением – это $y_1,..., y_7$);

- $X=\{x_1,..., x_L\}$ – множество (список) ЛУ (в примере это x_1, x_2,x_3).

Объединение производится по всем операциям ОУ:

$$S=\bigcup_g Sg, \quad Y=\bigcup_g Yg, \quad X=\bigcup_g Xg.$$

В общем случае МО описывается выражением y_m: $S_i:=\varphi_m(S_j, S_n)$. Здесь: φ_m – некоторая вычислимая функция (например, сумма), S_i, S_n – её аргументы, S_i – значение функции φ_m, вычисленное при заданных значениях аргументов $S_j = S_j^*$, $S_n = S_n^*$ (например: $S_j^* = 5$, $S_j^* = 10$, $S_j= 5+10=15$).

В общем случае ЛУ описывается выражением: $x_l:=\varphi_l (S_j)$, где φ_L – булева функция, S_j – аргумент функции φ_l, x_l – её значение при $S_j=Sj^*$.

Пример: $x_2 = 1$, если СЦ = 0, или $x_2 = 1$,если СЦ $\neq 0$.

Набор ЛУ $X=\{x_1, \ldots, x_L\}$ отображает состояние ОА.

Время не является аргументом функции ОА. Это означает, что функции ОА устанавливают только список действий Y и формируемых осведомительных сигналов X. Порядок выполнения МО и формируемых ЛУ в функциях ОА не указывается. Это означает, что функции ОА характеризуют только те средства, которые могут быть использованы для обработки информации, но не сам вычислительный процесс. Порядок выполнения МО – вычислительный процесс задают микропрограммы операций множества F: $МП_1$, …, $МП_G$, реализуемые УА и ОА. Следовательно, совокупность микропрограмм $МП_1$, …, $МП_G$ задает функцию УА. Функция УА задана, если заданы (описаны, выбраны, разработаны) МП всех операций $F=\{f_1, \ldots,f_G\}$.

Функцию ОА определяет его структура. Для реализации умножения ОА должен выполнять следующие функции:

1) $S_{умн}$ = A(0:n–1), B(0:n–1), C(0:n–1), СЦ(1:n);

2) $Y_{умн}$ – y_1:RgA, RgB RgC:=0; y_2: СЦ:=15; y_3: C:=C+|A|; y_4: C.B:=1(0.C.B); y_5: СЦ:=СЦ–1; y_6: C:=C+1; y_7:C(0):=A(0)⊕B(15);

3) $X_{умн}$ – x_1:B(15); x_2:СЦ=0; x_3:B(0); x_4:C=0.

Структура ОА для операции умножения представлена на рис. 4.2.1.

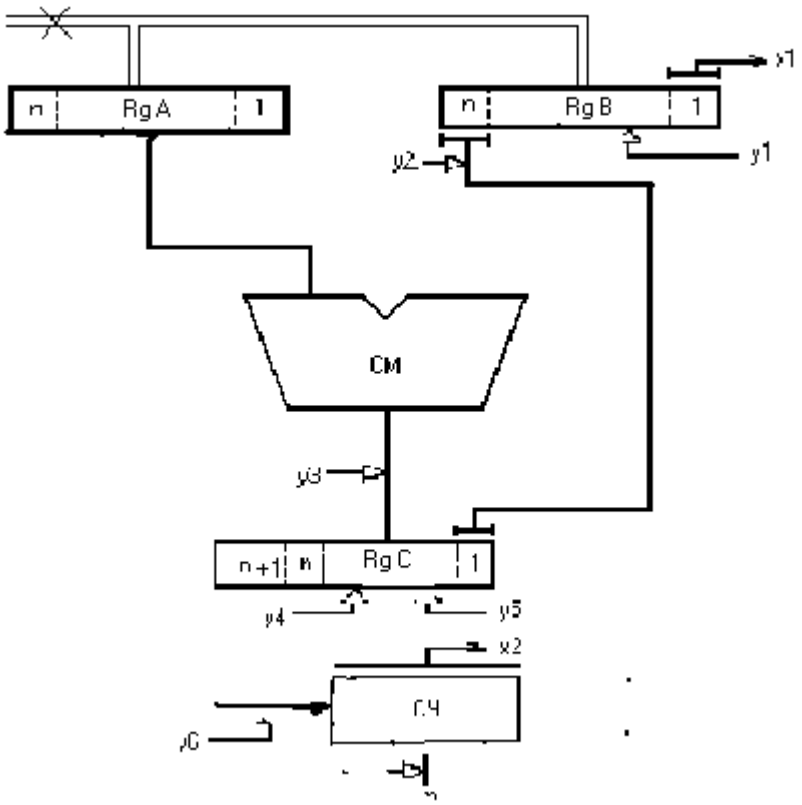


Рис. 4.2.1 – Схема операционного автомата (ОА)

Самым простым способом организации работы ОУ во времени является **синхронный** способ, при котором функционирование ОУ осуществляется тактами. Такт представляет фиксированный отрезок времени. За один такт ОУ выполняет одну или несколько совместимых МО. Такт задается как период T (T=const) следования сигналов синхронизации C, вырабатываемых генератором тактовых импульсов (ГТИ) (рис. 4.2.2):

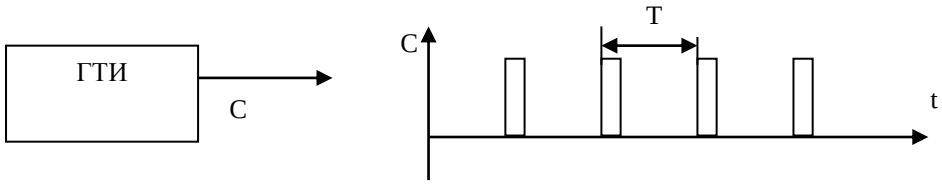


Рис. 4.2.2

Выполнение микроопераций в ОУ состоит из следующих четырех этапов: 1) этап выработки управляющих сигналов $y_m \in Y$ в УА; 2) этап выполнения МО элементами ОА; 3) этап формирования осведомительных сигналов $x_l \in X$ элементами ОА; 4) этап занесения результатов МО и осведомительных сигналов x_l в память S ОА. Далее эта последовательность этапов периодически повторяется (рис. 4.2.3).

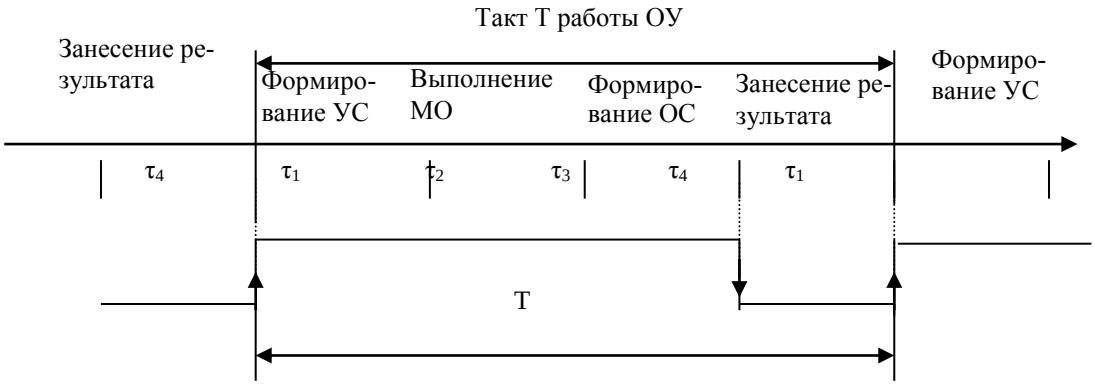


Рис. 4.2.3

Начало такта (начало последовательности этапов) обычно задается фронтом сигнала синхронизации С и привязывается к началу первого этапа. Отрицательный фронт сигнала С делит такт на две части – на два микротакта – и обычно привязывается к началу четвертого этапа (занесение результатов в память S ОА).

Рассмотрим МО сложения С:=С+В. Структура ОА, реализующего эту МО, и временная диаграмма его работы представлена на рис. 4.2.3. Как видно из этого рисунка, на выполнение МО и формирование ЛУ отводится первая часть такта. Она задается единичным значением сигнала С. На занесение результатов отводится вторая часть такта, которая задается нулевым (низким) уровнем сигнала С. Таким образом сигналы, инициирующие выполнение МО, формируются по сигналу С, а сигналы занесения результатов – по инверсному значению.

Тактовая организация работы ОУ вытекает из принципа микропрограммного управления: обработка информации сводится к выполнению МО и формированию осведомительных сигналов. Реализация МО и формирование ЛУ осуществляются комбинационными схемами, поэтому результаты с выходов КС необходимо сохранять для последующего использования.

Продолжительность такта Т при синхронной организации работы ОУ определяется суммой $\tau_1 + \tau_2 + \tau_3 + \tau_4$, которая определяется для наихудшего случая по формуле:

$$T = t_{yA} + \max(t_1, \ldots, t_M) + \max(t_{x1}, \ldots, t_{xL}) + t_S$$

Здесь: $\tau_1 = t_{yA} = \text{const}$ – время формирования управляющих сигналов Y в УА и является постоянным, не зависящим управляющих сигналов, y_m вырабатываются в данном такте; $\tau_4 = t_S = \text{const}$ – время занесения результатов. Время занесения в регистры можно считать одинаковым для всех регистров.

Время выполнения МО $y_1,\ldots, y_m$, а также время формирования осведомительных сигналов $x_1, \ldots, x_L$ в общем случае разное, поэтому продол-

жительность этапов τ_2, τ_3 определяется для наихудшего случая, т.е. как max от всех возможных значений.

Недостатки синхронного способа – потери времени при выполнении действий, продолжительность которых меньше максимальных значений. Для устранения можно использовать синхронный же способ с **переменной** длительностью такта: T=var. Реализация синхронного способа с тактом T=var усложняет структуру ОУ, в частности ГТИ, который должен вырабатывать сигналы с интервалом, который зависит от выполняемых в данном такте МО и ЛУ.

Продолжительность такта при синхронной организации работы ОУ можно уменьшить, если использовать конвейерный способ, при котором ОУ организуется в виде конвейерной цепочки, состоящей из двух блоков: УА и ОА, работающих одновременно, параллельно, а не последовательно.

При конвейерной организации этап выполнения i-й МО в ОА совмещается во времени (выполняется одновременно) с формированием управляющих сигналов для (i+1)–й МО в УА. В результате продолжительность такта работы ОУ определяется выражением: $T_k = \max(T_{УА}, T_{ОА})$, где $T_{ОА} = t_{вып} + t_{занес.рез.}$, $T_{УА} = t_{форм} + t_{занес}$, т.е. может сократиться в два раза, если $T_{УА} \approx T_{ОА}$. Конвейерная организация требует дополнительных организационных усилий и затрат оборудования.

Структурный базис ОА – это набор структурных элементов (электронных узлов), которые можно использовать для построения ОА. В этот перечень входят структурные элементы трёх типов:

- комбинационные схемы (КС) различного назначения. Используются для реализации МО и формирования осведомительных сигналов (для реализации множества X, Y);
- регистры: используются для хранения элементов (слов) информации (для реализации множества S);
- шины: используются для передачи информации (связи) между элементами структуры. ОА на уровне КС, регистров, шин.

4.3.Управляющие автоматы

Цифровым автоматом называется последовательностная схема, которая имеет набор состояний, обозначаемых обычно A_1, A_2,…, A_N. В моменты прихода тактовых импульсов автомат переходит из одного состояния в другое, определяемое как текущим состоянием, так и набором входных (осведомительных) сигналов X_1, X_2, ..., X_N. При этом формируется последовательность наборов выходных (управляющих) сигналов Y_1, Y_2, …, Y_N.

Для каждого автомата задается закон функционирования или алгоритм переходов из одного состояния в другое, под действием разных комбинаций входных сигналов с описанием комбинаций формируемых при

этом выходных сигналов. Такой алгоритм может быть задан либо в виде графа, либо в виде таблицы переходов.

Когда автомат не работает, он находится в начальном состоянии A_1. При запуске автомат сохраняет состояние A_1 в течение одного такта, за время которого формируются соответствующие значения входных сигналов X_1. По окончании первого такта автомат переключается в очередное состояние A_2 предписанное законом функционирования, и в ОА начинает выполняться следующий набор микроопераций. Момент окончания микропрограммы отмечается возвратом автомата в начальное состояние – A_1

Примером цифрового автомата служит счетчик – автомат, у которого нет вообще входных осведомительных сигналов, а есть только тактовый, на него подаются счетные импульсы. Число его состояний равно коэффициенту пересчета, граф этого автомата представляет собой кольцо с последовательным переходом от одного состояния к следующему. Более сложные цифровые автоматы могут иметь несколько возможных переходов из каждого состояния под действием разных наборов входных сигналов.

По способу формирования выходных сигналов автоматы подразделяют на автоматы Мили и автоматы Мура. Для автомата Мили определяется следующий закон функционирования

$$A(t + 1) = [A(t), X(t)]$$
$$Y(t) = [A (t), X (t)]$$

Выходные сигналы Y(t) зависят, как от состояния автомата A(t) в текущий момент времени t, так и от входных сигналов X(t).

Аналогично для автомата Мура

$$A(t+1) = [A (t), X(t)]$$
$$Y (t) = [A (t)]$$

Для него выходные сигналы Y(t) зависят только от состояния автомата A(t) в текущий момент времени t.

Реализация автоматов Мили, как правило, более проста, но в них необходимы дополнительные элементы для обеспечения синхронности формирования выходных сигналов.

Цифровые автоматы могут быть реализованы тремя разными способами:

- на элементах с жесткой логикой;
- на постоянном запоминающем устройстве (микропрограммные автоматы);
- на микропроцессорах или микроконтроллерах (программные автоматы).

Управляющие автоматы на элементах с жесткой логикой

В управляющих автоматах с жесткой логикой, для формирования кодов выходных сигналов (состояний) используется набор триггеров, на тактовые входы которых поступает тактирующий сигнал, входные сигналы x

подаются на комбинационные устройства (КЦУ), вырабатывающие сигналы управления (функции возбуждения) для триггеров. Выходные сигналы y формируются при помощи других комбинационных схем из выходных сигналов триггеров A (для автомата Мили) или из выходных сигналов триггеров и входных сигналов x (для автомата Мура, рис.).

Рис.4.3.1. Схема УА с жесткой логикой

Подобный автомат реализуется схемой, процесс синтеза которой называется структурным синтезом. Процесс структурного синтеза автомата разделяется на следующие этапы:

- выбор типа запоминающих и логических элементов;
- кодирование состояний автомата;
- синтез комбинационной схемы, формирующей сигналы возбуждения и выходные сигналы.

В.М. Глушковым разработан общий конструктивный прием, называемый каноническим методом структурного синтеза управляющего автомата. Этот метод позволяет свести задачу синтеза автомата к задаче синтеза комбинационной схемы путем построения системы булевых функций, выражающих зависимость выходных сигналов и сигналов возбуждения от входных сигналов и состояний автомата. Полученные булевы функции минимизируются (например, методом карт Карно) и используются в качестве формы для построения схемы автомата.

Схемы с жесткой логикой, как правило, позволяют обеспечить наибольшее быстродействие из всех возможных методов построения цифрового автомата, однако при возрастании сложности реализуемых алгоритмов схемы автоматов с жесткой логикой очень быстро становятся более сложными, чем схемы автоматов с микропрограммным или программным управлением. Поэтому схемы автоматов с жесткой логикой в настоящее время используют только в том случае, когда требуется максимальное быстродействие.

Одним из недостатков УА на жесткой логике является то, что любые изменения или модификации команд универсального процессора, требующие изменения микропрограмм, приведут к изменению структуры

управляющего автомата, а, следовательно, и топологии его внутренних связей.

Лучшим решением этой проблемы явилось построение УА на специализированных логических структурах с фиксированной топологией – программируемых логических матрицах (ПЛМ). ПЛМ является слоистой структурой, в каждом слое которой сосредоточены однотипные логические элементы. Топология связей построена таким образом, что на выходы каждого элемента последующего слоя подаются входные сигналы всех элементов предыдущего слоя. ПЛМ может выполняться как отдельная БИС, или формироваться внутри кристалла процессора, являясь весьма удобным элементом для создания управляющих автоматов.

При изготовлении ПЛМ образуется схема, допускающая множество вариантов обработки входных сигналов. Входные элементы позволяют иметь все входные переменные как в прямой, так и в инверсной форме. На входы любого элемента "И" поданы все входные переменные и их инверсии. Ко входам каждого элемента "ИЛИ" подключены выходы всех элементов "И". Наконец, выходные элементы позволяют получить любую из выходных функций в прямом или инверсном

Программируя ПЛМ, можно реализовать нужные системы булевых функций. Это позволяет строить управляющие автоматы весьма сложной структуры. В силу своей сложности УА, как правило, описывается большим количеством булевых функций многих переменных. Эти переменные, в свою очередь, часто бывают зависимыми. Поэтому оказывается необходимой совместная минимизация реализуемой ПЛМ системы булевых функций.

Следующим поколением устройств типа ПЛМ являются ПЛИС – программируемые логические интегральные схемы, позволяющие программно скомпоновать в одном корпусе электронную схему, эквивалентную схему, включающей от нескольких десятков до нескольких сотен ИС стандартной логики.

ПЛИС по сложности, назначению, многофункциональности делятся на две большие группы EPLD и FPGA.

EPLD – многократно программируемые – для сохранения конфигурации используется ППЗУ с ультрафиолетовым стиранием.

FPGA – многократно реконфигурируемые – для сохранения конфигурации используется статическое ОЗУ.

Управляющий автомат с программируемой логикой (УА ПЛ)

УА с ПЛ удалось реализовать с появлением компактных устройств памяти на БИС. Обобщенная структурная схема микропрограммного УА изображена на рис..

Основой управляющих автоматов с программируемой логикой является постоянное запоминающее устройство (ПЗУ). Входные (адресные) сигналы для ПЗУ формируются из номера текущего состояния, хранимого в

регистре (рис. 4.3.2.) и комбинации входных сигналов (X0, X3, подаваемые на входы A0, A3). В ячейке памяти с данным адресом в младших разрядах (D0, D2), хранится комбинация выходных сигналов (Y0,Y2), в старших, адрес перехода к новому состоянию. (Схема на рисунке изображает микропрограммную реализацию автомата Мили, выходные являются функцией входных).

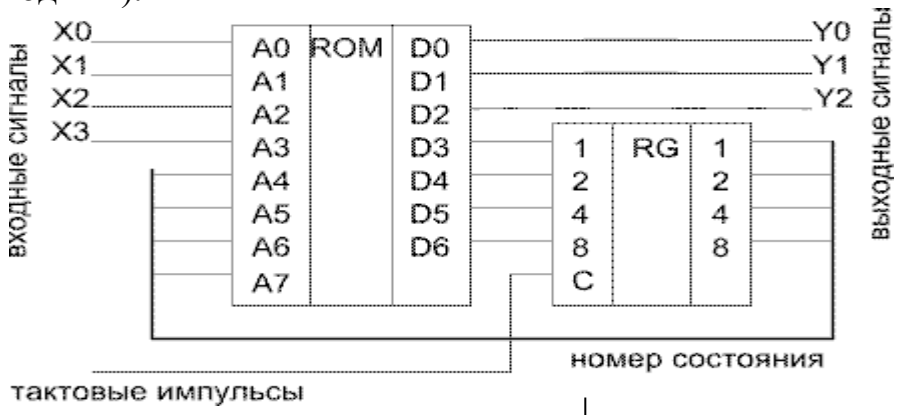


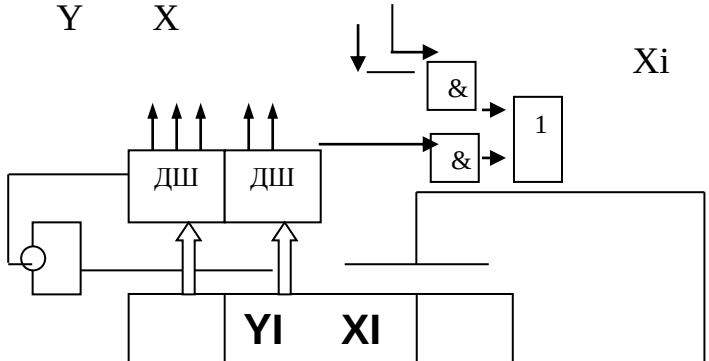
Рис.4.3.2. Микропрограммная реализация УА

Достоинством автоматов с микропрограммной логикой является простота реализации и особенно простота проектирования.

Среди недостатков следует отметить то, что, поскольку далеко не все комбинации входных сигналов и номеров состояний (адресные сигналы ПЗУ) реализуются на практике, то не все ячейки ПЗУ используются. Также и с выходными сигналами, зачастую количество их используемых комбинаций существенно меньше возможного (2^n), где n-количество линий, на которых формируются сигналы. Все это приводит к тому, что, либо приходится ставить на вход и выход автомата преобразователи кодов, а это ухудшает быстродействие схемы, либо использовать ПЗУ большого объема. Все это приводит к тому, что большую часть площади структуры современного микропроцессора занимает ПЗУ, в которой хранятся микропрограммы работы его блоков.

Автоматы с программируемой логикой являются микропроцессорными системами, в которых алгоритм функционирования реализован программным путем. Алгоритм программы для такого автомата строится следующим образом, каждая вершина графа автомата заменяется группой блоков, в которых формируются выходные сигналы и анализируются входящие, стрелки, соединяющие вершины графа автомата заменяются командами условного или безусловного перехода. (Рис.4.3.3)

Устойчивое состояние представляется в ячейках ПЗУ, вместо триггеров. В этом случае схема УА для всех операций одинакова (одна и та же).



RgМК

Рис. 4.3.3. Функциональная схема УА с ПЛ

УА с ПЛ делятся на:

- УА с принудительной адресацией;
- УА с естественной адресацией.

Недостатки УА с ПЛ с принудительной адресацией: длинные поля регистра микрокоманд и соответственно ячеек ПЗУ, потому что необходимо указывать два поля адреса. Так как адреса при безусловном переходе располагаются последовательно, для формирования следующего адреса можно применить счетчик. Такой УА автомат называется УА с естественной адресацией. Команды необходимо помечать: 0 – условная адресация, 1 – безусловная адресация.

Микропрограмма хранится в ПЗУ. МК считывается из памяти в регистр микрокоманды (PgМК). МК, в общем случае, имеет три поля – код операции, код условий, адреса следующей микрокоманды при условных и безусловных переходах.

Адрес первой МК определяет КОП, т.е. происходит вызов соответствующей микропрограммы. Адрес следующей микрокоманды может указываться в МК явным образом или формироваться естественным путем (при последовательной выборке МК). После выдачи управляющих сигналов на ОА происходит выполнение МК, после чего цикл (выборка-реализация) повторяется.

Возможны три варианта взаимного расположения циклов выборка-реализация.

Последовательный способ. В этом случае выборка следующей $МК_{i+1}$ не инициируется до момента окончания предыдущей $МК_i$. Достоинством метода является простота организации МК-цикла.

Параллельный способ (конвейер МК)

Имеет место совмещение этапов выборки $МК_{i+1}$ и реализация $МК_i$. При равенстве периодов выборки и реализации достигается фактическое сокращение МК-цикла в 2 раза.

Параллельно-последовательный способ

Используется при наличии МК условной передачи управления, когда адрес следующей МК зависит от результата предыдущей МК. Выборка $МК_{i+2}$, стоящей после $МК_{i+1}$ условного перехода, возможна только после завершения $МК_{i+1}$.

Используются два основных способа адресации - принудительная и естественная.

Принудительная адресация сводится к тому, что в каждой микрокоманде, включая операционные, указывается адрес следующей за ней микрокоманды (рис.4.3.3).

Естественная адресация характерна тем, что адрес следующей микрокоманды образуется путем увеличения адреса предыдущей микрокоманды на 1. Это позволяет исключить поле адреса из операционных микрокоманд и уменьшить разрядность регистра МК.

Для выполнения условных и безусловных переходов в микропрограмме используются управляющие микрокоманды, содержащие адрес перехода и поле признаков при обоих типах адресации.

Таким образом, операционные и управляющие микрокоманды должны различаться некоторым признаком. Признак р определяет тип МК (например,

р = 1 – операционная микрокоманда.

Для формировании адреса при естественной адресации имеется специальный счетчик адреса микрокоманд (СчА), формирующий адрес следующей микрокоманды. Алгоритм формирования адреса следующей МК зависит от ее типа, а именно:

- операционная МК - после выборки МК СчА := СчА + 1
- управляющая МК - после выборки происходит проверка условия, заложенного в МК. Если условие выполняется - СчА := АСМК, а если условие не выполняется - СчА := СчА + 1.

Существуют следующие способы кодирования микрокоманд.

1) Горизонтальное кодирование. Это простейший вариант кодирования микрокоманд, при котором каждый разряд поля кода микроопераций однозначно определяет управляющий сигнал для выполнения микрооперации. Достоинство данного способа состоит в том, что он допускает работу нескольких устройств, т.е. параллельное выполнение ряда МО, что повышает быстродействие.

Недостаток способа - при большом наборе МО (от нескольких десятков до нескольких сотен) возрастает разрядность МК и, следовательно, разрядность МК.

2) Вертикальное кодирование. При таком кодировании МК максимально сокращается поле кода операции. Но в этом случае требуется дешифратор МО, который увеличивает временные задержки и, следовательно, время выполнения МО.

3) Смешанное кодирование. Это кодирование устраняет основные недостатки, присущие горизонтальному и вертикальному кодированиям.

При таком кодировании в отдельных полях кода МО объединяют взаимоисключающие наборы МО для обеспечения параллельного выполнения МО с разных полей. Данный способ кодирования находит широкое применение в микропрограммных УА.

По сравнению с автоматами на элементах с жесткой логикой, микропрограммные автоматы обладают низким быстродействием.

В то же время в случае достаточно сложных алгоритмов работы программные автоматы оказывается наиболее простыми с точки зрения схемотехники. Особенно это ощутимо в случае использования микроконтроллеров.

Контрольные вопросы

1. В чем суть принципа академика В.М. Глушкова?
2. Почему при естественной адресации необходимо наличие счетчика?
3. В чем отличия жесткой и программируемой логик?
4. Почему при использовании RS-триггеров схемы управления триггерами усложняются?
5. Каким образом происходит выбор алгоритма операции в управляющих автоматах?
6. Какую роль играют дешифраторы при формировании управляющих воздействий?

Глава 5. ПРОЦЕССОРЫ

5.1 Понятие микропроцессор

Термин «микропроцессор» (МП) появился в связи с созданием интегральных схем (ИС), реализующих основные функции процессора: автоматическое выполнение команд программы путём поочерёдной их выборки из памяти. Адрес команды формируется в счётчике команд и выставляется на ША МП. Выбранная из ОЗУ команда принимается процессором и загружается в регистр команд – для хранения и исполнения. В соответствии с адресной частью команды операнды извлекаются из ОЗУ или внутренних регистров процессора, над ними выполняется операция (в АЛУ), а затем – засылка результата.

Степень интеграции была недостаточной, чтобы на одном кристалле (в одном корпусе) помещались все блоки процессора. Поэтому ИС выпускались в виде набора, комплекта микросхем, из которых можно было собрать процессор целиком. Таким образом, микропроцессор – это одна или несколько БИС, обеспечивающих выполнение функций процессора.

В комплект ИС, кроме схем, реализующих процессорные функции, входят и другие БИС, необходимые для построения ЭВМ и систем на основе ЭВМ: БИС ОЗУ, БИС ПЗУ, БИС управления, интерфейсные БИС и др.

ИС в микропроцессорный комплект (МПК БИС) объединяются по принципу совместимости: конструктивной, функциональной, электрической.

ИС, из которых строится процессор, образуют так называемый **базовый комплект ИС**. Если он состоит из одной БИС, то комплект называют однокристальным. В этом случае процессор обычно имеет фиксированную разрядность АЛУ, шины адреса, данных и др., а также фиксированную систему команд: МПК БИС с фиксированной системой команд.

Многокристальный МПК БИС содержит несколько типов ИС, на основе которых строится (собирается) процессор. В этом случае обеспечивается возможность строить АЛУ заданной разрядности и процессор с заданной системой команд. Систему команд формирует разработчик процессора. В базовый комплект в этом случае включаются две основные БИС: К настоящему времени степень интеграции ИС достигла уровня, при котором на одном кристалле МП удаётся разместить не только многоразрядные блоки самого процессора или нескольких процессоров (АЛУ, ЦУУ, РОН), но и дополнительные блоки скрытой буферной памяти – так называемой кэш-памяти значительного объёма (десятки, сотни килобайт).

Если на одном кристалле кроме процессора разместить другие устройства ЭВМ: ОЗУ, интерфейсные блоки для связи с внешним миром (с периферийными устройствами), то в этом случае мы имеем дело с так называемой однокристальной микро ЭВМ – **микроконтроллером**. Микроконтроллеры используются для построения встроенных в различные устройства электронных блоков управления.

Структурная организация процессоров вытекает из принципа программного управления: процессор – это устройство для реализации процесса выполнения программы. Процесс выполнения программы сводится к выполнению действий, известных как цикл выполнения команд:

1. Выборка команды из памяти.
2. Выборка операндов.
3. Выполнение операции.
4. Запись результата.
5. Переход к п.1.

Форматы команд и машинные операции

Под форматом команды понимается управляющее слово, разделенное на поля фиксированного назначения и фиксированной длины (разрядности).

Команда, состоящая из двух полей, имеет вид:

КО 1R1	k1 R2 k

Здесь: k=const, m=const, например k=8, m=16. Поле КО в виде двоичного кода задает тип операции (например, сложение) и тип данных, например сложение целых чисел (КО – сложение целых чисел), или сложение с плавающей запятой (КО – сложение с плавающей запятой), или сложение двоично-десятичных чисел (КО – сложение чисел BCD). Все они кодируются тремя разными кодами. В адресной части команды – в полях А – указываются адреса операндов.

Количество адресных полей и их длина зависят от различных факторов. Например, для двуместных операций типа сложение (вычитание, умножение и т.п.) адресная часть команды может содержать три адресных поля:

КО	A1	A2	A3

Схема выполнения трехадресной команды: [A3]:=[A1]*[A2].

Содержимое ячейки с адресом А1 – [A1] (первый операнд) и содержимое ячейки с адресом А2 – [A2] (второй операнд), извлеченные из памяти, вступают в операцию *, заданную полем КО, и результат операции засылается (записывается) в ячейку памяти с адресом А3.

В общем случае в зависимости от количества адресных полей принято различать команды: 0-адресные (безадресные), одноадресные, двухадресные, трехадресные.

Применяют команды различных форматов в зависимости от уровней памяти, т.е. в зависимости от того, адрес какого типа указан в адресной части команды – адрес типа R, в котором указывается номер регистра общего назначения (РОН), или адрес типа А – номер ячейки ОП. Отсюда различные типы форматов: RR – команды типа регистр-регистр, RM – команды типа регистр-память, MM – команды типа память-память.

Примеры двухадресных команд типов RR, RM, MM:

КО	1 R1 k	1 R2 k

k<<m. Схема выполнения:[R1]:=[R1]*[R2]

КО	1 R1 k	1 A2 m

Схема:[R1]:=[R1]*[A2]

КО	1 A1 m	1 A2 m

[A1]:=[A1]*[A2]

Множество (набор) команд, которые используются в ЭВМ, принято называть **системой машинных команд**: К={$К_1$, $К_2$, …, $К_N$}. Система команд К является основой языка ассемблер. Особенности системы машинных команд:

С системой команд неразрывно связано другое понятие – **система машинных операций** F={f_1, f_2, … , f_G}.

Система машинных операций содержит такие арифметические операции как, сложение, вычитание, умножение, деление и др., логические операции конъюнкция, дизъюнкция и др., операции сдвига кодов, чисел и т.п.

Машинная операция содержит действие, которое инициируется командой и реализуется аппаратной частью компьютера. Машинная команда представляет указание на действие. В компьютере на аппаратном уровне выполняются операции над различными типами данных: над числами (целыми, с плавающей запятой, двоично-десятичными), над символами, над логическими значениями (логические операции). Это порождает многообразие операций F.

В общем случае все машинные операции F принято разделять на классы: арифметико-логические (АЛО), операции пересылки данных; операции управления; операции ввода/вывода. В свою очередь, операции принято разделять на системные (привилегированные) и пользовательские. На уровне пользователя процессору запрещено выполнять привилегированные операции.

В современных ЭВМ в систему операций вводятся специальные операции, обеспечивающие обработку медийной информации: MMX-расширения, 3DNOW и др.

5.2 Простейший микропроцессор

В качестве простейшего, для понимания принципов работы, рассмотрим микропроцессор К580 (intel 8080), входящий в микропроцессорный комплект БИС серии КР580. Этот комплект предназначен для создания широкого класса средств вычислительной техники и обработки информации. Микропроцессор К580 выполнен по n-МОП технологии и по напряжениям логических уровней согласуется с интегральной логикой ТТЛ.

В состав базового комплекта серии КР580 входят следующие БИС:

- 8-разрядный параллельный центральный процессор КР580ИК80;
- программируемый последовательный интерфейс КР580ИК51;
- программируемый таймер КР580ВИ53;
- программируемый параллельный интерфейс КР580ВВ53;
- программируемый контроллер прямого доступа к памяти КР580ВТ57;

- программируемый контроллер прерываний КР580ВН59;

Центральный процессорный элемент является функционально законченным однокристальным параллельным 8-разрядным микропроцессором с фиксированной системой команд. Микропроцессор К580 является аналогом процессора 8080 фирмы Intel.

Краткая характеристика: тактовая частота 2 МГц; многокристальный процессор; число транзисторов около 30 тысяч; размер 5х5 мм^2; е-проводимость МОП. Данный процессор реализован по CISC–технологии. Обладает раздельными шиной адреса, шиной данных и шиной управления.

На рис. 5.2.1 приведена структурная схема БИС КР580ИК80.

Регистры данных. Для хранения участвующих в операциях данных предусмотрено семь 8-разрядных регистров. Регистр А, называемый аккумулятором, предназначен для обмена информацией с внешними устройствами (т.е. либо содержимое этого регистра может быть выдано на выход, либо со входа в него может быть принято число), при выполнении арифметических, логических операций и операций сдвига он служит источником операнда, в него помещается результат выполнения операции.

Шесть других регистров (B, C, D, E, H, L) образуют блок регистров общего назначения РОН (т.к. они могут использоваться для хранения как данных, так и адресов). Эти регистры могут использоваться как одиночные 8-разрядные регистры. В случаях, когда возникает необходимость хранить 16-разрядные двоичные числа, они объединяются в пары BC, DE, HL.

Регистры БР1, БР2, W, Z используются как буферные, программно-недоступные регистры.

16-разрядный указатель стека SP служит для адресации особого вида памяти, называемого стеком.

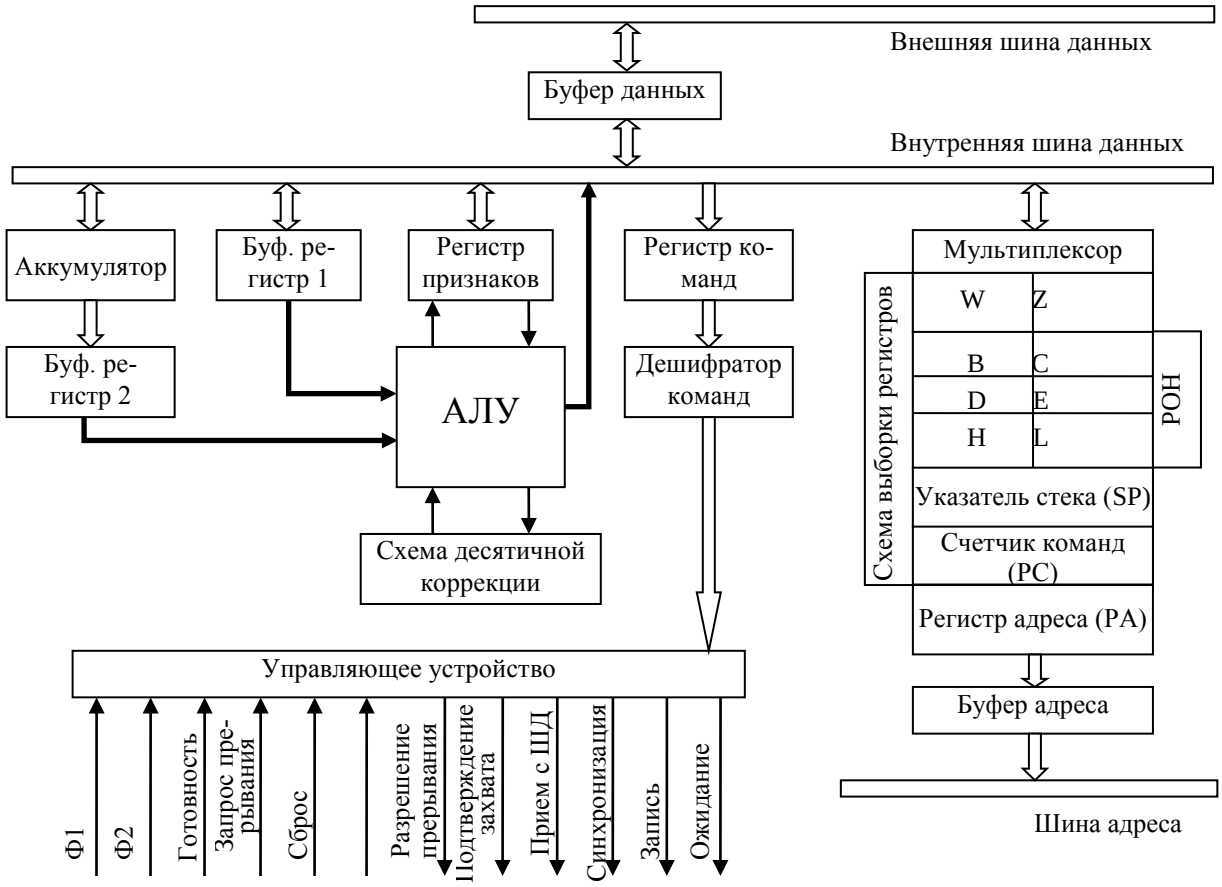


Рис. 5.2.1 – Структурная схема БИС КР580ИК80

Счетчик команд РС (16-разрядный) предназначен для хранения адреса команды. После выбора из оперативной памяти текущей команды содержимое счетчика увеличивается на единицу, и формируется адрес очередной команды (при отсутствии условных и безусловных переходов).

При обращении к памяти в качестве адреса может использоваться и содержимое любой пары регистров блока РОН.

При выдаче адреса содержимое соответствующих регистров передается в 16-разрядный регистр адреса РА, из которого далее через буферы БА адрес поступает на 16-разрядную шину адреса.

Арифметически-логическое устройство (АЛУ). АЛУ предназначено для выполнения четырех арифметических операций (сложение с передачей переноса в младший разряд и без учета этого переноса, вычитание с передачей заема в младший разряд и без него), четырех видов логических операций (конъюнкция, дизъюнкция, операция неравнозначности, сравнение), а также четырех видов циклического сдвига.

При выполнении арифметических и логических операций одним из операндов служит содержимое аккумулятора, и результат выполненной операции помещается в аккумулятор. Циклический сдвиг выполняется только над содержимым аккумулятора.

Регистр признаков (РП) содержит 5 разрядов. Регистр предназначен для хранения определенных признаков результата выполнения операций и управления. Триггеры этого регистра имеют следующее назначение:

- триггер Tc переноса – устанавливается в состояние, соответствующее переносу из старшего разряда при выполнении арифметических операций и содержимое выдвигаемого из аккумулятора разряда при выполнении операции сдвига;
- триггер Tz нуля – устанавливается в состояние логической 1, если результат операции АЛУ или операции приращения содержимого регистра равен нулю;
- триггер Ts знака – устанавливается в состояние, соответствующее значению старшего разряда операции АЛУ или операции приращения содержимого регистра;
- триггер Tp четности – устанавливается в состояние логической 1, если число единиц в разрядах результата четно;
- триггер Tv дополнительного переноса – хранит перенос из четвертого разряда, возникающий при выполнении операции.

Запуск микропроцессора. После подачи на соответствующие входы процессора питающих напряжений и тактовых импульсов последовательностей Ф1 и Ф2 подается сигнал уровня лог. 1 на вход *Сброс*. Этим сигналом сбрасываются в состояние лог. 0 счетчик команд PC, регистр команд, размещенные в управляющем устройстве триггеры разрешения прерывания, подтверждения захвата и ожидания. После окончания действия сигнала *Сброс* процессор выдает на шину адреса нулевое значение адреса.

Состояние захвата. Состояние захвата характеризуется тем, что процессор, заканчивая выполнение текущего цикла команды, переводит буферы шины данных и буферы шины адреса в состояние с высоким сопротивлением. Поэтому процессор отключается от внешних шин, предоставляя их в распоряжение некоторого внешнего устройства, и останавливает работу. После окончания сигнала *Захват* процессор выполняет следующий цикл с места, где было приостановлено выполнение программы.

Состояние прерывания. Это состояние, при котором по запросам внешних устройств выполнение текущей программы прерывается и переходит на выполнение новой программы – программы обслуживания прерывания. После окончания выполнения прерывающей программы процессор возвращается к выполнению основной программы с команды, на которой произошло прерывание (адрес команды запоминается в стеке).

Состояние останова. Выполнение программы прекращается, и происходит переход в состояние останова, процессор отключается от внешних шин и на выходе *Ожидание* устанавливается уровень лог. 1. Данное состояние может быть прервано сигналами запуска процессора либо перевода его в состояние ожидания.

На рис. 5.2.2 приведена структурная схема микропроцессорного устройства на микропроцессорном комплекте серии КР580.

Общий принцип функционирования микропроцессорного устройства заключается в следующем. Из процессора на шину адреса выдается адрес очередной команды. Считанная по этому адресу из памяти (например, из ПЗУ) команда поступает на шину данных и принимается в микропроцессор, где она исполняется. В счетчике команд микропроцессора формируется адрес следующей команды. После окончания исполнения данной команды на шину адреса поступает адрес следующей команды и т.д. В процессе исполнения команды могут потребоваться дополнительные обращения к памяти для вызова в процессор дополнительных байтов команды (в случае двух-, трехбайтовых команд) операндов или записи в память числа, выдаваемого из процессора.

Процесс выполнения команды разбивается на циклы, обозначаемые M_1, M_2, M_3, M_4, M_5. В каждом цикле производится одно обращение микропроцессора к памяти или УВВ.

В зависимости от типа команда может быть выполнена за один, два цикла и т.д. Самые длинные по времени исполнения команды выполняются в пять циклов. Каждый цикл включает в себя несколько тактов, обозначаемых T_1, T_2, T_3, T_4, T_5. Циклы могут содержать три, четыре либо пять тактов. Первые три такта во всех циклах используются для организации обмена с памятью и УВВ, такты T_4 и T_5 (если они присутствуют в цикле) – для выполнения внутренних операций в процессоре. На рис. 5.2.3 показана временная диаграмма цикла из пяти тактов.

Слово состояние (СС). Определяет коды и дальнейшее выполнение машинных циклов. СС приходит из микропроцессора из регистра управления. Для хранения слова состояния существует регистр Rg СС (на рис. 5.2.2. это блок «Фиксатор состояния»):

D_0 – обслуживание прерывания;
D_1 – запись-вывод;
D_2 – стек;
D_3 – подтверждение останова;
D_4 – вывод содержимого аккумулятора на устройство вывода;
D_5 – в данном цикле процессор принимает первый байт команды;
D_6 – ввод из устройства ввода в аккумулятор;
D_7 – чтение из памяти.

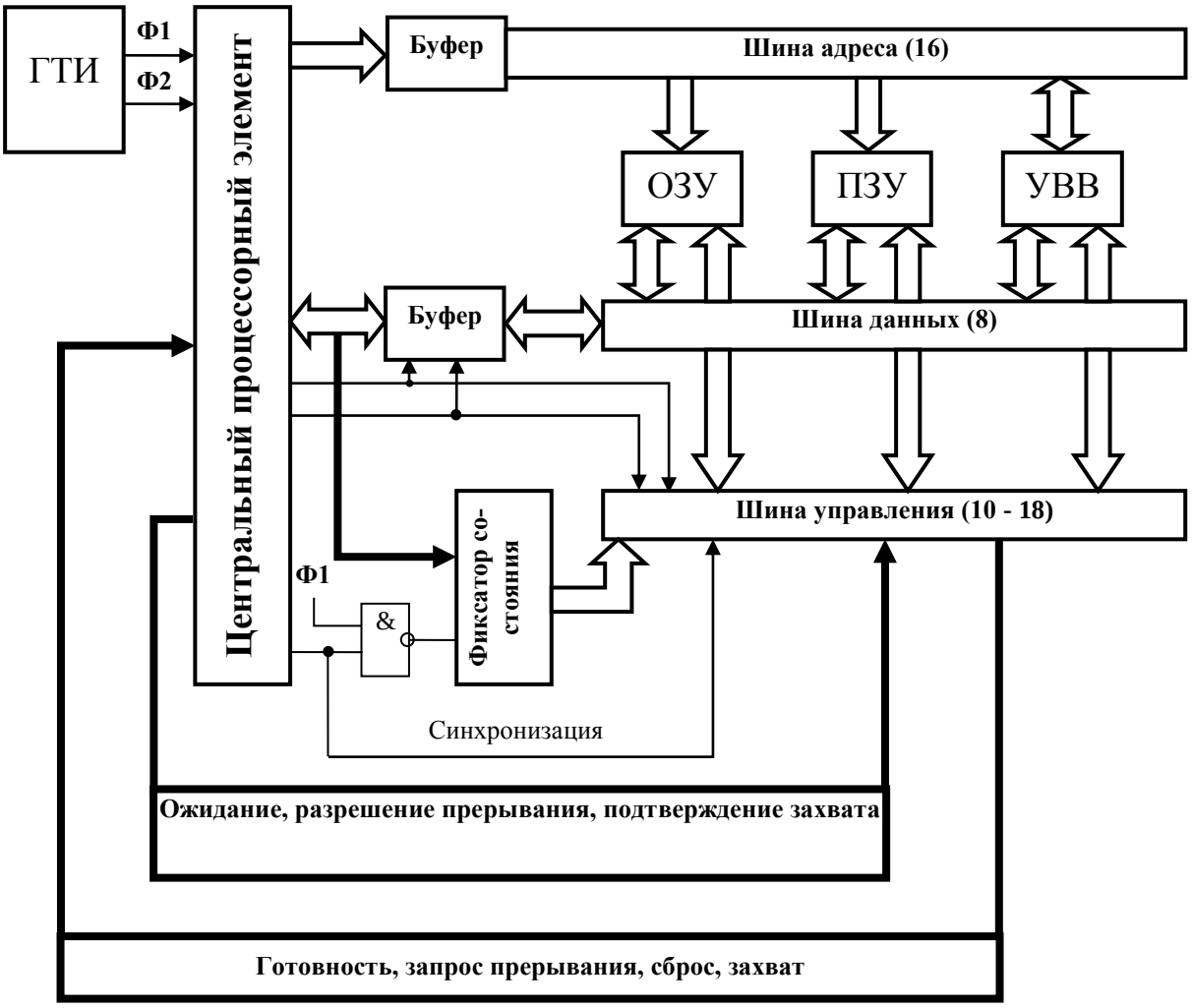


Рис. 5.2.2 – Структурная схема вычислительной системы
ГТИ – генератор тактовых импульсов

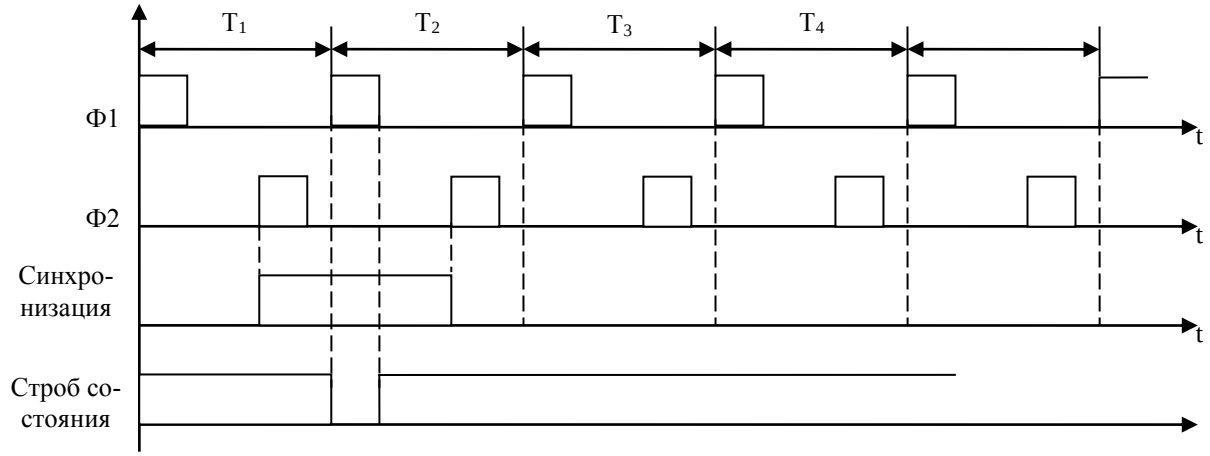


Рис. 5.2.3 – Сигналы синхронизации и строба состояния

В таблице 2 приведено соответствие сигналов состояния отдельным видам машинных циклов.

Таблица 2 – Соответствие сигналов состояния отдельным видам машинных циклов

Вид машинного цикла	Состояние микропроцессора							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Выборка первого байта команды	1	0	1	0	0	0	1	0
Чтение из памяти	1	0	0	0	0	0	1	0
Запись в память	0	0	0	0	0	0	0	0
Чтение стека	1	0	0	0	0	1	1	0
Запись в стек	0	0	0	0	0	1	0	0
Ввод из УВВ	0	1	0	0	0	0	1	0
Вывод на УВВ	0	0	0	1	0	0	0	0
Подтверждение прерывания	0	0	1	0	0	0	1	1
Подтверждение останова	1	0	0	1	1	0	1	0
Подтверждение прерывания при останове	0	0	1	1	1	0	1	1

Состав микропроцессорного набора

1. Микропроцессор КР580ИК80.
2. Шинные формирователи КР580ВА86 и КР580ВА87 (ШФ) – предназначены для соединения МКП с элементами процессорного пространства.
3. Генератор тактовых импульсов КР580ГФ24 – предназначен для формирования импульсов Ф1, Ф2, сигналов сброса, сигналов времени (отчетов времени).
4. Программируемый последовательный интерфейс КР580ВВ55 (последовательный порт).
5. Программируемый параллельный интерфейс КР580ВВ55 (параллельный порт).

Последовательный порт. Микросхема КР580ВВ55 представляет собой универсальное синхронно-асинхронное программируемое приемо-передающее устройство (УСАПП).

Процессор через шину данных осуществляет обмен байтами данных в параллельной форме (одновременно всеми разрядами). Передача данных по линии связи выполняется последовательной форме. Для сопряжения ШД с линией связи используется УСАПП. Это устройство преобразует снимаемые с ШД данные из параллельной формы в последовательную, пригодную для передачи их в линию связи; а принимаемые из линии связи данные преобразуются в параллельную форму, пригодную для выдачи на ШД.

Достоинства УСАПП: можно программировать. Задаем слово команд и слово инструкций. Программируемые данные записываются в аккумулятор, далее из него по шине данных перемещаются в регистр управления.

Параллельный порт (ППИ). С этого порта осуществляется обмен данными между процессором и различными периферийными устройствами (ПУ). Для подключения ППИ к шине данных (ШД) процессорного устройства в ППИ предусмотрен 8-разрядный канал КД. ПУ могут подключаться

к 8-разрядным каналам ППИ КА, КВ, КС. Канал КС состоит из двух 4-разрядных каналов КС1 и КС2. Каналы КА, КВ, КС снабжены регистрами. В канале КА предусмотрено два регистра, один из них используется для приема данных, поступающих из ШД процессорного устройства, и выдачи их на ПУ. Другой – для приема данных, поступающих от ПУ, и выдачи их на шину данных. В каналах КВ, КС1 и КС2 имеется по одному регистру, который обеспечивает передачу данных между процессором и ПУ в требуемом направлении. Все каналы снабжены буферными устройствами (входными и выходными формирователями с тремя состояниями), через которые осуществляется связь ППИ с внешними шинами.

Обмен между процессором и ПУ распадается на две фазы: обмен между регистром выбранного канала ППИ (каналов КА, КВ, КС) и ШД процессорного устройства и обмен между регистрами каналов ППИ и ПУ.

Организация памяти

Физическая память, к которой микропроцессор имеет доступ по шине адреса, называется оперативной памятью. На самом нижнем уровне память компьютера можно рассматривать как массив битов. ОЗУ организовано как последовательность ячеек – байтов. Каждому байту соответствует свой уникальный адрес (его номер), называемый физическим. Диапазон значений физических адресов зависит от разрядности шины адреса микропроцессора.

Механизм управления памятью – полностью аппаратный и обеспечивает:

- компактность хранения адреса в машинной команде;
- гибкость механизма адресации;
- защиту адресных пространств задач в многозадачной системе;
- поддержку виртуальной памяти.

Виды адресации операндов в памяти:

- прямая адресация – при этом способе адресом операнда является указанный в команде (в байте кода операции) адрес регистра микропроцессора;
- непосредственная – операнды (один или два) задаются непосредственно в команде вслед за байтом кода операции во втором (либо во втором и третьем) байте;
- косвенная адресация – в команде указывается пара регистров блока РОН (путем указания адреса одного из регистров этой пары), содержимое которой служит адресом, по которому в оперативной памяти находится операнд.

5.3. Микропроцессоры фирмы Intel

Процессор 8086

Общая характеристика: тактовая частота 33 МГц; приблизительное число транзисторов 29 тысяч; размер 10x10 мм2; использована e-проводимость МОП. Данный процессор реализован по CISC-технологии. Можно выделить свойства: раздельная шина адреса, шина данных и шина управления; регистры общего назначения, расширенный стек, конвейер, реализована сегментация памяти, существует возможность мультипроцессирования.

Структура – программистская модель включает систему команд, перечень (спецификацию) всех регистров, возможность описания памяти, внешних устройств (рис. 5.3.1).

Микропроцессор 8086 был выпущен в 1978 году и представляет собой 16-битовую архитектуру с внутренними регистрами, имеющими 16-битовую разрядность, а разрядность системной шины адреса – 20 бит, за счет чего может адресовать до 1 Мбайт оперативной памяти. Архитектура данного процессора расширена дополнительными регистрами. Поскольку почти каждый регистр в этой архитектуре имеет определенное назначение, 8086 по классификации частично можно отнести к машинам с накапливающим сумматором, а частично – к машинам с регистрами общего назначения и его можно назвать расширенной машиной с накапливающим сумматором. Микропроцессор 8086 стал основой, завоевавшей впоследствии весь мир серии компьютеров IBM PC, работающих под управлением операционной системы MS-DOS.

Процессор 80286

Краткая характеристика: f=33 МГц; корпус с 68 выводами. Шина адреса – 24 бита, шина данных – 16 бит. GDTR/LDTR – регистры глобальные/локальные дескриптивной таблицы. Независимое выполнение четырех задач.

Процессор 80286 принципиально отличается от 8086. Он функционирует в двух режимах – реальном и защищенном. В реальном режиме процессор 80286 является практически аналогом 8086, но имеет большее быстродействие. В реальный режим процессор переключается после аппаратного сброса или после включения питания компьютера.

В защищенный режим процессор переключается из реального режима специальной командой. Здесь он расширяет адресное пространство до 16 Мбайт. В защищенном режиме процессор 80286 имеет встроенную поддержку мультизадачных операционных систем, а также изолирует адресные пространства отдельных задач друг от друга посредством регистра задач (TR). В микропроцессоре присутствуют дескриптивные регистры, предназначенные для описания сегментных регистров.

Кольца защиты предназначены гибкой и надежной защиты ОС и программ друг от друга. Здесь используются привилегии четырех уровней:

1) кольцо 0 – ядро ОС, системные драйверы;

2) кольцо 1 – программы обслуживания аппаратуры, драйверы, программы, работающие с портами ввода/вывода компьютера;
3) кольцо 2 – системы управления базами данных, расширения ОС;
4) кольцо 3 – прикладные программы, запускаемые пользователем.

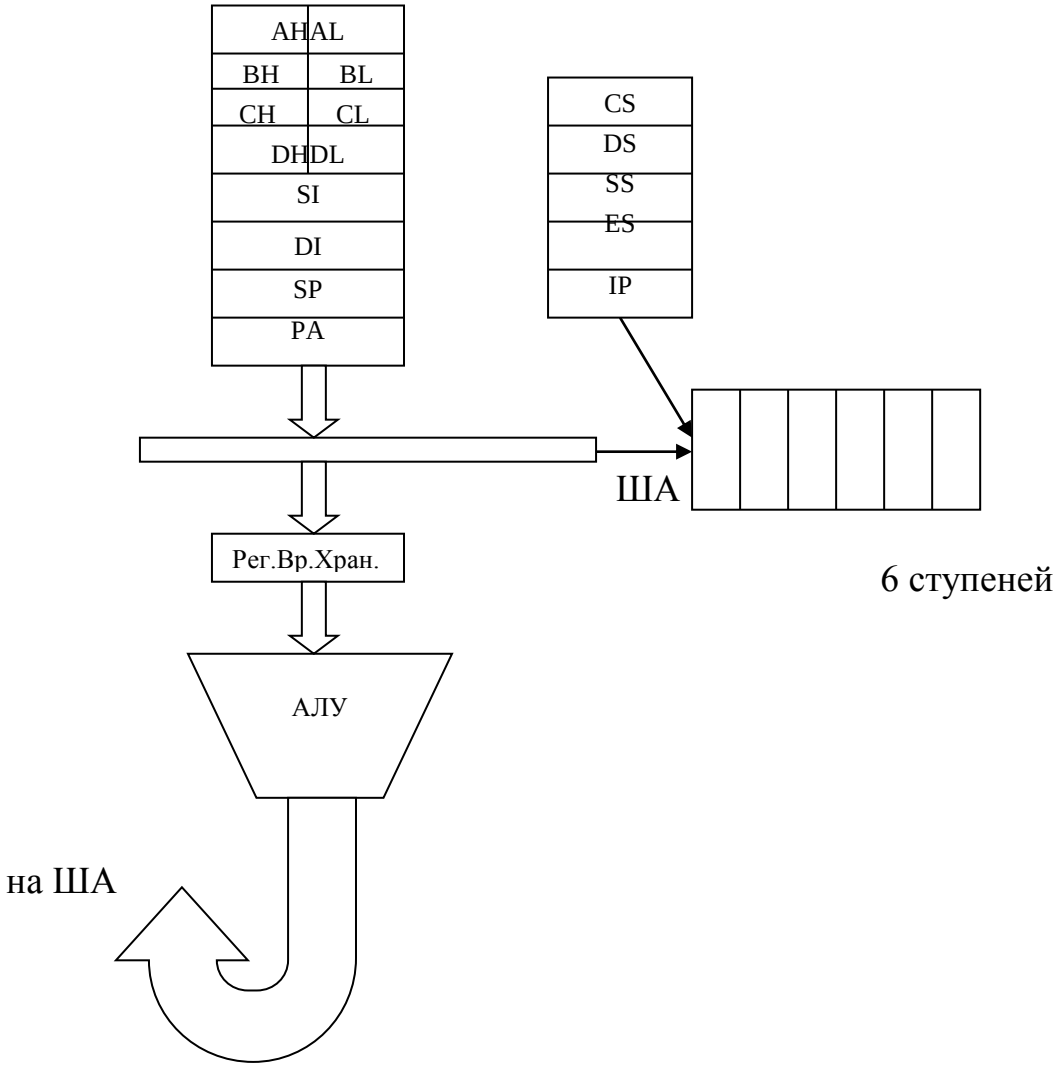


Рис. 5.3.1 – Программистская модель микропроцессора 80x86

Виртуальная память. Основная идея виртуальной памяти в том, чтобы хранить (и обновлять) содержимое большой виртуальной памяти на диске, «подкачивая» отдельные участки виртуальной памяти в реальную оперативную память по необходимости (свопинг).

Увеличены вычислительные возможности примерно в два раза, усовершенствованы средства доступа, новый таймер, имеется совместимость с нижестоящими сериями. Команды 246+16 команд управления памятью, шины адреса и шины данных разделены.

Процессор Intel 80386 (i386)

Процессор 80386 содержит следующие отличия (рис. 5.3.2)

- технология CHMOS (H – высокой плотности – 0,8 микронов);
- применение 32 разрядов в архитектуре;
- размер кристалла 10x10 мм;
- 275 тыс. транзисторов;
- частоты 16, 20, 25, 33МГц;
- 3,8 операций в секунду.

Функциональные новшества:

- многозадачность;
- встроенное управление памятью;
- виртуальная память;
- разделение на страницы;
- защищенный режим;
- большое адресное пространство от 1 Мбайта до 4 Мбайт;
- расслоение памятью;
- сохраняет совместимость по кэш-память.

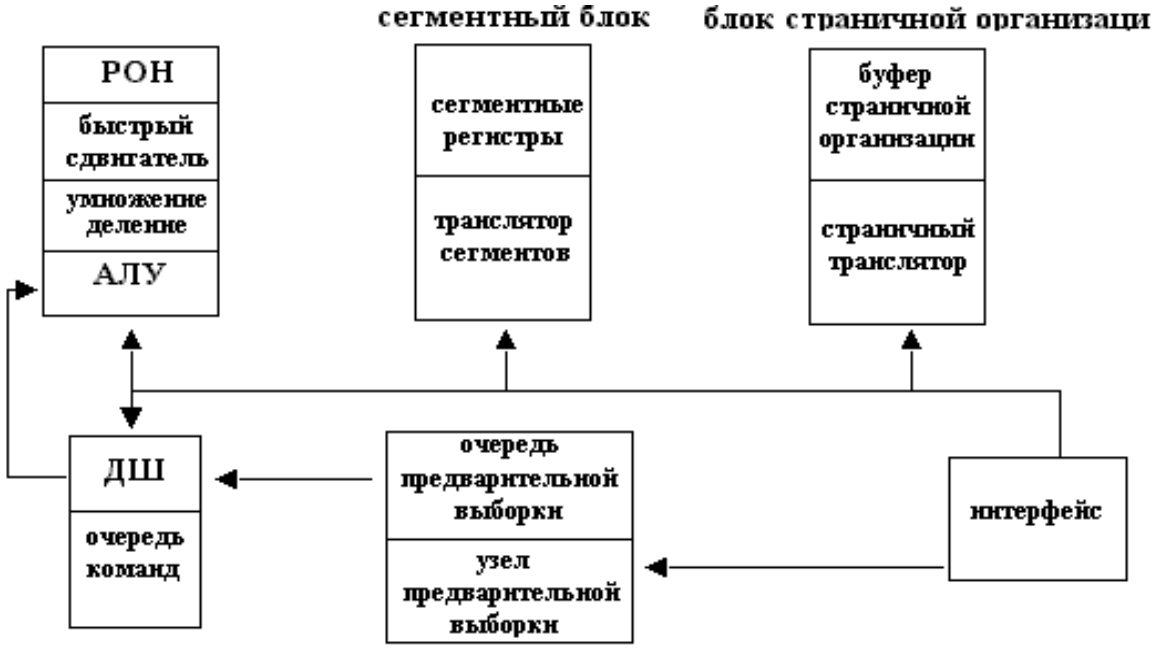


Рисунок 5.3.2 – Функциональная структура микропроцессора i386

32-битная архитектура 80386 поддерживает операции с большими числами, большими структурами данных, большими программами или большим числом программ. Физическое адресное пространство 80386 состоит из 4 Гбайт; его логическое адресное пространство состоит из 64 терабайт (тбайт).

Типы данных включают в себя 8-, 16- или 32-битные целые и порядковые, упакованные и неупакованные десятичные, указатели, строки бит, байт, слов и двойных слов. Микропроцессор 80386 имеет полную систему

команд для операций над этими типами данных, а также для управления выполнением программ. Способы адресации 80386 обеспечивают эффективный доступ к элементам стандартных структур данных: массивов, записей, массивов записей и записей, содержащих массивы (рис. 5.3.3).

В 80386 также используются регистр GDTR (32-разрядная адресация плюс смещение), регистр LDTR (16-разрядный селектор), 64-разрядный регистр для ускоренной операции сдвига, 4 разряда управления CR0 – CR3:

- CR1 – резервный;
- CR2 – линейный адрес отката страниц;
- CR3 – регистр, отвечающий за разбиение на страницы.

Разряд CR0 состоит из:

P	Резерв	1	2	3	4	5

Если P=1, то включается механизм разбиения на страницы.
Остальные пять полей характеризуют:
1 – включение защиты;
2 – присутствие сопроцессора;
3 – эмуляцию сопроцессора;
4 – переключение задач;
5 – тип расширения процессора.

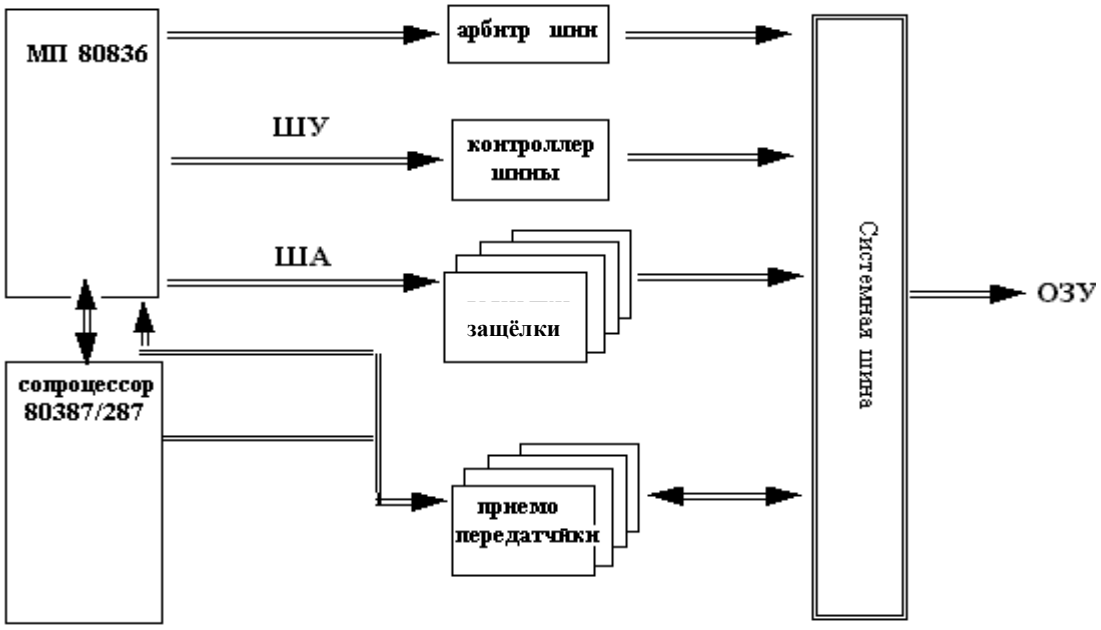


Рисунок 5.3.3 – Физическая структура микропроцессора i386

Процессор Intel 80486 (i486)

Внесены следующие обновления:

- увеличение плотности монтажа;
- увеличение количества регистров до 31:

- РОН (16 регистров),

- регистры для защищенного режима,
- DR0 – DR7 – регистры отладки,
- TR3 – TR5 – регистры проверки кэш-памяти,
- 6 – TR7 – регистры контроля буфера трансляции страниц;

▪ формат команды – 1–15 байт:

- предел операнда,
- предел адреса,
- размер адреса,
- байт повтора;

▪ шина на 64 бита;

▪ одночастотная синхронизация;

▪ передача данных через шину пакетными циклами по 32 бита за 1 такт;

▪ встроенный самоконтроль.

Качественные характеристики:

▪ выполнение полного набора арифметических и логических операций по 8/16/32 бита;

▪ прямая адресация 4-Гбитной физической памяти при раздельных ША и ШД;

▪ поддержка внутреннего устройства с плавающей точкой;

▪ целостность памяти поддерживается сегментной и страничной адресацией

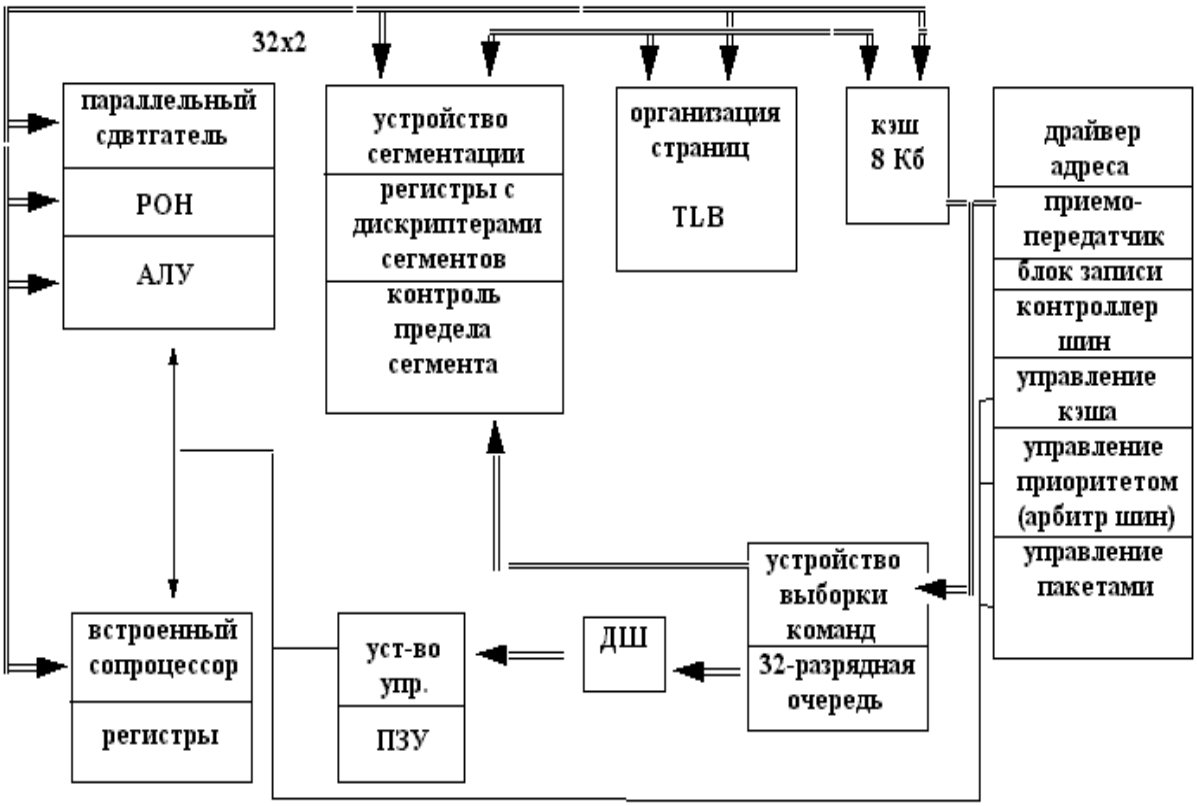
▪ наличие внутренней кэш-памяти объемом 8 Кбайт (предусмотрена внешняя кэш-память);

▪ совмещение процедур выборки декодирования, преобразование адреса, выполнение команды за 1 такт (аппаратная реализация);

▪ передача данных через системную шину двойным словом за 1 такт.

В МП i486 произошло повышение быстродействия генератора в 5 раз. Один миллион транзисторов объединенной кэш-памяти (сверхбыстрой оперативной памяти) вместе с аппаратурой для выполнения операций с плавающей запятой и управлением памяти на одной микросхеме. Процессор поддерживает программную совместимость с семейством процессоров 80x86. Часто используемые операции выполняются за один цикл, что сравнимо со скоростью выполнения RISC-команд. Новые возможности расширяют многозадачность систем. Встроенная система тестирования проверяет микросхемную логику, кэш-память и преобразование адресов памяти. Возможности отладки включают в себя установку ловушек контрольных точек в выполняемом коде и при доступе к данным. Внутренний кэш прозрачен для работающих программ, Рис. 5.3.4.

Рис. 5.3.4 – Микропроцессор Intel 486



Процессор Intel Pentium

Особенности процессора:

- 3,6 млн. транзисторов;
- суперскалярная архитектура (многопроцессорность, одновременное выполнение многих операций);
- раздельное копирование кэш-команд и данных;
- предсказание правильного адреса перехода;
- высокопроизводительное вычисление с плавающей запятой;
- расширенная до 64 бит шина данных (ШД);
- поддержка многопроцессорного режима (аппаратная);
- средства задания размеров страниц;
- средства обнаружения ошибок и функциональной избыточности;
- самотестирование;
- управление производительностью (ведется статистика, обращение к шине, правильность переходов);
- наращиваемость мощности с помощью overdrive-процессоров;
- двухконвейерная организация архитектуры;
- выполнение операций за 1 такт;
- кэш с обратной записью (128-256 Кбайт);
- блок предсказания переходов производит дополнение каждого программного цикла программы и предсказывает наиболее вероятный переход (до 80%);
- 8-тактовый конвейер;

- передача 528 Мб/сек по шине;
- два integer-процессора;
- аппаратная реализация операций умножения, деления, сдвига.

Частота таймера увеличена до 200 МГц. Напряжение питания составляет 3.3 вольта. В процессоре Pentium введено предсказание переходов (рис. 5.3.5). Для обеспечения эффективной производительности переходы должны предсказываться максимально точно, иначе после выполнения перехода будет выясняться, что считалось совсем не то, что нужно. Максимальная эффективность предсказаний для CPU Pentium составляет примерно 80%.

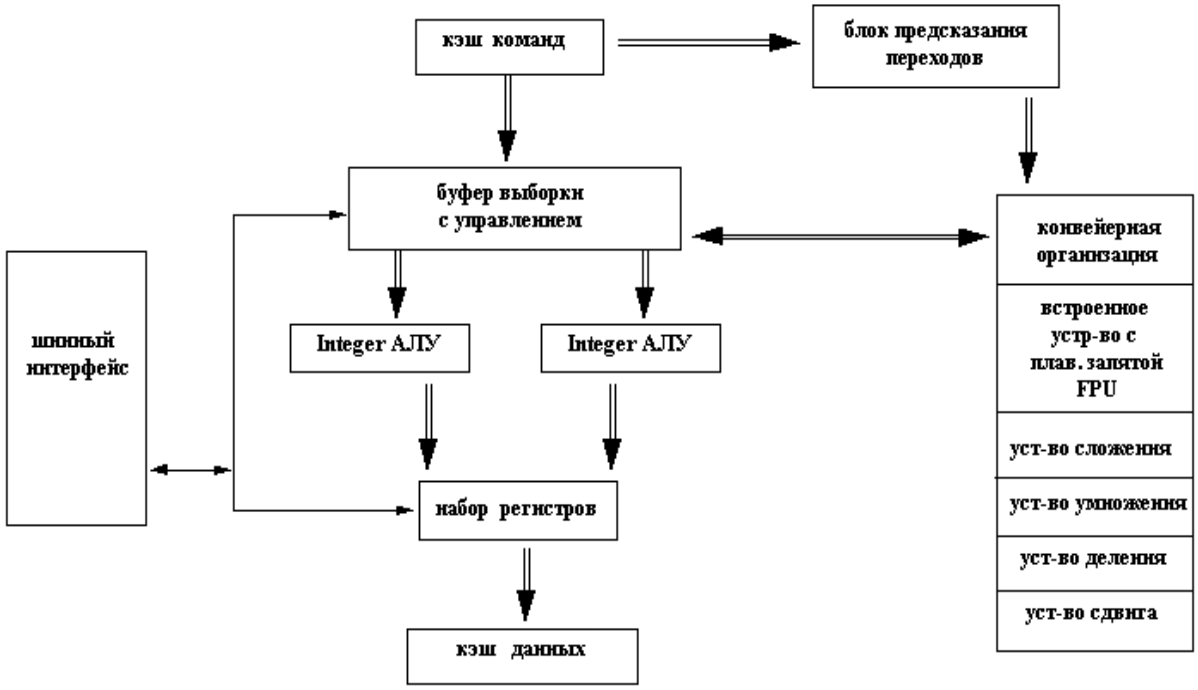


Рисунок 5.3.5 – Структурная схема микропроцессора Pentium

Процессор Pentium Pro

Процессор Pentium Pro имеет двенадцать ступеней конвейеров, и их стало три. Количество транзисторов составляет 4,5–5 миллионов. Применяются статистический и динамический методы предсказания переходов с эффективностью до 90%. Кэш второго уровня (L2), встроенный в микросхему процессора, повысил эффективность использования процессорного времени, за счет работы на более высоких частотах, чем системная плата.

Pentium Pro при равных тактовых частотах выполняет расчеты на 20–40% быстрее, чем обычный Pentium (рис. 5.3.6).

Pentium Pro поддерживает многопроцессорные конфигурации (до 4-х штук в системе).

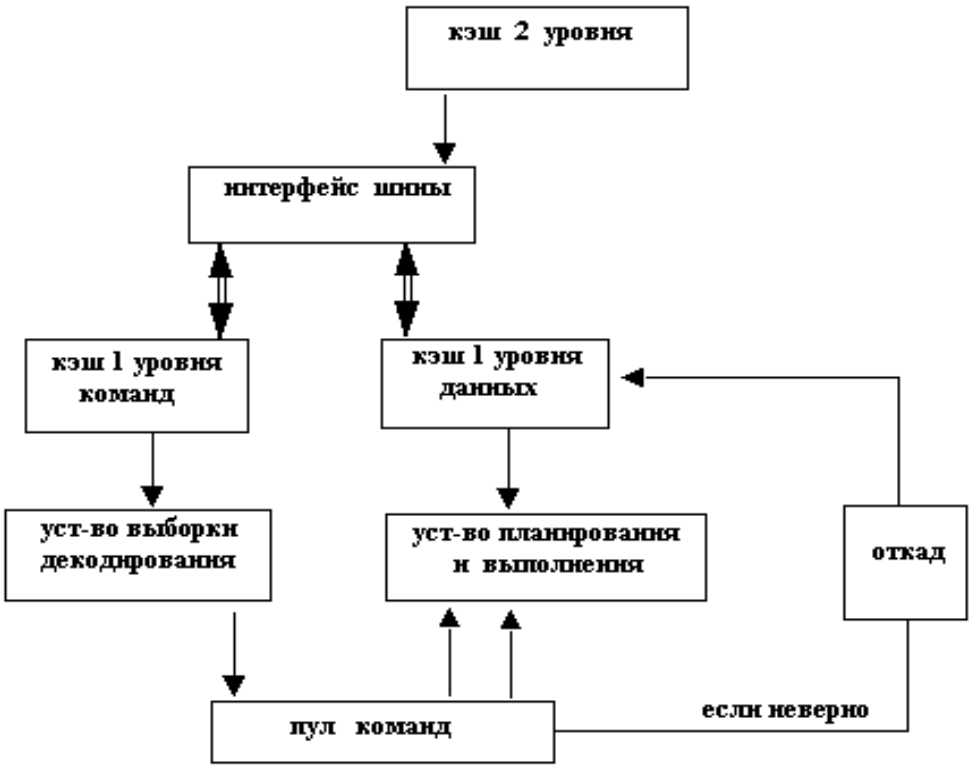


Рисунок 5.3.6 – Структурная схема микропроцессора Intel Pentium Pro

Процессор Pentium 4

Процессор Pentium 4 имеет тактовую частоту от 1,3 до 1,5 ГГц. и поддерживает весь набор команд IA-32. Расширенный набор команд SSE обрабатывает по два упакованных 64-разрядных числа с плавающей запятой или по два упакованных 64-разрядных целых числа, хранящихся в 12-разрядных регистрах. Повышение тактовой частоты достигается за счет использования длинных конвейеров с короткими ступенями (их количество 20), а также за счет усовершенствованных структуры и технологий производства схем.

Процессор содержит раздельные кэши первого уровня для команд и данных. Кэш данных имеет объем 8 Кбайт, 4-канальную множественно-ассоциативную структуру и состоит из блоков по 64 байта. Кэш команд предназначен для хранения декодированных сегментов потока команд, включающих до нескольких ветвей исходной программы. В случае повторения сегментов выполнение программы протекает быстрее, но требуется проверка, если ли переход к одной и той же ветви программы (стратегия кэширование с отслеживанием). Декодированные команды представлены как микрооперации. Каждая команда IA-32 способна определять до восьми микроопераций. В кэше с отслеживанием может находиться множество сегментов потока команд, содержащих до 12 тысяч микроопераций.

Кэш второго уровня объемом 256 Кбайт, интегрированный в микросхему, состоит из 128-байтовых блоков и обладает 8-канальной множественно-ассоциативной структурой. Соединение между кэшами L1и L2 обеспечивает пересылку данных со скоростью 48 Гбайт/с.

Процессор Motorola 68060 и семейство процессоров ColdFire

Процессор 68060 функционирует на тактовых частотах от 50 до 75 МГц и благодаря новой организации и новой технологии производства обладает в 2,5 раза большей производительностью, чем процессор 68040 с тактовой частотой 40 МГц.

Процессор 68060 имеет конвейерную суперскалярную архитектуру. Конвейер включает четыре базовые ступени, а в случае обратной записи в память используются еще две ступени. На одном такте может начаться выполнение не более трех команд. Аппаратная реализация блока обработки команд составляют три функциональных арифметических устройства: одно целочисленное и два с плавающей запятой, интегрированные кэши объемом 8 Кбайт для команд и данных. Каждый кэш имеет четырехканальную множественно-ассоциативную структуру и состоит из 16-байтовых блоков. Два буфера быстрого преобразования адресов, предназначеные для трансляции виртуальных адресов в физические, содержат по 64 записи и имеют 4-канальную множественно-ассоциативную структуру. Для увеличения скорости прохождения команд через конвейер применяется технология предсказания ветвлений.

Процессор AMD K7

Уже седьмое поколение AMD началось с оригинального процессора Athlon, и вызвано развитием суперскалярности и суперконвейерности, которая охватила и блок FPU. Особенности процессора Athlon:

- AMD Athlon содержит девять исполняемых потоков: три для адресных операций, три для целочисленных вычислений, и три для выполнения команд с плавающей точкой).
- Системная шина обеспечивает пиковую пропускную способность до 2,6 Гбайт/с.
- Расширенная технология процессора AMD Athlon построена на 21 инструкции AMD 3DNow.
- Архитектура кэш-памяти: AMD Athlon имеет кэш L1 (128KB), включает высокоскоростной 64-битный контроллер кэш-памяти второго уровня (L2), поддерживающий объем кэш-памяти второго уровня от 512Кб до 8Мб.

5.4 Организация современных микропроцессоров

Внутренняя организация процессоров постоянно совершенствуется, используя развитие технологий для повышения производительности систем. Принципы создания высокопроизводительных процессоров основа-

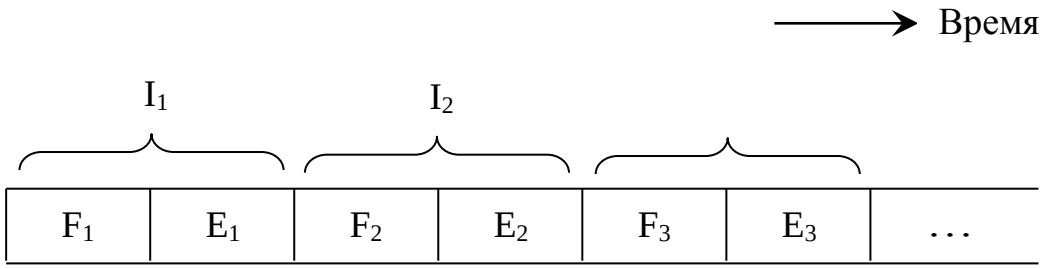
ны на обеспечении параллельной работы максимального количества различных функциональных устройств. Один подход заключается в том, что процессоры имеют конвейерную организацию, и выполнение очередной команды начинается до завершения предыдущей. При другом подходе, называемым суперскалярным, из памяти выбираются и одновременно выполняются несколько команд.

Конвейерная обработка команд заключается в том, что процессор выполняет программу, по очереди выбирая из памяти и активизируя ее команды. Шаги выборки и выполнения команды представляются как I_i, и F_i, и E_i, как показано на рис. 5.4.1, а.

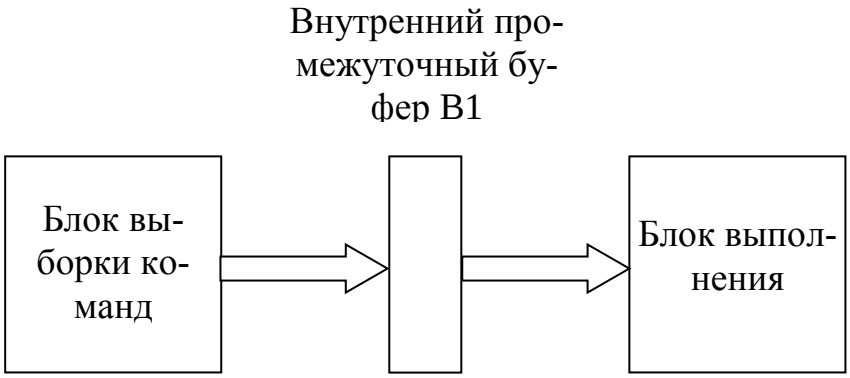
Компьютер имеет два отдельных функциональных блока – для выборки команд и для их выполнения (рисунок 5.4.1, б). Команда извлекается из памяти устройством выборки и помещается в промежуточный буфер B1, необходимый для обработки команд в то время, как блок выборки выбирает из памяти следующую. Результаты выполнения команды размещаются по указанному в ней адресу. Источник и приемник данных находятся в блоке, помеченном как «Блок выполнения».

Процессор управляется тактовым сигналом с такой частотой, при которой и шаг выборки, и шаг выполнения занимают один такт, рис. 5.4.1, в. В первом такте блок выборки извлекает из памяти команду I_1 (шаг F_1) и сохраняет ее в буфере B1. На втором такте выбирается команда I_2 (шаг F_2). Тем временем блок выполнения осуществляет операцию, указанную в команде I_1, которую он считывает из буфера B1 (шаг E_1). По окончании второго такта обработка команды I_1 завершается, к этому моменту из памяти уже считывается команда I_2. В буфере B1 сохраняется команда I_2, заменяя команду I_1, которая больше не нужна. Шаг E_2 производится блоком выполнения в течение третьего такта, пока команда I_3 извлекается из памяти блоком выборки. И так далее.

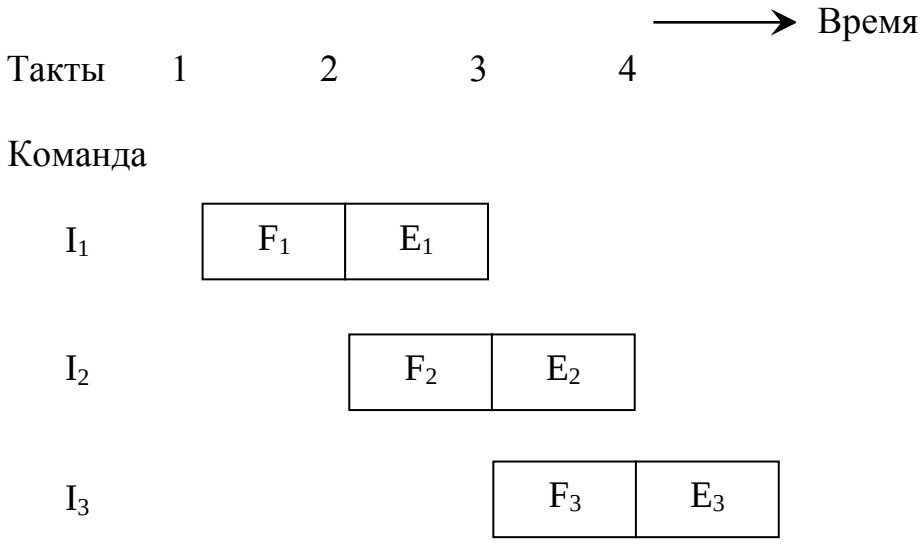
В результате и блок выборки, и блок выполнения команд все время заняты, а скорость команд вдвое больше, чем при последовательной обработке, которая схематически показана на рисунке 5.4.1, а. Блоки выборки и выполнения команд составляют двухступенчатый конвейер, на каждой ступени которого совершается один шаг обработки команды. Для хранения информации, передаваемой с одной ступени обработки команды на другую, применяется промежуточный буфер B1. В конце каждого такта в этот буфер загружается новая информация.



а



б



в

Рис. 5.4.1 – конвейерная обработка команд: последовательное выполнение (а); аппаратная организация (б); конвейерная обработка команд (в)

Периоды простоя иногда называют остановом конвейера или пузырями. Образовавшись в результате задержки на одной ступени, пузырь спускается вниз, пока не достигнет последнего блока.

Различают также структурные конфликты, возникающие при одновременном доступе двух команд к аппаратному ресурсу. Структурные конфликты наиболее вероятны при обращении к памяти. Если команды и данные расположены в одном и том же кэше, они не могут извлекаться из кэша одновременно. Поэтому пока одна команда не получит из кэша необходимую информацию, другая не сможет продолжить работу. Поэтому для устранения таких задержек процессоры оборудуются раздельными кэшами команд и данных.

Суперскалярная обработка команд

Максимальная пропускная способность конвейера равна одной команде за такт. С целью повышения производительности процессор может быть оборудован несколькими обрабатывающими устройствами, чтобы на каждом из них обрабатывалось параллельно несколько команд. При такой организации процессора на одном такте запускается на выполнение несколько команд. Процессоры такого типа называются суперскалярными.

Для данного режима требуется эффективное соединение с кэш-памятью и несколько блоков выполнения. Блок выборки команд считывает из кэша по несколько команд за один раз и сохраняет их в очереди. На каждом такте блок диспетчеризации извлекает из очереди и декодирует одну или две команды. Одна команда обрабатывает целочисленные значения, а другая – числа с плавающей запятой, и при отсутствии конфликтов обе команды диспетчеризируются на одном такте.

В суперскалярном процессоре конфликты сильнее влияют на производительность, чем в обычном конвейерном процессоре. Компилятор предотвращает многие конфликты, эффективно выбирая и переупорядочивая команды. В общем случае предельное повышение производительности достигается за счет такого переупорядочения команд компилятором, при котором максимально используются возможности всех доступных устройств.

Принимая решение о диспетчеризации необходимо, чтобы все нужные команде ресурсы были доступны. Должны быть свободны временные регистры, поскольку в них могут записываться результаты выполнения команды. Их резервирование для определенной команды является частью процесса диспетчеризации. Необходим доступ к буферу реорганизации (очередь команд, куда они записываются в том порядке, в каком они следуют в программе). Когда команда получает в свое распоряжение все ресурсы, включая блок выполнения, она диспетчеризируется для выполнения

Быстродействие микропроцессоров можно повысить:

1) путем увеличения тактовой частоты, т.е. за счет уменьшения продолжительности такта T: $t_{ком}=n*T$;

2) путем параллельной обработки (организации работы), при которой как бы уменьшается количество тактов n на выполнение команд программы – $t_{ком}=T_{КОНВ}=n*T$;

3) путем уменьшения времени обращения к памяти:

$t_{ком}=t_{ВК}+kt_{ВО}+t_{АЛУ}+t_{РЕЗ}$, т. к. время $t_{КОМ}=f(t_{ОБР})$ – существенно зависит от быстродействия памяти.

Увеличение тактовой частоты достигается за счет совершенствования технологии СБИС, применением сокращенного набора команд в RISC-процессорах, Использование простых команд одинакового формата и небольшой длины существенно упрощает структуру конвейеров, При этом уменьшаются временные задержки, что увеличивает тактовую частоту.

Повышение быстродействия в рамках CISC-процессоров решается путем использования RISC-ядра и аппаратного транслятора CISC-команд в RISC-команды.

Способы уменьшения времени обращения к памяти. Простой способ при построении ОП использовать быстродействующие ОЗУ статического типа (SRAM). Но это простое решение дорого стоит. Поэтому используется иерархическая организация памяти.

Использование буферной памяти небольшого объема и высокого быстродействия между процессором и ОП, позволяет сократить количество медленных обращений к ОП. Сокращение происходит и за счет размещения команд и данных в буфере на разных уровнях. Такая организация памяти уменьшает простои процессора в ожидании данных. Пока процессор обрабатывает блоки данных, размещенные в буфере, производится обмен блоками (программ и данных) между уровнями памяти, т. е. обеспечивается совмещение во времени обработки в процессоре с пересылкой блоков между уровнями памяти.

Эффект сокращения времени обращения к памяти будет тем больше, чем больше время обработки информации, расположенной в буфере, по сравнению с временем пересылки между ОП и буфером. Этот обусловлено свойством локальности вычислительных процессов, заключающееся в том, что процессор многократно использует одни и те же ячейки памяти при выработке некоторого результата (итерационный цикл).

Буфер организуется в виде кэш-памяти (кэш-тайник) скрытой от пользователя, чтобы не усложнять систему команд процессора. Если использовать РОН в качестве буфера, то это привело бы к усложнению форматов команд и системы команд в целом, что обязательно привело бы к усложнению структуры процессора и трансляторов. Использование быстрой кэш-памяти позволяет также сократить такт работы конвейера.

Второй способ уменьшения времени обращения заключается в **расслоении обращений к памяти,** который также использует свойство локальности адресов. При обращении к памяти процессор обычно генерирует естественную последовательность соседних адресов А, А+1, …, А+К–1 (К – длина цепочки адресов). В этом случае возможно К параллельных одновременных обращений в память по адресам, принадлежащим различным блокам памяти, состоящей из К блоков.

Распараллеливание вычислительных процессов обеспечивается: 1) конвейерной организацией суперскалярных процессоров, 2) совершенствованием системы команд, 3) мультипрограммным режимом работы процессора.

В современных суперскалярных процессоров быстродействия обеспечивается путем распараллеливания обработки данных при сохранении простых последовательных программ и языков программирования. В этом случае компиляторы и аппаратура извлекают параллелизм из последовательной программы (алгоритма) и используют эту информацию для эффективной загрузки параллельно работающих устройств (конвейеров) процессора.

При этом используется распараллеливание уровня команд, и компилятор (или процессор) преобразует полностью упорядоченное множество команд последовательной программы в частично упорядоченное множество, структурированное по данным и управлению (преобразует статическую структуру программы в динамическую). Динамическая структура состоит из последовательностей команд, представленных в порядке их исполнения. В цепочку объединяются команды, которые нужно выполнить последовательно, а в разные цепочки – команды, выполняемые параллельно. После этого каждая цепочка запускается в свой конвейер на исполнение. Поэтому в конвейерах одновременно исполняется несколько цепочек команд. Причем команды иногда исполняются не в том порядке, в котором они записаны в последовательной программе. Переупорядочение команд используется для эффективной загрузки конвейеров.

Если преобразование в динамическую структуру осуществляется процессором, то процессор продвигает «окно исполнения» по статической структуре программы, выбирая из памяти в это окно сразу несколько команд (по числу конвейеров).

Извлечение параллелизма. Главное препятствие высокопараллельного исполнения команд – зависимости между ними по управлению и по данным. Зависимости по управлению разрешаются путем предсказания переходов. Зависимости по данным бывают двух типов – истинными и вызванные ограниченными ресурсами. Истинные зависимости приходится выполнять, вызванные ограниченными ресурсами– разрешимы.

Для устранения зависимостей по управлению, причинами которых являются команды условного перехода, используется **метод предсказания.** На основе анализа исполняемого участка программы (окна исполнения) предсказывается адрес перехода, и в конвейер на исполнение посылается цепочка команд, сформированная в соответствии с предсказанным переходом. Истинное условие перехода формируется позже, поэтому предсказание перехода не может быть абсолютным. Если предсказание оказалось ошибочным, то конвейер сбрасывается и теряется время на его рестарт.

Способы предсказания переходов: 1) на основе априорной (статической) информации о вероятностях переходов (по «да» и по «нет»). По большей из них предсказывается переход. Динамическое предсказание переходов выполняется на основе информации, собираемой в процессе выполнения программы и накапливаемой в виде истории ветвлений для каждой команды условного перехода. Это может быть счетчик, который декрементируется в случае ошибки предсказания и инкрементируется в случае правильного предсказания. Предсказание осуществляется по значению числа в счетчике.

Команды в окне исполнения могут быть зависимыми по данным. Это означает, что их можно выполнять только в том порядке, в котором они заданы в последовательной программе. Существуют специальные механизмы устранения зависимостей по данным. После устранения зависимостей по данным и по управлению (предсказания переходов) команды организуются в цепочки и запускаются в конвейеры.

В случае аппаратного выделения параллелизма процессор формирует элементы параллелизма из последовательной программы в процессе ее выполнения. Из-за ограниченных возможностей аппаратуры (по сравнению с компилятором) используются простые формы параллелизма. Пример простого естественного параллелизма представляет целочисленная обработка и обработка с плавающей запятой. Использование этого параллелизма привело к появлению процессоров разнесенной структурой, состоящих из двух субпроцессоров, каждый из которых управляется собственным потоком команд. Условно их называют адресным А-процессором и исполнительным Е-процессором. А-процессор выполняет все целочисленные (прежде всего адресные) вычисления и формирует обращения к памяти по чтению и записи. Е-процессор реализует вычисления с плавающей запятой.

При чтении из памяти А-процессор отправляет данные либо в очередь АА (целочисленные данные), либо в очередь ЕА (плавающая запятая). Расщепление последовательной программы на потоки команд для А- и Е-процессоров осуществляется либо на уровне компилятора, либо в процессоре специальным блоком – расщепителем. Обмен с памятью в процессорах с расщепленной архитектурой организуется путем посылки транзакций (входных сообщений), несущих информацию об адресах ячеек и направлении обмена (ЧТ, ЗП) и самих данные, их назначение (тип данных).

Интерфейс с памятью посредством транзакций чтения и записи позволяет поставить между процессором и памятью коммутационную среду, с помощью которой переходят к организации многопроцессорных и многоядерных систем.

Функциональная организация типичного суперскалярного процессора представлена на рис. 5.4.2. Основные функциональные блоки:

1) блок предварительной выборки команд в окно исполнения (с предсказанием переходов), КЭШ-команд первого уровня и буфер команд;

2) блок декодирования команд и формирования цепочек команд для конвейеров (с фиксированной и плавающей запятой);

3) блок исполнительных устройств (конвейеров) и КЭШ данных первого уровня;

4) блоки (файлы) регистров с фиксированной и плавающей запятой;

5) блок переупорядочения и записи результатов в КЭШ или ОП;

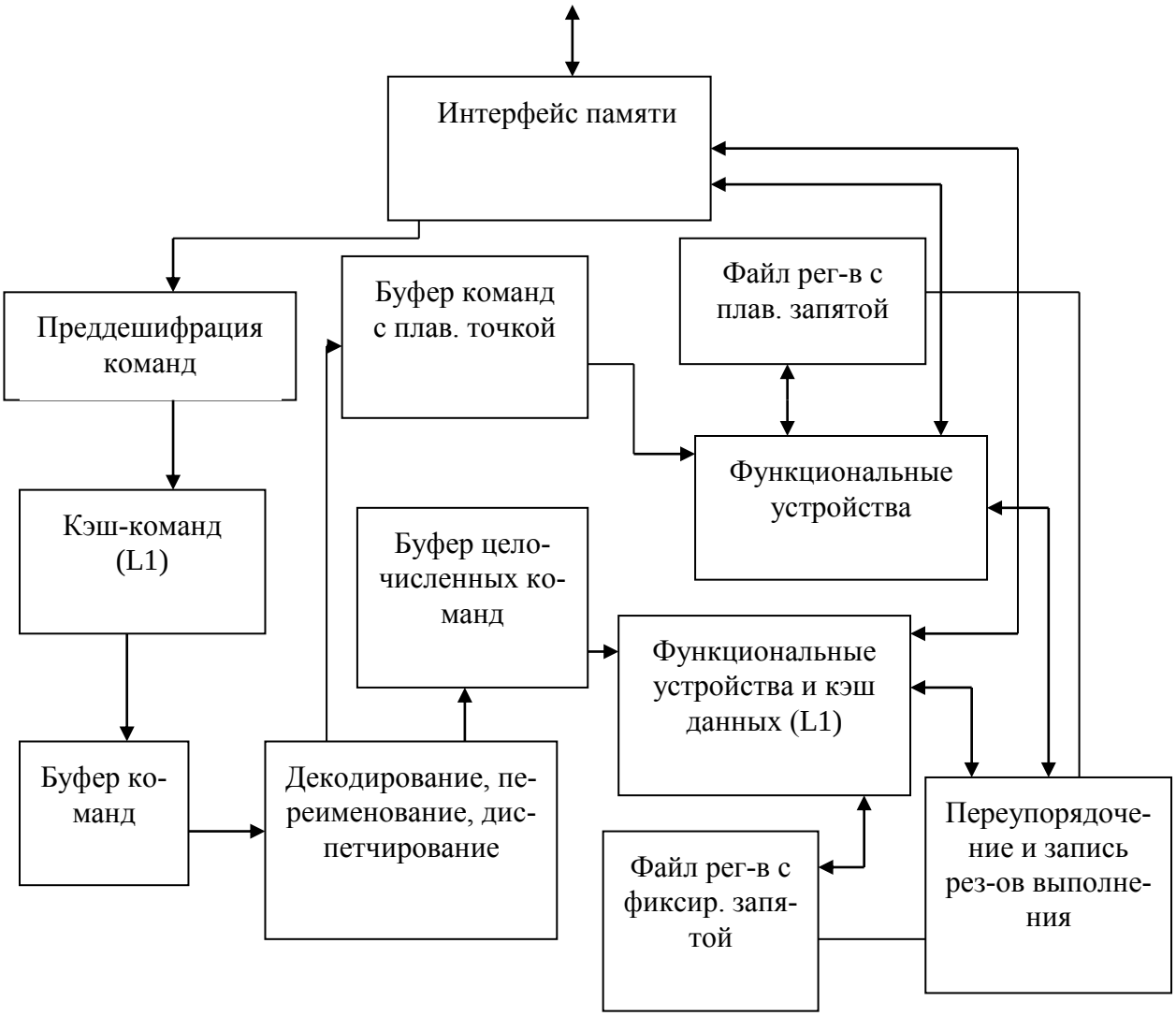


Рис. 5.4.2 – Организация суперскалярного процессора

6) интерфейс памяти (ОП и КЭШ второго уровня).

Функционирование происходит следующим образом:

1. За один такт с конвейеров процессора сходит несколько результатов (несколько команд), следовательно, на вход конвейеров также необходимо загружать несколько команд, т.е. за один такт из памяти необходимо извлекать несколько команд. Поэтому между основной (медленной) памя-

тью и блоками обработки ставятся раздельные блоки кэш-памяти для размещения данных и команд.

Для уменьшения потерь времени, связанных с отсутствием в кэш требуемых команд, в состав системы кэширования вводятся средства предсказания переходов. Цепочки команд, выбранные в соответствии с предсказанными переходами, заносятся в буфер команд.

2. Далее декодирование команд, переименование ресурсов, диспетчирование. На этой стадии выделяются истинные зависимости по данным, устраняются лишние зависимости, производится распределение (диспетчирование) команд по буферам конвейеров. Для разрешения лишних зависимостей используется процедура «переименования регистров».

3. Исполнение команд. Сформированные для каждой команды код операции, указатели (адреса) операндов, указатель результата заносятся в буферы. После этого наступает этап динамической проверки готовности значений операндов, необходимых для выполнения операции. Операцию можно исполнять, если готовы ее операнды и свободны ресурсы, на которых она может быть выполнена.

CISC-процессоры характеризуются сложностью форматов команд, их длиной и запутанным устройством управления, что препятствует повышению тактовой частоты. Поэтому переход на RISC-команды позволяет увеличивать тактовую частоту: блоки конвейеров проще, время их работы меньше, такт работы конвейера меньше, частота процессора выше. Недостаток RISC-процессоров заключается в больших объемах машинных программ.

Совершенствование системы команд. Операции (и команды) процессора делятся на скалярные и векторные.

Скалярные операции – это операции, в которых операнды и результаты являются числами, – скалярами. При обработке массивов использование скалярных команд оказывается не эффективным ввиду затрат времени на выборку команд из памяти: одна команда – один скалярный результат. Для снижения этого при обработке массивов используют векторные команды. Одна векторная команда возбуждает операцию над массивом данных, т. е. в векторной команде операнды и результат являются массивами.

Внедрение векторных команд повышает скорость обработки за счет уменьшения затрат времени на адресацию, выборку и дешифрацию команд: вместо нескольких обращений к памяти за скалярными командами – выборка одной векторной команды.

Мультизадачный режим обеспечивает рост производительности компьютера за счет эффективной загрузки процессора в целом. Переключение процессора с одной задачи на другую в современных компьютерах осуществляется часто и занимает много времени. Снижение времени переключения достигается за счет аппаратной поддержки. При этом каждой задаче предоставляется свое подмножество регистров, и переключение

контекста сводится к переадресации: имя регистра → номер регистра. В этом случае сохранения регистров в памяти не требуется, что приводит к экономии времени переключения контекста.

Архитектура VLIW. Альтернативным суперскалярной архитектуре вариантом увеличение быстродействия является процессор с длинным командным словом (VLIW). Длинная команда используется для задания совокупности параллельно выполняемых операций. Поскольку сведения о параллелизме указаны в команде, то аппаратуре процессора не нужно заниматься извлечением параллелизма и, следовательно, она организована проще: задержки в ней меньше, такт – меньше, быстродействие – выше. Подготовкой таких команд занимается компилятор или программист.

Достоинства VLIW-процессоров. Сведения о параллелизме извлекает компилятор из всей программы, поэтому он делает это эффективнее, нежели аппаратура суперскалярного процессора через узкое окно исполнения. VLIW-процессор имеет более простое устройство управления и поэтому может работать на более высокой частоте.

Недостатки VLIW-процессоров. У компилятора нет сведений о динамике процессов, поэтому он не может обеспечить переупорядочения операций, что снижает эффективную загрузку конвейеров.

Возможности совершенствование процессоров с суперскалярной архитектурой практически исчерпаны. Перспективой является **процессор с мультискалярной архитектурой.** Основная идея заключается в мультискалярной модели выполнения программы. Основное отличие от суперскалярной модели: для параллельной обработки предлагается использовать не параллельные конвейеры (структура которых зависит от специфики команд), а универсальные по функциям (одинаковые по структуре) процессоры, у каждого из которых есть собственный счетчик команд. В остальном все достаточно похоже.

В мультискалярных процессорах (МСКП) также все основано на извлечении параллелизма уровня команд из последовательной программы, представленной на языке высокого уровня. С целью извлечения параллелизма такая программа разбивается на совокупность задач. **Задача** – часть программы, выполнению которой соответствует непрерывная **динамическая** последовательность команд (например, одиночная итерация цикла). Разбиение программы на задачи осуществляется с помощью программных средств и аппаратуры процессора. Задачи статически разграничиваются аннотациями. Зависимости по управлению между ними представляются в виде графа управляющих зависимостей (ГУЗ). В нем вершинами являются задачи, а дугами задается порядок их выполнения.

Структура МСК-процессора представлена на рисунке 5.5.4. Динамика выполнения программы (то есть задач) задается как обход ГУЗ-программы (по дугам, по стрелкам ГУЗ). На каждом шаге обхода ГУЗ МСК-процессор назначает одну задачу на один из свободных процессор-

ных элементов (ПЭ) для выполнения и продолжает обход ГУЗ. Назначение обеспечивается передачей ПЭ начального значения программного счетчика (похоже на вызов программы). Значения регистров (данных) копируются в каждый ПЭ. Запущенные таким образом задачи выполняются параллельно на множестве ПЭ, что обеспечивает исполнение нескольких команд за один шаг.

МСК-процессор можно рассматривать как параллельную многопроцессорную систему с программой-планировщиком, которая назначает задачи на ПЭ. Следует отметить, что задачи, выполняемые на ПЭ, не являются независимыми. В них остаются зависимости по данным и управлению. Эти зависимости обслуживаются так: 1) каждый ПЭ поддерживает последовательную модель выполнения задачи; 2) последовательный (определяемый алгоритмом) порядок выполнения для совокупности ПЭ поддерживается с помощью организации циклической очереди ПЭ. Указатели начала и конца очереди идентифицируют ПЭ, которые выполняют самую раннюю и самую позднюю из назначенных задач.

По мере выполнения команд в ПЭ производится и потребление значений переменных программы. Эти значения связаны с регистрами и памятью. При последовательном выполнении область хранения переменных рассматривается как единый набор регистров и ячеек памяти. Единство поддерживается коммутатором. Потери производительности МСК-процессоров возможны из-за наличия в ПЭ тактов бесполезных вычислений (неверное предсказание), тактов ожидания операндов (от других ПЭ), свободных тактов (когда ПЭ стоит без задачи – свободен).

Достоинства МСК-процессоров – большая вероятность правильного предсказания переходов. При стандартном подходе в суперскалярных процессорах вероятность правильного предсказания перехода резко падает с увеличением глубины (уровня) перехода. Если вероятность предсказания одного перехода составляет обычно 0.9, то вероятность правильности предсказания на пять ветвлений вперед составляет величину порядка 0.6. Ясно, что при этом существенно падает производительность.

Мультискалярный подход сохраняет высокий уровень вероятности правильного предсказания перехода на существенно большую глубину, так как планировщик ветвлений должен предсказывать только ветви, которые определяют задачи.

МСК-процессор во многом похож на суперкомпьютер с общей памятью. Главное их отличие: суперкомпьютер требует, чтобы компилятор делил программу на задачи с прямым указанием зависимостей между задачами. Эти зависимости компилятору сообщает программист в виде специальных операторов синхронизации и межпроцессорных коммуникаций, которые используются в языках параллельного программирования. МСК-процессор не требует никакой априорной информации относительно зависимостей по управлению и данным, что обеспечивает преемственность по-

следовательного ПО, и, следовательно, не требуется использовать (более сложные) языки параллельного программирования.

В МСК-процессорах объединяются принципы низко- и высокоуровневого распараллеливания, методы анализа статической и динамической структур программы. В них соединены в целое процессы автоматического распараллеливания (распараллеливающий компилятор) и аппаратные средства, реализующие сгенерированные компилятором программы. Компилятор сам генерирует указания аппаратуре о зависимостях в виде специальных команд и отметок команд.

Следует отметить, что развитие архитектуры процессоров происходит при постоянном стремлении обеспечить преемственность ПО и аппаратуры. Задачи сохранения преемственности и повышения быстродействия процессоров противоречат друг другу.

Одна из попыток разрешения этого противоречия основана на использовании **концепции открытых систем**. Со временем растет сложность решаемых задач и, как следствие, растет сложность программных систем. Однако разнообразие типов (архитектур) процессоров ограничивает область применения сложных программных систем, т. к. для процессоров других типов их приходится разрабатывать заново, что не эффективно. Поэтому с целью расширения области применимости сложных и дорогостоящих программных систем предпринимаются попытки стандартизации (унификации) архитектуры процессоров. Однако на уровне машинных команд этого сделать так и не удалось (в силу конкуренции фирм и различного назначения процессоров). Аналогичные попытки были сделаны на уровне ассемблера, языков высокого уровня, интерфейса прикладных программ с операционными системами.

Контрольные вопросы

1. Что такое машинный такт и машинный цикл?
2. В чем суть суперскалярности микропроцессоров?
3. Почему применяется конвейерная обработка данных?
4. Почему появляется пузырь и как от него избавляются?
5. Чем объяснить наличие многих способов адресации?
6. В чем отличие RISC- и CISC-технологий обработки?

Глава 6. ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА

6.1 Общие принципы организации ввода-вывода

Простая схема подключения устройств ввода-вывода к компьютеру заключается в использовании общей шины (рис. 6.1.1). Все устройства, подключенные к шине, могут обмениваться между собой информацией.

Обычно шина состоит из трех наборов линий, предназначенных для передачи адресов, данных и управляющих сигналов. Каждому устройству ввода-вывода присваивается уникальный набор адресов. Когда процессор помещает на адресные линии конкретный адрес, распознавшее этот адрес устройство отвечает на команду, помещенную на управляющие линии. Процессор запрашивает либо операцию чтения, либо операцию записи, и запрошенные данные пересылаются по линиям данных. Организация системы ввода-вывода, при которой устройства ввода-вывода и память разделяют одно адресное пространство, называется вводом-выводом с отображением в память.

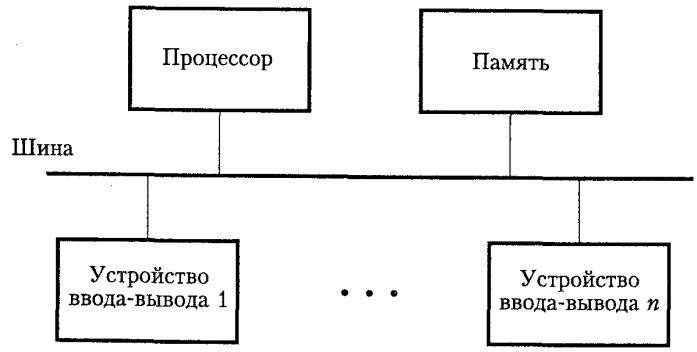


Рис. 6.1.1 – Архитектура системы с общей шиной

При использовании ввода-вывода с отображением в память обмен данными с устройствами ввода-вывода выполняется с помощью тех же машинных команд, что и при обращении к памяти.

Технология ввода-вывода с отображением в память применяется в большинстве компьютерных систем. Некоторые процессоры для выполнения операций ввода-вывода поддерживают специальные команды In и Out и отдельное 16-разрядное адресное пространство для устройств ввода-вывода. Использование части адресного пространства памяти упрощает программное обеспечение.

При наличии отдельного адресного пространства ввода-вывода используется меньшее количество адресных линий. На отношение запрошенной операции чтения или записи к системе ввода-вывода указывает передаваемый по шине специальный сигнал. Получив такой сигнал, память игнорирует запрошенную операцию, а устройства ввода-вывода анализируют младшие разряды переданного по шине адреса, чтобы узнать, кому из них направлен запрос.

Аппаратные элементы, необходимые для присоединения устройств ввода-вывода к шине, представлены на рис. 6.1.2. Когда адрес устройства появляется на адресных линиях, устройство распознает его с помощью дешифратора адреса. Данные, которыми устройство обменивается с процессором, хранятся в регистрах данных. Регистр состояния содержит информацию, относящуюся к функционированию устройства ввода-вывода.

Регистры данных и состояния соединяются шиной данных, и им присваиваются уникальные адреса. Дешифратор адреса, регистры данных и состояния, управляющие схемы, необходимые для координирования операций ввода-вывода, составляют схему сопряжение, или интерфейс, устройства.

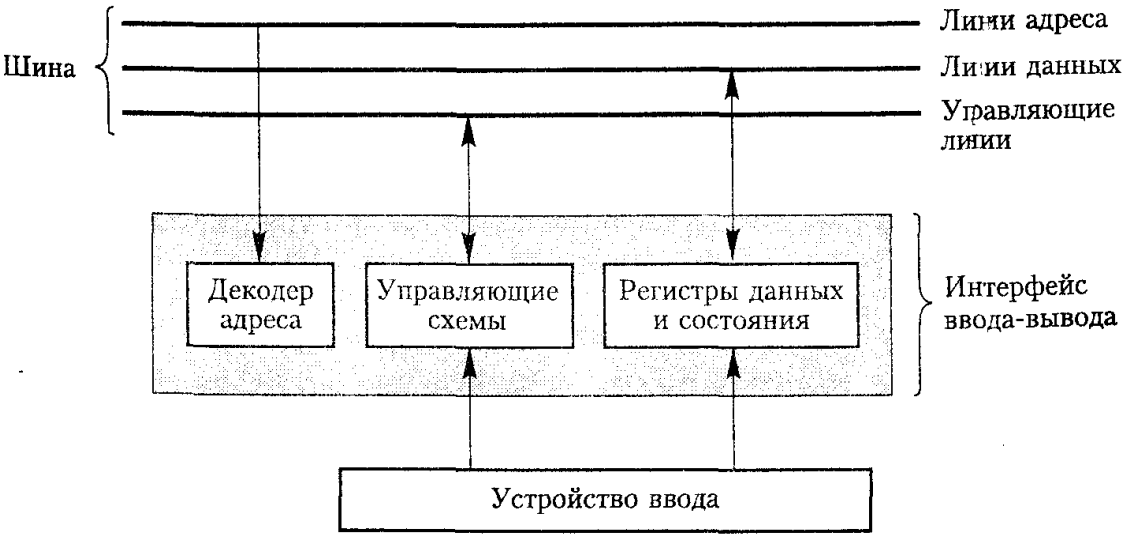


Рис. 6.1.2 – Интерфейс ввода-вывода для устройства

Скорость работы устройств ввода-вывода значительно отличается от скорости работы процессора. За время ввода символов с клавиатуры между последовательными нажатиями клавиш процессор может выполнить миллионы команд.

Команда, считывающая символ с клавиатуры, должна выполняться только тогда, когда таковой находится во входном буфере интерфейса клавиатуры. Кроме того, необходимо гарантировать, что один и тот же символ не будет прочитан из этого буфера дважды. Для входного устройства, подобного клавиатуре, в схему сопряжения в виде одного из разрядов регистра состояния включается флаг состояния SIN. Он устанавливается в 1, если символ вводится с клавиатуры, и сбрасывается в 0, если символ считывается процессором. Проверяя значения флага SIN, программное обеспечение гарантирует корректность операции чтения данных. Для этого обычно организуется программный цикл, считывающий регистр состояния и проверяющий состояние флага SIN. Обнаружив, что флаг установлен в 1, программа считывает значение из регистра входных данных. Аналогичным образом может осуществляться управление операциями вывода, но в этом случае применяется флаг состояния SOUT.

Схема, при которой процессор проверяет флаг состояния, представляет программно-управляемый ввод-вывод. Существует еще два распространенных механизма реализации ввода-вывода: прерывания и прямой доступ к памяти. Если для управления вводом-выводом используются прерывания, синхронизация достигается за счет того, что устройство ввода-

вывода само сообщает о своей готовности, отправляя через шину специальный сигнал.

Прямой доступ к памяти является характерным для высокоскоростных устройств ввода-вывода. Эта технология позволяет интерфейсным схемам устройства самостоятельно обмениваться данными с памятью, без участия процессора.

6.2 Ввод-вывод с прерываниями

Прерывание – инициируемый определенным образом процесс, временно переключающий микропроцессор на выполнение другой программы, затем с возобновлением выполнения прерванной программы. Механизм прерываний эффективнее, чем периодический опрос всех устройств ввода-вывода. Для прерываний выделяется одна управляющая линия шины, называемая линией запроса прерывания.

При возникновении прерывания микропроцессор прекращает выполнение текущей программы и переключается на процедуру обработки прерываний. При этом действия процессора похожи на вызовы подпрограмм. Перед вызовом программы обработки прерывания содержимое регистра счетчика команд временно сохраняется в памяти. Команда возврата из прерывания, расположенная в конце программы обработки прерывания, загрузит этот сохраненный адрес в регистр PC, в результате чего выполнение будет продолжено с точки прерывания.

Перед вызовом программы обработки прерывания необходимо сохранить всю информацию, которая может быть изменена в ходе ее выполнения, и перед выходом из программы обработки прерывания эта информация должна быть восстановлена. После этого исходная программа продолжает свое выполнение. К числу сохраняемой и восстанавливаемой информации относятся значения флагов условий и содержимое всех тех регистров, которые используются и прерванной программой, и программой обработки прерывания.

Задача сохранения и восстановления информации может автоматически выполняться процессором или же командами программы. Процесс сохранения и восстановления регистров включает обмен данными с памятью, увеличивающий время задержки между получением запроса прерывания и вызовом программы его обработки (задержка обработки прерывания). В современных процессорах сохраняют только минимальное количество информации (одержимое счетчика команд и регистра состояния процессора) необходимое для обеспечения целостности программ. Дополнительная информация сохраняется программным путем в начале работы программы обработки прерывания и восстанавливается перед ее завершением.

Последовательность событий, происходящих в ходе обработки запроса прерывания:

1. Устройство генерирует запрос прерывания.
2. Процессор прерывает текущую выполняемую программу.
3. Последующие прерывания запрещаются, для чего изменяются управляющие биты в регистре PS (за исключением схем, в которых линия запроса прерывания управляется фронтом сигнала).
4. Устройство информируется о том, что его запрос распознан, и в ответ сбрасывает сигнал запроса на прерывание.
5. Запрошенное прерыванием действие выполняется программой обработки прерывания.
6. Прерывания разрешаются, выполнение программы возобновляется.

Интерфейс устройства ввода-вывода генерирует запрос прерывания, когда устройство готово к операции пересылки данных (флаг SIN устанавливается в значение 1). Запросы прерываний должны генерироваться только теми устройствами ввода-вывода, которые используются данной программой. Другие устройства не должны генерировать запросов прерываний, даже если они готовы выполнить операции ввода-вывода. В интерфейсных схемах устройств ввода-вывода заложен механизм, определяющий возможность генерации запросы прерываний. Для этого используется один разряд, разрешающий или запрещающий прерывания..

Существует два независимых механизма управления запросами прерывания. Со стороны устройства решение о том, можно ли ему генерировать запросы прерывания, зависит от состояния разряда разрешения на прерывание. Со стороны процессора решение о том, будет ли принят запрос на прерывание, зависит либо от разряда разрешения прерываний в регистре PS, либо от системы приоритетов.

В современных компьютерных системах управление прерываниями для пользовательских программ выполняет операционная система компьютера, координирующая вышеизложенные действия.

Устройства ввода-вывода запрашивают прерывания путем активизации линии шины, называемой линией запроса прерывания. В большинстве компьютеров прерывания могут запрашиваться несколькими устройствами ввода-вывода. В этом случае для обслуживания прерываний также используется одна линия (рис. 6.2.1). Каждое из устройств подсоединяется к этой линии с помощью ключа, соединенного с «землей». Для того чтобы запросить прерывание, устройство замыкает этот ключ. Таким образом, если ни один из сигналов запроса прерывания от $INTR_1$ до INTRn не активен, то есть если все ключи открыты, напряжение на линии запроса прерывания равно V_{dd}. Это неактивное состояние линии. Когда устройство запрашивает прерывание, замыкая свой ключ, напряжение на линии падает до 0, в результате чего процессор получает сигнал запроса прерывания

INTR, равный 1. Поскольку замыкание одного или нескольких ключей приводит к падению напряжения на линии до 0, значение INTR является логической суммой (ИЛИ) запросов отдельных устройств:
$INTR = INTR_1 + \ldots + INTR_n$.

Для обозначения сигнала запроса прерывания часто используют форму дополнения $\overline{INTR}$, поскольку сигнал активен, когда напряжение на линии равно 0.

В электронной реализации схемы, показанной на рис. 6.2.1, для управления линией $\overline{INTR}$ используются специальные вентили, называемые вентилями с открытым коллектором (для двухполюсных схем) или вентилями с открытым стоком (для схем МОП). Такие вентили эквивалентны ключу, соединяющему линию с «землей»: когда ключ открыт, значением на выходе вентиля является 0, а когда закрыт – 1. Уровень напряжения, а также логическое состояние выхода вентиля в соответствии с приведенным выше равенством зависят от сигналов, поданных на все соединенные с шиной вентили. Резистор R называется повышающим резистором, поскольку при открытых ключах он позволяет поднять напряжение на линии до уровня, соответствующего состоянию высокого напряжения.

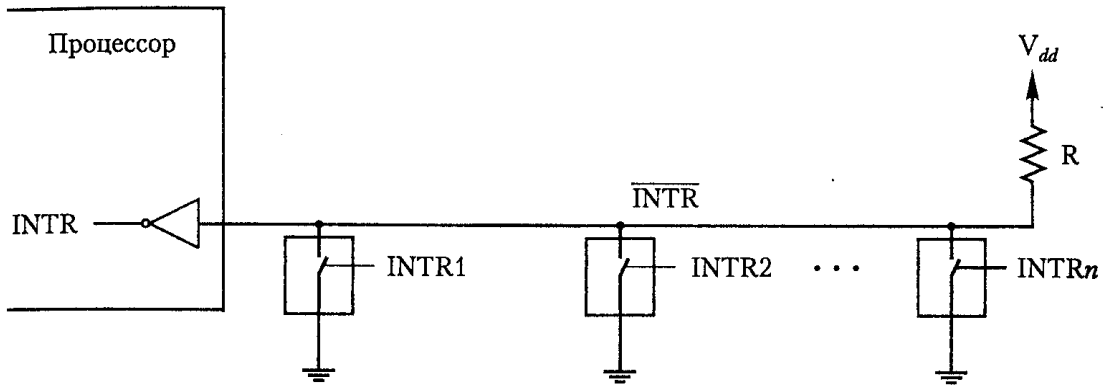


Рис 6.2.1 – Архитектура шины с открытым стоком, используемая при реализации типичной линии запроса прерывания

С процессором соединено несколько устройств, инициирующих прерывания, Эти устройства функционально независимы, они генерируют прерывания без какой-либо определенной последовательности.

При получении процессором запроса прерывания по общей линии (рис. 6.2.2), требуется информация для определения устройства, активизирующего линию. Если два устройства активизировали линию одновременно, нужно выбрать одно из них для обслуживания. Когда первое устройство будет обслужено, наступит очередь второго.

Информация, необходимая для идентификации устройства, запросившего данное прерывание, имеется в его регистре состояния. При запро-

се прерывания устройство устанавливает в 1 один из разрядов в регистре состояния (разряд IRQ). Простейший способ определения устройства, запросившего прерывание, заключается в опросе всех присоединенных к шине устройств ввода-вывода. Сначала обслуживается устройство, у которого разряд IRQ был установлен первым. Для обработки запроса вызывается соответствующая программа. Главным недостатком этой схемы является время, уходящее на проверку IRQ-разрядов тех устройств, которые не запрашивали прерывание.

Для сокращения времени при опросе устройств реализуют схему с векторными прерываниями. В этом варианте процессор выполняет программу обработки прерывания после поступления запроса, а устройство, запросившее прерывание, идентифицируется с помощью специального кода, пересылаемого процессору по шине. Этот код определяет начальный адрес программы обработки прерывания, предназначенной для данного устройства.

Программа обработки прерываний от конкретного устройства располагается по одному и тому же адресу с соответствующей подпрограммой. Во многих компьютерах такой переход автоматически выполняется механизмом обработки прерываний. Место в памяти, на которое указывает вызвавшее прерывание устройство, используется для хранения начального адреса программы обработки прерывания. Процессор считывает этот адрес, называемый вектором прерывания, и загружает его в регистр PC. Кроме адреса вектор прерывания может содержать новое значение регистра состояния процессора.

В большинстве компьютеров устройства ввода-вывода направляют код вектора прерывания по шине данных с использованием управляющих сигналов шины, и это является гарантией того, что устройства не будут мешать друг другу. В момент запроса прерывания запрещены, и возможны еще большие задержки. Устройство, вызывающее прерывания, должно дождаться готовности процессора и лишь после этого поместить данные на шину. При готовности процессора получить код вектора прерывания, он активизирует линию подтверждения прерывания INTA, а устройство ввода-вывода отвечает на это отправкой кода вектора прерывания и выдачей сигнала INTR.

При запрещении всех прерываний на время выполнения программы обработки прерывания запрос одного устройства не сможет вызвать более одного прерывания. Если имеется несколько устройств, то их прерывания обрабатываются по очереди, а начатая программа обработки прерывания выполняется до конца.

Большая задержка с ответом на запрос прерывания приводит к неверному функционированию устройств. Пример: таймер реального времени, которое направляет процессору запросы прерываний через фиксированные промежутки времени. Процессор выполняет короткую программу обработ-

ки прерывания, увеличивающую хранящийся в памяти набор значений счетчиков, содержащих количество секунд, минут и т.д. Правильное функционирование таймера возможно при условии, что время задержки перед обработкой запроса прерывания значительно меньше временного интервала между запросами. Это требование будет выполняться лишь в том случае, если запрос прерывания от таймера будет приниматься во время выполнения программы обработки прерывания, вызванного другим устройством.

Для правильной организации ввода-вывода используется система приоритетов устройств: запросы прерываний принимаются от устройств с более высоким приоритетом.

Приоритет процессора обычно задается несколькими разрядами в слове, определяющем его состояние и может быть изменен программно в регистре PS. Эти привилегированные команды выполняются лишь при условии, что процессор работает в режиме супервизора. Перед началом реализации прикладных программ процессор переключается в пользовательский режим. Таким образом, пользовательская программа не может случайно или намеренно изменить приоритет процессора и нарушить работу системы.

Многоуровневая схема приоритетов реализуется отдельными линиями запроса и подтверждения прерываний для каждого устройства, рис. 6.2.2. Каждой линии запроса прерывания присваивается свой уровень приоритета, и запросы прерываний направляются на арбитражную схему процессора. Запрос принимается только в том случае, если у него более высокий уровень приоритета, чем у процессора в данный момент.

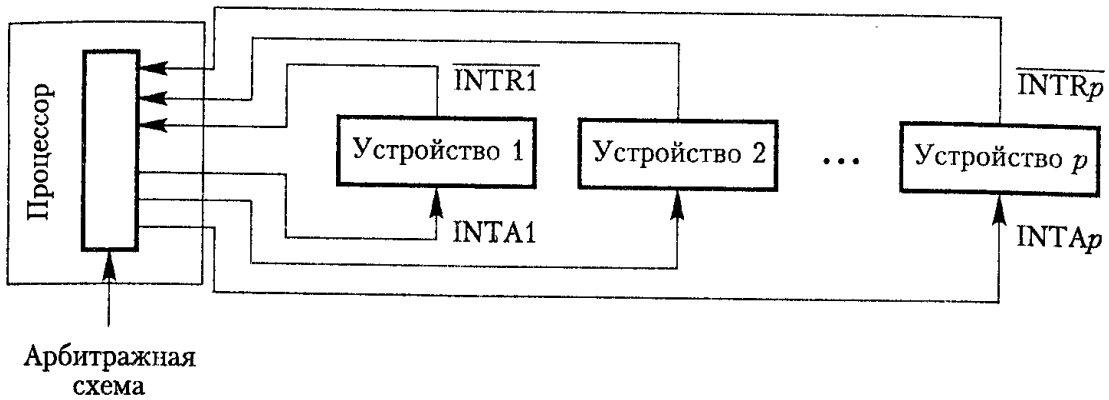


Рис. 6.2.2 – Реализация приоритетов прерываний с использованием индивидуальных линий подтверждения прерывания

При одновременном поступлении запросов прерываний от двух или более устройств процессор должен располагать средствами для выбора между этими запросами. При использовании схемы приоритетов, показанной на рис. 6.2.2, решение получается довольно простым: процессор выбирает запрос с наивысшим приоритетом. Если несколько устройств исполь-

зуют одну линию запроса прерывания, как на рис. 6.2.2, то в таком случае проще всего опрашивать регистры состояния устройств ввода-вывода. Причем приоритеты этих устройств будут определяться порядком их опроса. При использовании векторных прерываний выбирается только одно устройство, которое должно отправить свой код вектора прерывания. Широко распространена схема соединения устройств в виде гирляндной цепи (рис. 6.2.3 а). При такой схеме линия запроса прерывания $\overline{\text{INTR}}$ является общей для всех устройств, а линия подтверждения прерывания INTA соединяет устройства в гирляндную цепь, так что сигнал по очереди проходит через каждое из них.

Когда несколько устройств одновременно генерируют запрос прерывания и активизируется линия $\overline{\text{INTR}}$, процессор отвечает установкой сигнала на линии INTA в значение 1. Первым этот сигнал получает устройство 1. Если обслуживание ему не требуется, оно пересылает сигнал устройству 2. Если же устройство отправило запрос прерывания и ждет ответа, оно блокирует сигнал INTA и помещает свой идентификационный код на линии данных. Таким образом, в гирляндной схеме наивысший приоритет имеет устройство, которое ближе всего с точки зрения схемы подключения расположено к процессору (основание).

Для реализации схемы, приведенной на рис. 6.2.3, а, требуется значительно меньше соединений. Процессор может выбирать устройства с учетом их приоритетов. Эти схемы можно объединить и в более универсальную структуру. На рис. 6.2.3, б устройства объединены в группы, каждой из которых назначен свой приоритет. Внутри группы устройства соединены в гирляндную цепь. Такая структура используется во многих компьютерных системах.

Ввод-вывод по прерываниям процессора с точки зрения производительности ЭВМ более эффективен, так как время ожидания готовности ПУ процессор может использовать для выполнения другой работы (программы). При этом способе в момент готовности (F=1) контроллер ПУ вырабатывает сигнал прерывания, по которому процессор прерывает выполнение текущей программы и приступает к обслуживанию данного ПУ. Обслуживание осуществляется путем передачи управления специальной подпрограмме обмена, написанной для данного ПУ, которая обеспечивает обмен с РД ПУ. После ее завершения управление возвращается прерванной программе. Сигналов прерывания вырабатывается столько, сколько байтов информации вводится (выводится).

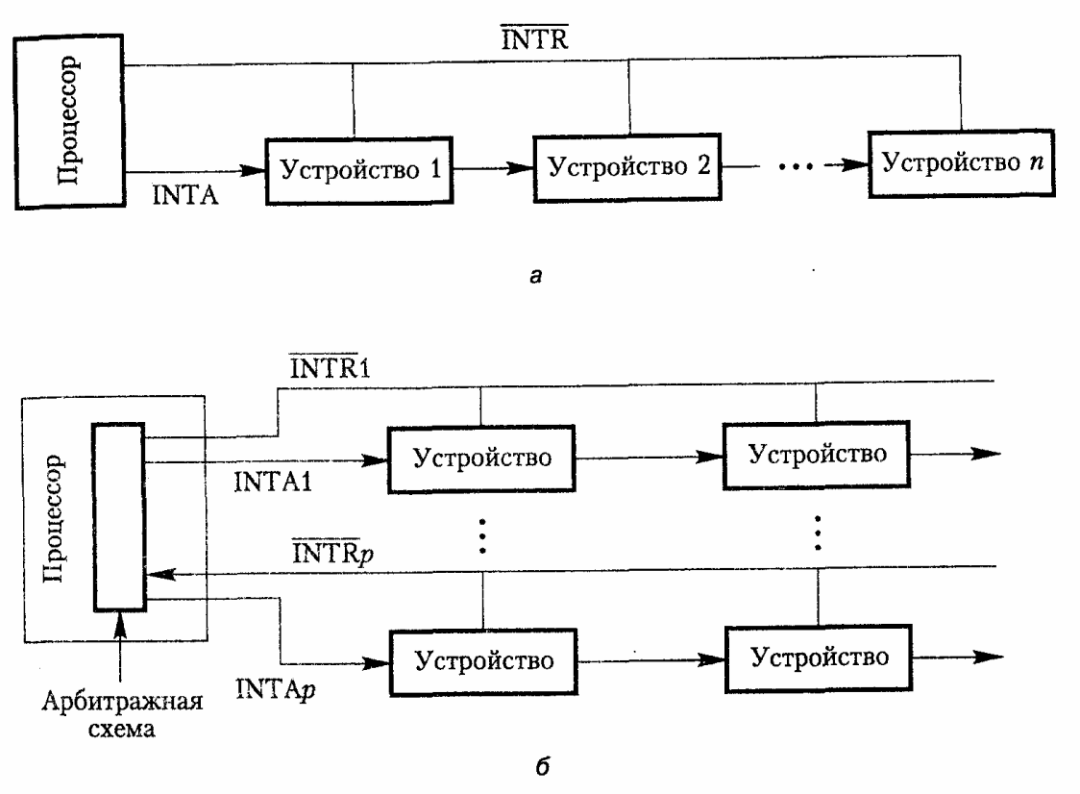


Рис. 6.2.3 – Схемы приоритетов прерываний:
гирляндная цепь (а); группы с приоритетами (б)

Недостаток обмена по прерываниям процессора:

- накладные расходы времени на передачу одного байта данных;
- время на завершение текущей команды, так как прерывание наступает только между командами;
- время на сохранение информации о состоянии прерванной программы (обычно в стеке);
- время на переход к подпрограмме (обычно по вектору прерывания);
- время на восстановление состояния и возврат из подпрограммы.

Эти расходы обычно многократно превышают время обмена между РД ПУ и ячейкой памяти. Поэтому обмен по прерываниям процессора применяется для обслуживания медленнодействующих ПУ (клавиатура, принтер, и т. п.).

6.3 Ввод-вывод с прямым доступом к памяти

Для быстрой пересылки больших блоков данных применяется механизм прямого доступа к памяти (ПДП, DMA – Direct Memory Access), минуя процессор, который отключается от шин адреса и данных. Процессор отвечает только за программирование DMA: настройку на определен-

ный тип передачи, задание начального адреса и размера массива обмениваемых данных.

Операции ПДП выполняются управляющей схемой (контроллер ПДП), входящей в состав интерфейса устройства ввода-вывода. Контроллер ПДП выполняет ту же задачу, что и процессор, обращающийся к основной памяти. Для каждого пересылаемого слова он генерирует адрес памяти и сигналы шины, управляющие пересылкой данных. Контроллер ПДП увеличивает адрес, по которому будет записываться каждое следующее слово, и отслеживает количество таких операций.

Пока контроллер ПДП производит пересылку данных, запросившая ее программа не может продолжать свою работу, и процессор часто используется для выполнения другой программы. По окончании пересылки процессор информируется с помощью сигнала прерывания и может вернуться к исходной программе.

Пример использования прямого доступа к памяти в компьютерной системе приведен на рис 6.3.2. Контроллер ПДП соединяет шину компьютера с высокоскоростной сетью. Если контроллер управляет двумя дисками, то поддерживаются два канала передачи данных. Для этого в контроллере имеется два набора регистров, предназначенных для хранения адресов памяти и счетчиков слов.

Для пересылки блока данных из основной памяти на диски сначала программа записывает адрес и значение счетчика слов в регистры соответствующего канала ПДП дискового контроллера и предоставляет контроллеру информацию, позволяющую идентифицировать данные при их чтении с диска. После этого контроллер ПДП уже независимо от процессора выполняет указанную операцию. При ее завершении в регистре состояния и управления канала ПДП устанавливается разряд DONE. Если в это время разряд IE равен 1, контроллер направляет процессору запрос прерывания и устанавливает разряд IRQ. Регистр состояния может использоваться и для хранения другой информации, например значения индикатора, указывающего, успешно ли выполнена пересылка и не произошло ли при этом каких-либо ошибок.

Процессор и контроллеры ПДП обращаются к памяти поочередно. Запросы устройств ПДП на использование шины всегда имеют более высокий приоритет, чем запросы процессора. Среди устройств ПДП наивысшим приоритетом обладают высокоскоростные внешние устройства, в том числе диски, скоростной сетевой интерфейс, графический дисплей. Применяемая при этом технология чередования называется захватом циклов. В качестве альтернативы контролеру ПДП для пересылки блока данных без прерываний может быть предоставлен монопольный доступ к основной памяти. Такой режим называется блочным или монопольным.

Рис. 6.3.2 – Использование контролеров ПДП в компьютерной системе

Большинство контроллеров ПДП содержат буфер для хранения данных. Пересылка выполняется в монопольном режиме при максимальной скорости, с которой могут работать память и шина компьютера. После этого данные пересылаются из буфера со скоростью, определяемой пропускной способностью подключаемого устройства.

6.4 Организация программируемого ввода-вывода

Основной задачей передачи данных является согласование работы быстрой памяти и медленных ПУ. Согласование обычно осуществляется асинхронным методом: УВВ запускает процесс обмена, посылая сигнал запроса на ввод-вывод блока данных. Контроллер ПУ обеспечивает считывание (запись) первого (очередного) байта с носителя и вырабатывает ответный специальный сигнал – сигнал готовности к передаче (приему) второго (очередного) байта данных. По сигналу готовности УВВ (ЦП или КВВ) принимает (передает) очередной байт данных. А контроллер ПУ, передав (приняв) байт данных, сбрасывает сигнал готовности и т.д. до окончания передачи блока данных. За время передачи блока данных сигнал готовности вырабатывается n-раз, где n – объем блока данных (объем передачи).

Описанные действия, обеспечивающие согласование работы памяти и ПУ и передачу одной порции данных, принято называть алгоритмом обмена, возбуждаемым по сигналу готовности ПУ. Он состоит из следующих действий.

1. Формирование запроса на обращение к УВВ в момент готовности ПУ к обмену. По этому запросу УВВ осуществляет обмен с буферным регистром ПУ и по мере формирования слова данных УВВ формирует запрос на обращение к памяти ЭВМ.

2. В момент, когда память освобождается от обслуживания других запросов, управление памятью передается УВВ (ЦП или КВВ), которая посылает в память адрес ячейки, направление обмена (запись в память или чтение) и порцию данных (слово или байт), подлежащую записи.

3. Формирование адреса А ячейки памяти для обмена очередной порцией данных: A:=A+1(A:=A–1). Подсчет переданного количества байтов: C:=C–1.

4.Если C<>0, то переход к пункту 1. Выработка сигнала об окончании ввода (вывода) блока данных (по условию C=0).

В зависимости от способа реализации описанного алгоритма различают два способа ввода-вывода информации: 1) программного управления и 2) аппаратного управления вводом-выводом информации.

В первом случае программа ввода (вывода), обеспечивающая обмен с адресуемым ПУ, реализуется средствами процессора и соответствующим контроллером ПУ. Процессор, реализуя программу обмена, управляет работой контроллера ПУ, посылая в него соответствующие приказы.

Во втором случае алгоритм обмена реализуется средствами КВВ и контроллера ПУ, которые на этапе начальной подготовки настраиваются на конкретный обмен. Настройка осуществляется путем передачи в КВВ параметров обмена А, С и направления обмена. Настройку осуществляет ЦП. На этапе передачи данных алгоритм обмена реализуется аппаратурой КВВ и контроллера автоматически. При этом КВВ фактически обеспечивает прямой доступ к памяти, а контроллер ПУ – управление механизмом ПУ.

В случае если КВВ реализуется как ПВВ, то это также программный способ управления обменом. ЦП, выполняя специальные команды, инициирует ПВВ (на этапе начальной подготовки) и получает информацию о состоянии ввода (вывода) (на этапе завершения обмена). На этапе передачи данных вводом (выводом) управляет ПВВ, выполняя команды программы, написанной для обмена с ПУ. Кроме того, ПВВ управляет работой ПУ, посылая в контроллер ПУ соответствующие приказы.

При программной реализации алгоритма обмена, в свою очередь, различают два способа ввода-вывода, в зависимости от того, каким образом обнаруживается готовность ПУ к обмену: первый способ – путем опроса (сканирования) флага готовности и второй способ – по прерываниям процессора.

Способ обмена по опросу флага обладает непроизводительными затратами процессорного времени на ожидание готовности ПУ и ожидания флага. Поскольку быстродействие ПУ, даже самых скоростных, невелико, то время ожидания оказывается существенным. Поэтому этот способ в ЭВМ общего назначения практически не используется.

Контрольные вопросы

1. Что такое ввод-вывод с отображением в память?
2. Роль прерываний при организации ввода-вывода?
3. Почему прямой доступ в память предпочтительней ввода-вывода с прерываниями?
4. Что такое маска прерываний и когда она используется?
5. В чем особенности гирляндной цепи?
6. Пояснить роль портов при вводе-выводе?

Глава 7. ШИНЫ И ИНТЕРФЕЙСЫ

7.1 Понятие интерфейса

В основу построения современных компьютеров положен магистрально-модульный принцип, позволяющий организовывать системы различных конфигураций из типовых модулей. Основные положения этого принципа:

- все устройства компьютера (процессоры, ЗУ, контроллеры и др.) представлены в виде модулей, совместимых на конструктивном, электрическом и функциональном уровнях;
- объединение модулей в систему осуществляется на основе одной или нескольких магистралей.
- объединение модулей в систему осуществляется по определенным правилам сопряжения, называемым интерфейсами.

Интерфейс – это совокупность аппаратных и программных средств, реализующих протоколы (правила и стандарты) по организации связей в магистрально-модульной системе. Классификация интерфейсов.

- по способу соединения модулей в структуру интерфейсы различают:
- по способу передачи информации: параллельные, последовательные и параллельно-последовательные.
- по принципу обмена: синхронные и асинхронные.
- по режиму передачи информации. Изолированнее шины (для прямого доступа в память) и неизолированные шины.

Основные элементы интерфейса:

- совокупность правил обмена (протокол обмена);
- аппаратная часть интерфейса;

• программное обеспечение интерфейса (алгоритм управления обменом, реализующий протокол обмена).

Современные интерфейсы, обычно делятся на три класса: параллельные, последовательные и комбинированные.

Параллельный интерфейс обеспечивает однонаправленную передачу n-разрядного двоичного кода (слова). Параллельные интерфейсы обеспечивают высокую пропускную способность, которая измеряется количеством байтов информации в единицу времени (секунду).

Электрические цепи интерфейса в зависимости от их назначения принято разделять на две основные группы: информационные и управляющие. Совокупность (набор) электрических цепей, объединенных в группу по назначению, и принято называть шиной, т.е. различают шины информационные ШИ и управления ШУ.

Информационные шины (ШИ) используются для передачи в разные моменты времени данных, команд и адресов. Тип передаваемой информации указывается (сообщается) приемному устройству путем посылки осведомительного сигнала: D, C, A (100 – данные, 001 – адрес, 010 – команда).

Цепи, образующие шину управления ШУ, используются для передачи различных сигналов управления, т.е. сигналов, которые формируются одним устройством (модулем), передаются по цепям ШУ и используются для управления другим устройством (модулем) в процессе обмена информацией по ШИ. Управляющие сигналы используются для синхронизации обмена. Для передачи информации по ШИ (по цепям параллельного интерфейса) используется один из двух способов (принципов) управления: синхронный или асинхронный.

При **синхронном** способе управление передачей осуществляется сигналом синхронизации (стробом), вырабатываемый передающим устройством и поступающим в приемное устройство, где используется для приема информации.

Достоинство способа – простота.

Недостатки:

- потери времени для тех пар устройств, для которых фактическое время передачи меньше T_1;

- низкая надежность передачи, так как нет сигнала, что приемное устройство приняло информацию по СС.

В тех случаях, когда разброс времени передачи большой, целесообразно применять второй способ – **асинхронный** («запрос-ответ»). При этом способе приемное устройство, принявшее информацию по сигналу синхронизации СС, вырабатывает ответный сигнал, подтверждающий факт приема информации с ШИ.

Область применения асинхронных интерфейсов – подключение устройств различного быстродействия, например периферийных устройств.

Шина процессора представляет шину, управляемую теми же сигналами, что и микросхемой процессора. К ней могут быть подключены устройства, которым требуется очень высокая скорость взаимодействия с процессором, и в частности, основная память. На материнской плате обычно имеется еще одна шина, способная поддерживать большее количество устройств. Эти две шины соединены между собой с помощью специальной схемы, называемой *мостом* и предназначенной для преобразования сигналов в соответствии с протоколами, регулирующими применение этих двух шин. Шина, логически выходящая на контакты микропроцессора, называется локальной.

Структура шины зависит от электрических характеристик процессора, в том числе от его тактовой частоты. Однако на шину расширения эти ограничения не распространяются, поэтому для нее можно использовать стандартную схему сигналов.

Широко применяются три стандарта шин: PCI (Peripheral Component Interconnect), SCSI (Small Computer Systems Interface) и USB (Universal Serial Bus), рис. 7.1.4.

Стандарт PCI определяет шину расширения на материнской плате. Шины стандарта SCSI и USB предназначены для подключения дополнительных устройств как внутри, так и вне корпуса компьютера. SCSI представляет собой высокоскоростную параллельную шину, предназначенную для подключения таких устройств, как диски и дисплеи. Шина USB поддерживает последовательную передачу данных. Она используется для подключения самого разнообразного оборудования, от клавиатур до игровых устройств, а также для сетевых соединений.

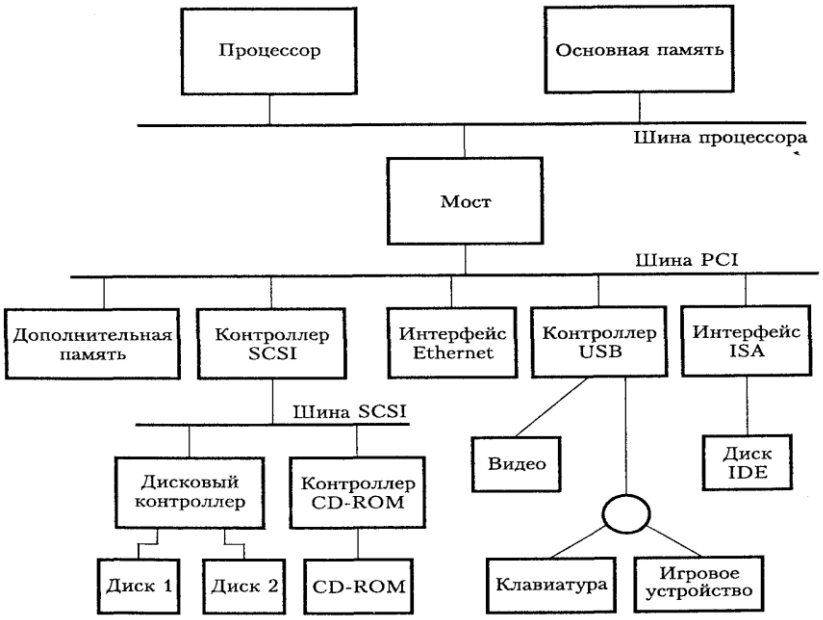


Рис. 7.1.4 – Пример компьютерной системы, в которой используется несколько стандартов интерфейса

7.2 Шины

Процессор, основная память и устройства ввода-вывода соединяются посредством шины, основным назначением которой является предоставление канала связи для пересылки данных. Пересылка данных по шине подчиняется определенному набору правил, (шинный протокол), управляющих поведением соединенных с шиной устройств, а также последовательностью выдачи информации на шину, управляющих сигналов и т.п.

Линии шины, используемые для пересылки данных, бывают трех типов: линии данных, линии адреса и управляющие линии. Управляющие сигналы определяют, какую операцию – чтения или записи – следует выполнить. Обычно для этой цели используется линия $R/\overline{W}$. Значение 1 на этой линии соответствует операции чтения, а значение 0 – операции записи. Когда команда допускает использование операндов разных размеров, например байтов, слов и длинных слов, размер данных также указывается на управляющих линиях.

Сигналы управления шиной используются для синхронизации операций и определяют, в какой момент процессор и устройства ввода-вывода могут поместить данные на шину или прочитать их. Для тактирования пересылки данных по шине синхронные и асинхронные схемы..

В любой операции пересылки данных по шине одно из устройств играет роль хозяина шины. Это устройство инициирует пересылку данных с помощью команд чтения или записи. Поэтому его можно назвать инициатором. Обычно хозяином шины является процессор, но эту роль могут выполнять и другие устройства, поддерживающие функцию прямого доступа к памяти. Устройство, к которому обращается хозяин шины, называется подчиненным или целевым.

В случае синхронной шины все устройства получают синхронизирующую информацию по общей тактовой линии. На эту линию подаются тактовые импульсы со строго фиксированной частотой. Промежуток времени между последовательными тактовыми импульсами в простейшей синхронной шине составляет цикл шины, в течение которого выполняется одна операция пересылки данных.

Пример последовательность событий, происходящих при выполнении операции ввода (чтения) данных приведен на рис 7.2.1.

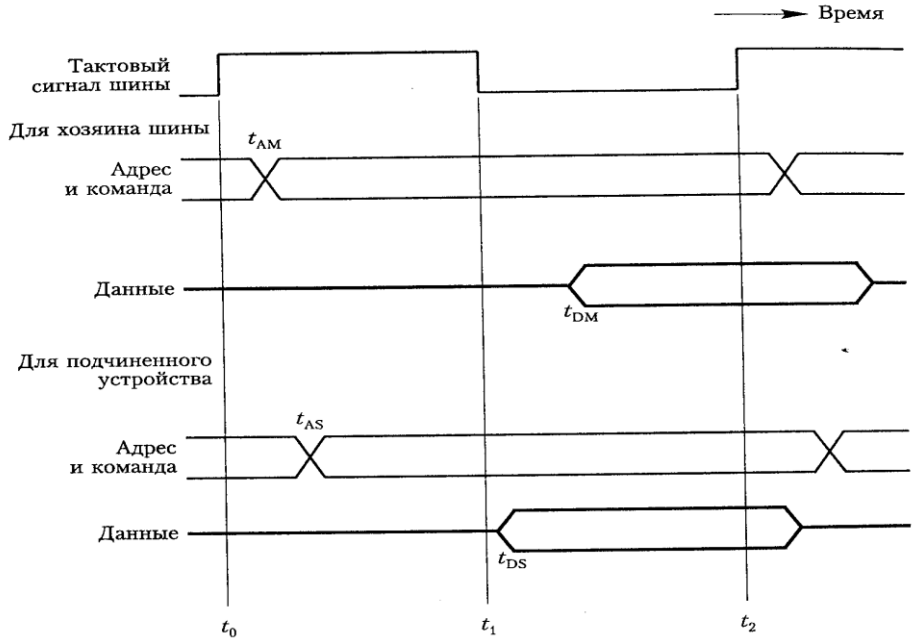


Рис. 7.2.1 – Временная диаграмма для операции пересылки входных данных

В момент времени t_0 хозяин шины помещает на адресные линии адрес устройства и отсылает по управляющим линиям необходимую команду. В этой команде определяется операция ввода и, если нужно, задается длина считываемого операнда. Инфор-

мация передается по шине со скоростью, определяемой ее физическими и электрическими характеристиками. В течение промежутка времени от t_0 до t_1 информация на шине ненадежна, поскольку состояние сигналов изменяется. В момент времени t_1 адресуемое подчиненное устройство помещает запрошенные входные данные на линии данных.

В конце тактового цикла, то есть в момент времени t_2, хозяин шины стробирует (сохраняет в буфере в указанный момент времени) данные на линиях данных в свой входной буфер. Для правильной загрузки данных в любое устройство хранения, в том числе в регистр на основе триггеров, они должны находиться на его входе в течение времени, достаточного для их сохранения. Поэтому период времени $t_2 - t_1$ должен быть больше максимального времени распространения сигнала по шине в сумме со временем установки входного буферного регистра хозяина шины.

Похожая процедура выполняется и при выводе данных. Хозяин шины помещает на линии данных выходные сведения, а на линии адреса и управляющие линии – адрес и команду. В момент времени t_2 адресуемое устройство стробирует линии данных и загружает информацию в свой буфер данных.

Для исключения влияния задержек в большинство шин включают управляющие сигналы, передаваемых в качестве ответа устройства и информирующих хозяина шины о том, что подчиненное устройство распознало адрес и готово участвовать в операции пересылки данных. Для упрощения используется сигнал высокой частоты, при котором цикл пересылки занимает несколько тактов. Таким образом, количество тактов, затрачиваемых на операцию пересылки данных, зависит от конкретной пары устройств.

Пример реализации такого подхода приведен на рис. 7.4.2. В течение такта 1 хозяин шины пересылает адрес и информацию о команде, запрашивая операцию чтения. Подчиненное устройство получает и декодирует эту информацию. По следующему переднему фронту тактового сигнала, то есть в начале такта 2, устройство принимает решение ответить на запрос и начинает процедуру доступа к запрошенным данным. На извлечение данных требуется время, поэтому подчиненное устройство не может ответить немедленно. На третьем такте данные готовы и помещаются на шину. В то же время подчиненное устройство выдает управляющий сигнал, называемый

Slave-ready (подчиненный готов). Хозяин шины, ожидавший этого сигнала, стробирует данные в свой входной буфер в конце такта 3. На этом операция пересылки данных по шине заканчивается, и хозяин шины может отправить по ней новый адрес, чтобы на такте 4 начать другую операцию пересылки.

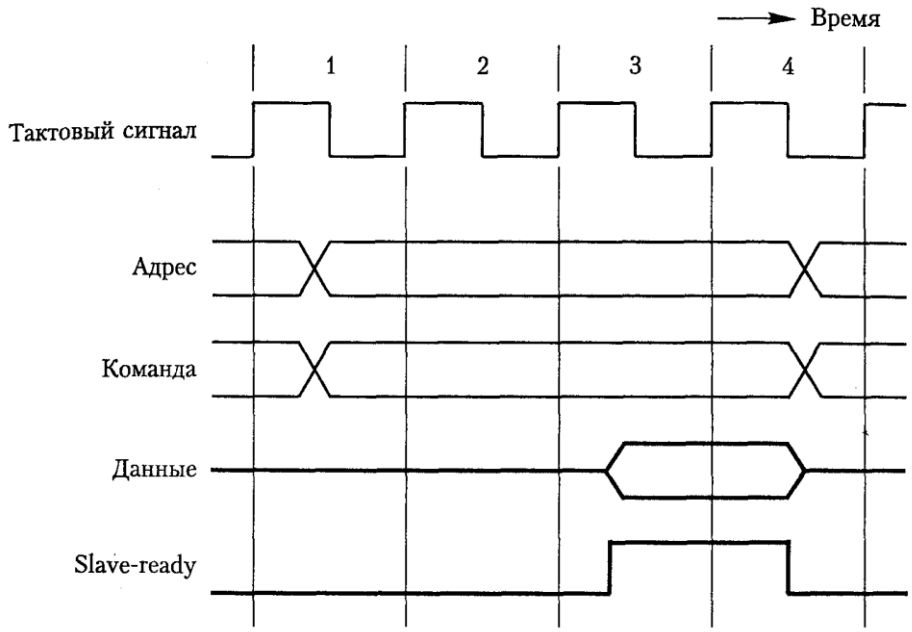


Рис. 7.4.2 – Пересылка данных с использованием нескольких тактов

Сигнал Slave-ready – это направляемое подчиненным устройством хозяину шины сообщение, говорящее о том, что им отправлены правильные данные. Подчиненное устройство отвечает во время такта 3. Другое устройство может ответить раньше или позже. Таким образом, сигнал Slave-ready позволяет изменять длительность операции пересылки данных в соответствии с возможностями конкретного устройства. Если адресуемое устройство не отвечает, хозяин шины ждет в течение заранее определенного количества тактов, а затем отменяет операцию. Отсутствие ответа может быть результатом отправки неверного адреса или некорректного функционирования устройства.

Другая схема управления пересылкой данных по шине использует механизм подтверждения связи (квитирования) между хозяином шины и подчиненным устройством. Концепция квитирования является обобщением идеи использования сигнала Slave-ready. В схеме с квитированием тактовая линия заменяется двумя управляющими линиями синхронизации: Master-ready и Slave-

ready. Первая принадлежит хозяину шины, который передает по ней сигнал готовности к транзакции, а по второй отвечает подчиненное устройство.

Хозяин шины помещает на нее адрес и информацию о команде, эатем по линии Master-ready он сообщает об этом всем устройствам. В ответ подключенные к шине устройства декодируют адрес. То устройство, для которого предназначена команда, выполняет таковую и информирует об этом хозяина шины по линии Slave-ready. Хозяин дожидается этого сигнала и только после этого удаляет с шины свои сигналы. В случае операции чтения он стробирует данные в свой входной буфер.

7.3 Шина PCI

Шина PCI – это системная шина появившейся в процессе стандартизации для высокоскоростных дисковых и графических устройств. Она поддерживает функции, типичные для шины процессора, но в стандартном формате, независимо от типа процессора. Подключенные к этой шине устройства представляются процессору непосредственно соединенными с его собственной шиной. Им назначаются адреса из адресного пространства памяти процессора.

В современных компьютерах при выполнении операции пересылки данных перемещаются блоки информации. Поэтому необходима кэш-память. Данные пересылаются между кэш-памятью и основной памятью в виде пакетов, по несколько слов в каждом. Участвующие в такой пересылке слова сохраняются по последовательным адресам памяти. Шина PCI предназначена для поддержки режима работы чтения или записи пакетов длиной в одно слово.

Шина поддерживает три независимых адресных пространства: памяти, ввода-вывода и конфигурации. Адресное пространство ввода-вывода используется такими процессорами, как Pentium, имеющими отдельное адресное пространство ввода-вывода. Стандартом PCI для совместимости с большим количеством устройств рекомендуется применять именно этот подход.

На рис. 7.3.1 показана схема, на которой основная память компьютера непосредственно соединена с шиной процессора. Мост PCI создает для основной памяти отдельное физическое подключе-

ние. По причинам электротехнического характера шина может быть разделена на сегменты, соединенные между собой посредством мостов. Независимо от того, к какому сегменту шины подключено конкретное устройство, оно может отображаться в адресное пространство процессора.

Сохранение адресной информации на шине необходимо, пока не выбрано подчиненное устройство, сохраняющее адрес в своем внутреннем буфере. Адрес должен находиться на шине в течение одного такта, после чего адресные линии могут быть освобождены для пересылки данных в последующих тактах. Благодаря этому снижается стоимость операции пересылки, зависящая от количества проводов шины.

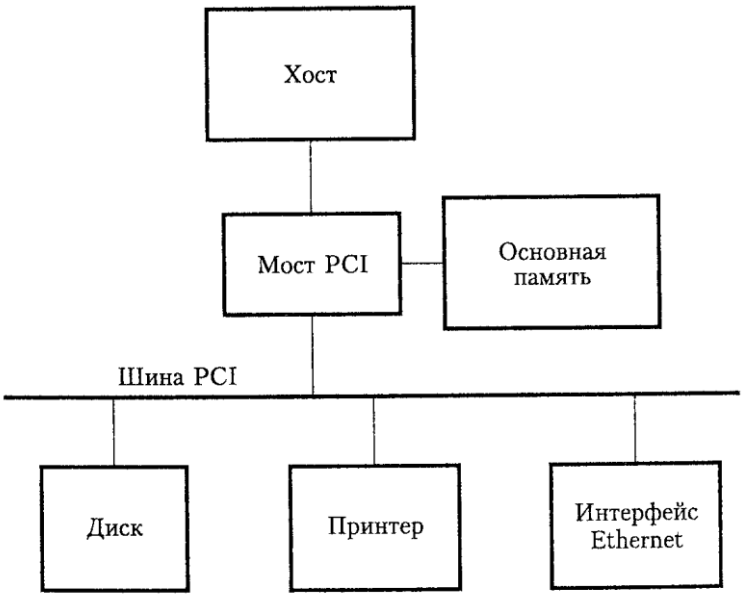


Рис. 7.3.1 – использования шины PCI
в компьютерной системе

Хозяином шины в каждый конкретный момент времени может быть только одно устройство. Это устройство имеет право инициировать операции пересылки данных с помощью команд чтения и записи. Согласно терминологии PCI, хозяин шины называется инициатором, которым может быть либо процессор, либо контроллер ПДП. Адресуемое устройство, отвечающее на команды чтения и записи, называется целевым.

Полная операция пересылки данных по шине, включающая пересылку адреса и пакета данных, называется транзакцией. Пе-

ресылка отдельного слова в ходе транзакции называется фазой. Максимальная скорость передачи данных по шине 528 Мб/с.

7.4 Шина SCSI

Аббревиатура SCSI (Small Computer Systems Interface) означает интерфейс малых компьютерных систем. Стандарт определен Национальным институтом стандартизации США (American National Standards Institute, ANSI). Согласно спецификации стандарта, такие устройства, как диски, должны соединяться с компьютером при помощи 50-проводного кабеля длиной до 25 м, по которому данные могут передаваться со скоростью до 5 Мбайт/с.

Определены стандарты SCSI-2 и SCSI-3, каждый из которых имеет несколько опций. Узкая (narrow SCSI) шина SCSI имеет восемь линий данных и передает данные по одному байту. Широкая шина SCSI (wide SCSI) состоит из 16 линий данных и передает информацию по 16 бит. Существует несколько вариантов электрических сигнальных схем. Передача данных по шине SCSI выполняется в асимметричном режиме (Single-Ended SCSI, SE), при использовании для каждого сигнала проводника с общим замыканием через «землю» для всех сигналов. Для каждого сигнала также может предназначаться отдельный обратный провод. В этом случае возможно использование двух уровней напряжения.

Для различных версий SCSI используются разъемы: 50-, 68- и 80-контактные. Последняя версия стандарта поддерживает скорость передачи с поддержкой 640 Мбайт/с. Максимальная скорость передачи зависит от длины кабеля и количества подключенных к нему устройств. Для достижения максимальной скорости обычно используют кабель длиной не более 1,6 м для сигнальной схемы SE и не более 12 м для сигнальной схемы LVD. Для подключения удаленных устройств предоставляют специальные расширители шины. Максимальная «вместимость» шины составляет 8 устройств для Narrow SCSI и 16 устройств для Wide SCSI.

Шина SCSI соединяется с шиной процессора через SCSI-контроллер. Для пересылки пакетов данных от главной памяти к устройству и в обратном направлении применяется технология прямого доступа к памяти. Пакет может содержать блок данных,

команды, направляемые процессором устройству, или информацию о состоянии устройства.

Принцип взаимодействия с дисками отличается от принципа взаимодействия с основной памятью. Данные хранятся на диске блоками, называемыми секторами, каждый из которых может содержать несколько сот байтов. Данные не всегда записываются в последовательно расположенные секторы, потому что в одних секторах могут уже храниться ранее записанные данные; другие могут быть дефектными, а следовательно, должны быть пропущены. Поэтому для обслуживания запроса чтения или записи может потребоваться доступ к обязательно последовательным секторам диска.

Протокол SCSI ориентирован на режим работы с задержками из-за механической природы диска. Обращение к первому сектору, из которого считываются или в который записываются данные, выполняется с довольно значительной задержкой порядка нескольких миллисекунд. После этого некоторый объем данных пересылается с очень высокой скоростью, но затем может произойти еще одна задержка и т.д.

Контроллер, подключенный к шине SCSI, может быть инициатором или целевым устройством. Инициатор обладает способностью выбирать конкретное целевое устройство и направлять ему команды, определяющие выполняемую операцию. Контроллер со стороны процессора функционирует как инициатор. Дисковый контроллер является целевым устройством и выполняет команды, получаемые от инициатора. Инициатор устанавливает логическое соединение с выбранным целевым устройством. Соединение может временно разрываться и возобновляться по мере возникновения необходимости в пересылке команд и пакетов данных. При временном разрыве соединения шина используется другими устройствами. Эта способность чередовать запросы пересылки данных определяет высокую производительность шины.

7.5. Шина USB

Современные компьютерные системы включают множество устройств: клавиатуры, микрофоны, цифровые видеокамеры, динамики, дисплеи. Для их подключения применяется универсальная процессорно-независимая последовательная шина – Universal Serial Bus (USB), являющаяся промышленным стандартом. USB поддер-

живает два режима функционирования, получивших названия низкоскоростной (1,5 Мбит/с) и полноскоростной (12 Мбит/с). В последней версии этой спецификации, USB 2.0, введен третий режим, названный высокоскоростным (480 Мбит/с).

В USB реализованы следующие преимущества:

- простая, дешевая система соединения, позволяющая преодолевать сложности, возникающие из-за ограниченного числа имеющихся в компьютерах портов ввода-вывода;
- обеспечен широкий диапазон параметров пересылаемых данных, присущих различным устройствам ввода-вывода, в том числе модемам (скорость, объемы и временные характеристики процесса обмена данными);
- облегчен процесс подключения устройств за счет поддержки режима plug-and-play.

Система обнаруживает новое устройство, идентифицирует его и соответствующее ему программное обеспечение (драйвер, а также другие необходимые для его работы средства), назначает адреса и устанавливает логические соединения для взаимодействия с другими устройствами.

Для шины USB выбран последовательный формат пересылки данных, обеспечивающий ее наименьшую стоимость и наибольшую гибкость. Тактирующий сигнал и данные кодируются вместе и передаются как единый сигнал. В результате нет никаких ограничений в отношении тактовой частоты или расстояний, связанных со сдвигом данных, благодаря чему становится возможной высокая пропускная способность соединений с высокой тактовой частотой. Шина USB поддерживает три скорости пересылки данных, а именно 1,5; 12 и 480 Мбит/с, что соответствует нуждам самых разных устройств ввода-вывода.

Шина USB имеет древовидную структуру для одновременного подключения большого количество устройств, удаляемых и подсоединяемых в любое время, (рис. 7.5.1). В узлах дерева располагаются устройства, называемые хабами и действующие как промежуточные управляющие компоненты между хостом и устройствами ввода-вывода. Корневой хаб соединяет все дерево с хост-компьютером. Листьями дерева являются устройства ввода-

вывода (клавиатура, динамики, соединение с сетью, цифровой телевизор и т.п.), в терминологии USB называемые *функциями.*

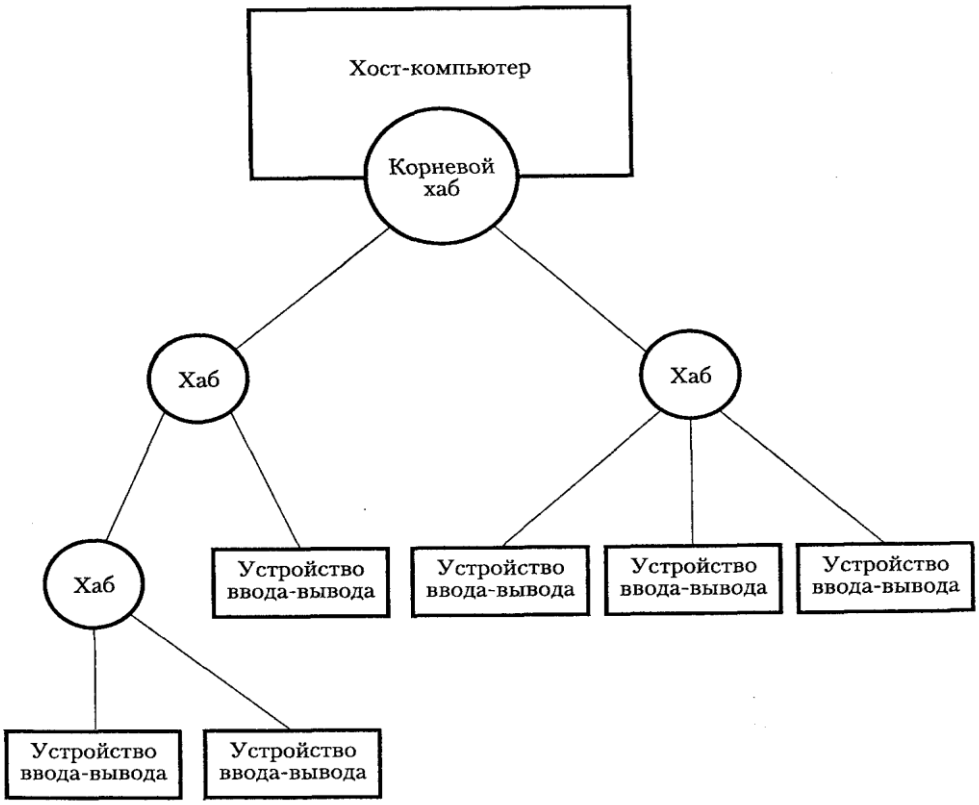


Рис. 7.5.1 – Структура дерева USB

Древовидная структура позволяет соединять множество устройств с помощью простых последовательных соединений «точка-точка». Каждый хаб имеет ряд портов, к которым можно подключать любые устройства, в том числе и другие хабы. В нормальном режиме хаб копирует полученное входное сообщение в свои выходные порты. В результате посланное компьютером сообщение передается всем устройствам ввода-вывода, но отвечает на него только адресуемое устройство.

В основе функционирования шины USB лежит принцип опроса устройств. Устройство может отослать сообщение только в ответ на запрос хоста. Поэтому передаваемые хосту сообщения не конфликтуют и не пересекаются друг с другом, и никакие два устройства не могут отослать сообщения одновременно. Это ограничение позволяет применять простые и недорогие хабы. Если к шине наряду с низко- и полноскоростными линиями подключе-

ны высокоскоростные USB 2, шина может работать в разделенном режиме, чтобы не задерживать сообщения, направляемые по высокоскоростным линиям. В таком режиме сообщения, передаваемые через высокоскоростное соединение, всегда пересылаются на высокой скорости до хаба, к которому подключено низкоскоростное устройство. А от этого хаба к устройству – на низкой. Последний этап пересылки займет много времени, в течение которого разрешается производить высокоскоростной трафик к другим узлам. В течение этого времени шина делится между высокоскоростным и низкоскоростным трафиками.

Каждому устройству на шине USB (хаб или устройство ввода-вывода) назначается 7-битовый адрес. Этот адрес локален для дерева USB и никак не соотносится с адресами, используемыми на шине процессора. К хабу может быть подключено любое количество устройств и других хабов, адреса которых назначаются произвольным образом. Когда устройство активизируется или подключается к хабу, оно имеет адрес 0. Аппаратное обеспечение хаба обнаруживает новое устройство, о чем делает соответствующую пометку в своей информации о состоянии. Периодически хост опрашивает все хабы, собирая сведения об их состоянии, и узнает о добавленных или удаленных устройствах. Когда хост узнает о подключении нового устройства, он с помощью специальной последовательности команд направляет в порт хаба сигнал сброса, считывает из памяти устройства информацию о его возможностях, направляет этому устройству конфигурационную информацию и присваивает ему уникальный USB-адрес. После этого начинается обычное функционирование устройства, которое теперь имеет новый адрес.

Информация, пересылаемая через соединения USB, организуется в пакеты, каждый из которых включает один или несколько байтов данных. Существует множество типов пакетов, выполняющих разные управляющие функции.

Пересылаемую по шине USB информацию можно разделить на две категории: управляющая информация и данные. Управляющие пакеты используются для адресации устройств при инициировании пересылки данных, а также для подтверждения факта получения правильных данных и сообщений об ошибках. Пакеты данных содержат входные и выходные дан-

ные, которыми хост обменивается с устройством, и некоторую другую информацию.

Контрольные вопросы

1. В чем отличие синхронных и асинхронных шин?
2. Пояснить понятие квитирования?
3. Какой порт используется при соединении клавиатуры с компьютером?
4. В чем преимущество двойной буферизации при организации интерфейсных схем?
5. Роль контроллеров при управлении шиной?
6. Какой формат передачи данных использует шина USB?

Глава 8. ОРГАНИЗАЦИЯ ПАМЯТИ

8.1 Классификация устройств памяти

Существует большое количество типов ЗУ, используемых в ЭВМ и системах. Эти устройства различаются принципом действия, логической организацией, конструктивной и технологической реализацией, функциональным назначением и т.д.

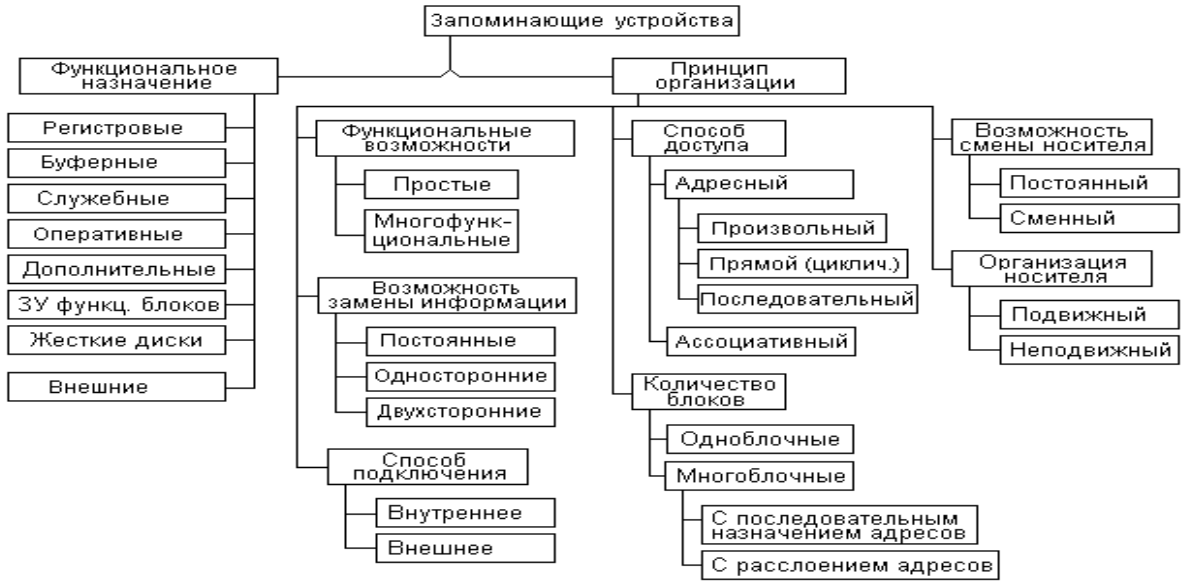


Рис. 8.1.1 – Классификация ЗУ

Классификация ЗУ по функциональному назначению:

1. Верхнее место занимают ***регистровые ЗУ***, входящие в состав процессора и представляющие набор регистров процессора. ЗУ реализованы на том же кристалле, что и процессор, и предназначены для хранения небольшого количества информации (до нескольких десятков слов, а в RISC-архитектурах – до сотни), которая обрабатывается в текущий момент времени или часто используется процессором. Это позволяет сократить время выполнения программы за счет использования команд типа регистр-регистр и уменьшить частоту обменов информацией с более медленными ЗУ ЭВМ. Обращение к этим ЗУ производится непосредственно по командам процессора.

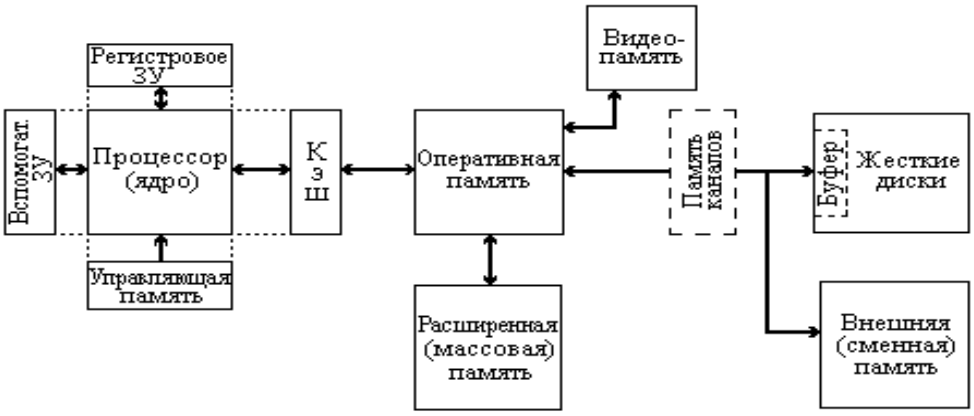


Рис. 8.1.2 – Возможный состав системы памяти ЭВМ

2. Буферные ЗУ (КЭШ) сокращают время передачи информации между процессором и медленными уровнями памяти компьютера. В буфере размещается только та часть информации из медленного ЗУ, которая используется в настоящий момент. Если доля *h* обращений к памяти со стороны процессора, удовлетворяемых непосредственно буфером (кэшем) высока (0,9 и более), то среднее время для всех обращений оказывается близким ко времени обращения к кэшу, а не к медленному ЗУ.

Двухуровневая память состоит из кэш- и оперативной памяти, как показано на рис. 8.1.2. Время обращения к кэшу t_c = 1 нс (10–9 с), время tm обращения к более медленной памяти в десять раз больше – t_m = 10 нс, а доля обращений, удовлетворяемых кэшем, h = 0,95. Тогда среднее время обращения к такой двухуровневой памяти $T_{ср}$ составит $T_{ср}$ = 1 * 0.95 + 10 * (1 – 0.95) =

1.45 нс, т.е. всего на 45% больше времени обращения к кэшу. Значение h зависит от размера кэша и характера выполняемых программ и иногда называется отношением успехов или попаданий (рис. 8.1.3).

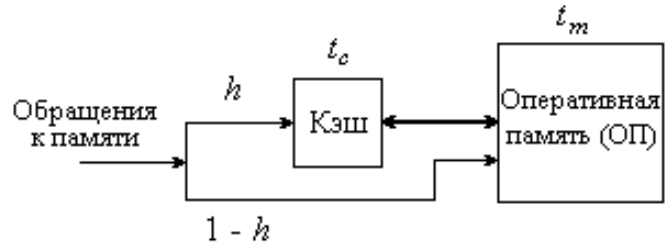


Рис. 8.1.3 – К расчету времени обращения

К расчету среднего времени обращения (t_c – время обращения к кэш-памяти, t_m – время обращения к ОП, h – доля обращений, обслуживаемых кэш-памятью, 1 – h – доля обращений, обслуживаемых ОП).

Размеры кэш-памяти существенно изменяются с развитием технологий. Кэш-память может состоять из двух (а в серверных системах – даже трех) уровней: первого (L1) и второго (L2), также отличающихся своей емкостью и временем обращения.

Конструктивно кэш уровня L1 входит в состав процессора (поэтому его иногда называют внутренним). Кэш уровня L2 либо также входит в микросхему процессора, либо может быть реализован в виде отдельной памяти. На параметры быстродействия процессора большее влияние оказывают характеристики кэш-памяти первого уровня.

Время обращения к кэш-памяти, которая обычно работает на частоте процессора, составляет от десятых долей до единиц наносекунд, т.е. не превышает длительности одного цикла процессора.

Обмен информацией между кэш-памятью и медленными ЗУ для улучшения временных характеристик выполняется блоками. Управляют этим обменом аппаратные средства процессора и операционная система.

3. Внутренним уровнем памяти являются **служебные ЗУ**:

- ЗУ микропрограмм, (управляющая память),
- вспомогательные ЗУ, используемые для управления многоуровневой памятью.

В управляющей памяти, использующейся в ЭВМ с микропрограммным управлением, хранятся микропрограммы выполнения команд процессора, а также различных служебных операций.

Вспомогательные ЗУ для управления памятью (теговая память, буфер переадресации) представляют различные таблицы для быстрого поиска информации в разных ступенях памяти, отображения ее свойств, очередности перемещения между ступенями и пр.

Емкости и времена обращения к таким ЗУ зависят от их назначения. Обычно – это небольшие (до нескольких Кбайт), но быстродействующие ЗУ.

4. Оперативное ЗУ (ОЗУ) является основным запоминающим устройством ЭВМ, в котором хранятся программы и данные, резидентные программы, модули операционной системы и т.п. Используется термин *RAM* (*random access memory*), означающий память с произвольным доступом.

Оперативная память реализуется на интегральных схемах, стандартные ее объемы гигабайты, а времена обращения – наносекунды.

5. ЗУ отдельных функциональных блоков *ко*мпьютера. Их назначение обычно сводится к буферизации данных, извлекаемых из каких-либо устройств и поступающих в них.

- видеопамять графического адаптера, используемого в качестве буферной памяти для снижения нагрузки на основную память и системную шину процессора;

- буферная память контроллеров жестких дисков, а также память, использовавшаяся в каналах (процессорах) ввода-вывода для организации одновременной работы нескольких внешних устройств.

Для видеопамяти объем может достигать величин, сравнимых с оперативными ЗУ, а быстродействие – даже превосходить быстродействие последних.

7. *Ж*есткие диски (Внешняя память). В этих ВЗУ хранится информация, используемая активно операционной системой, основными прикладными программами, а также используемыми пакетами и справочными данными.

Емкость этой ступени памяти, включающей в свой состав до десятков дисков и обеспечивающая хранение большого количе-

ства данных, зависит от области применения ЭВМ. Типовая емкость жесткого диска удваивается примерно каждые полтора года.

8. Остальные запоминающие устройства объединяются группу **внешних ЗУ**, и информация, хранимая в этих ЗУ, расположена на носителях: компакт-дисках, сменных магнитных и магнитооптических дисках, флэш-носители, стримеры, внешние винчестеры и др.

По ***функциональным возможностям*** ЗУ можно разделять:

- на простые, предназначенные для хранения информации;
- многофункциональные, позволяющие не только хранить, но и перерабатывать хранимую информацию в самих ЗУ.

По ***возможности изменения информации*** различают ЗУ:

- постоянные (или с однократной записью);
- односторонние (с перезаписью или перепрограммируемые);
- двусторонние.

В *постоянных* ЗУ (ПЗУ) информация заносится либо при изготовлении, либо посредством записи, выполняемой однократно.

Односторонними называют ЗУ, которые имеют существенно различные времена записи и считывания информации. Наиболее распространенными типами таких ЗУ являются перепрограммируемые постоянные ЗУ или компакт-диски с перезаписью – CD-RW. Время записи в устройствах этих типов значительно превышает время считывания информации.

Двусторонние ЗУ имеют близкие значения времен чтения и записи. Типичными представителями таких ЗУ являются оперативные ЗУ и ЗУ на жестких дисках.

По способу доступа различают ЗУ:

- с адресным доступом;
- с ассоциативным доступом.

При *адресном доступе* для записи или чтения место расположения информации в ЗУ определяется ее адресом. В зависимости от механизма доступа, различают следующие виды адресного доступа:

- произвольный;
- прямой (циклический);

- последовательный.

Термин «память с произвольным доступом» (*random access memory – RAM*) применяют к ЗУ, в которых выбор места хранения информации производится подключением входов и выходов элементов памяти (через буферы, усилители и логические элементы) к входным и выходным шинам ЗУ. Это наиболее быстрый вид адресного доступа, применяемый в оперативных ЗУ и кэш-памяти.

При прямом (циклическом) происходит и перемещение данных относительно механизма чтения/записи, механизма чтения/записи относительно данных или и то и другое.

При последовательном доступе к блоку данных необходимо переместить носитель так, чтобы участок, на котором располагается требуемый блок данных, оказался под блоком головок чтения/записи.

При ***ассоциативном доступе*** место хранения информации при чтении и записи определяется значением ключа поиска. Каждое записанное и хранимое в ассоциативной памяти слово имеет поле ключа. Значение этого ключа сравнивается со значением ключа поиска при чтении данных из памяти. В случае совпадения сравниваемых значений информация считывается из памяти.

По ***организации носителя различают*** ЗУ:

- с неподвижным носителем;
- с подвижным носителем.

В первых, из них носитель механически неподвижен в процессе чтения и записи информации, например, в оперативных и кэш ЗУ, твердотельных дисках, ЗУ с переносом зарядов и др.

Для ЗУ второй группы чтение и запись информации сопровождаются механическим перемещением носителя в различных ЗУ с магнитной записью, например в жестких и гибких дисках.

По ***возможности смены носителя*** ЗУ могут быть:

- с постоянным носителем;
- со сменным носителем.

В ЗУ первого вида носитель является частью самого устройства и не может быть извлечен из него в процессе нормального функционирования.

В ЗУ второй группы может устанавливаться в ЗУ и извлекаться из него в процессе работы (CD-ROM-дисководы, карты памяти, магнито-оптические диски).

По ***способу подключения*** к системе ЗУ делятся:

- на внутренние (стационарные);
- внешние (съемные).

В первом случае ЗУ, как правило, является обязательным компонентом вычислительной системы или интегрируется с другими ее компонентами (например, кэш-память).

Во втором случае устройство подключается к системе дополнительно и представляет собой отдельный блок.

По ***количеству блоков***, образующих модуль или ступень памяти, можно различать:

- одноблочные ЗУ;
- многоблочные ЗУ.

В многоблочное ЗУ входят банки памяти, допускающие возможность параллельной работы. При одновременной работе блоков нужно равномерно распределять доступ по различным блокам путем расслоения.

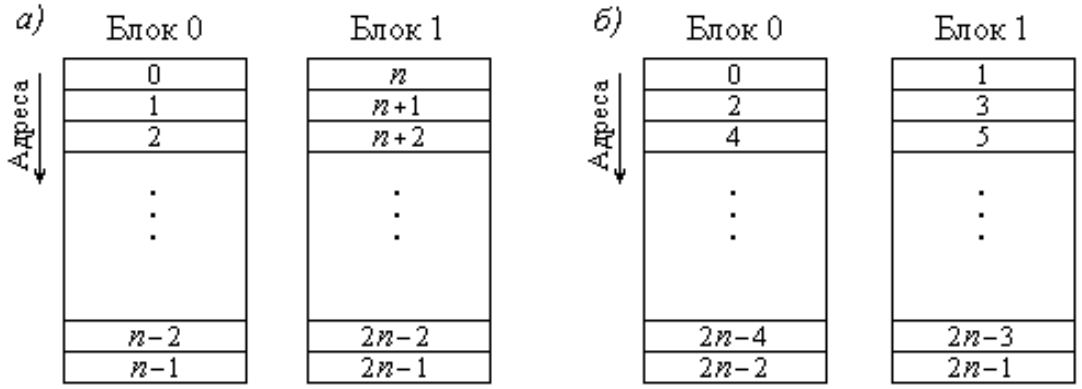


Рис. 8.1.4 – Распределение адресов адресного пространства памяти по блокам: *а* – последовательное, *б* – с расслоением по 2 блокам

8.2 Основные понятия

К числу основных относятся следующие понятия: запоминающий элемент (ЗЭ), ячейка памяти, запоминающее устройство (ЗУ), память ЭВМ.

Определение памяти (ГОСТ): память – это часть преобразователя информации, предназначенная для преобразования информации во времени (запоминания) и выдачи информации.

Функции памяти: 1) хранение информации; 2) прием информации по запросу – запись; 3) выдача информации по запросу – чтение.

Операции чтения и записи информации в память принято называть **обращением к памяти.**

Запоминающий элемент – это место хранения бита информации. Типичный пример ЗЭ – триггер. На основе ЗЭ организуется хранение более крупных единиц информации – слов.

Ячейка памяти – это фиксированная совокупность ЗЭ, обращение к которым производится одновременно как к единому целому. Ячейка памяти – это место хранения слова информации.

Доступ к информации, хранимой в ячейках памяти, организуется по адресному принципу: ячейки памяти нумеруются числами 0, 1, …, E–1, номер ячейки называют ее **адресом**. Количество ячеек памяти E – **емкость памяти** – и длина адреса связаны отношением $E = 2^m$, m – длина адреса в битах. Таким образом, адресуемым элементом памяти является ячейка, а единицей обмена с памятью является слово: за одно обращение к памяти можно прочитать или записать одно слово информации.

В памяти третьего уровня (во внешней памяти) адресуются блоки, состоящие из фиксированного количества слов. Блок является единицей обмена с внешней памятью. Блоки как ячейки памяти нумеруются номерами 0, 1, …, E–1 и рассматриваются как адреса блоков.

Доступ к ячейкам памяти с заданным адресом A обеспечивается схемой селекции, выбирающей ячейку при обращении с целью записи или чтения слова (блока) информации. В простых случаях схема селекции выполняется на основе дешифратора, в более сложных случаях, кроме дешифратора, используются и другие схемы и механизмы.

Определение запоминающего устройства (ЗУ). Совокупность ячеек памяти, объединенных в единое целое схемой селекции, называется ЗУ.

Запоминающее устройство состоит из двух частей: запоминающих элементов ЗЭ и блока управления БУ (рис. 8.2.1). Блок

управления БУ обеспечивает выбор ячейки с адресом А: доступ к этой ячейке, и управление операциями чтения или записи по запросам ЧТ или ЗП.

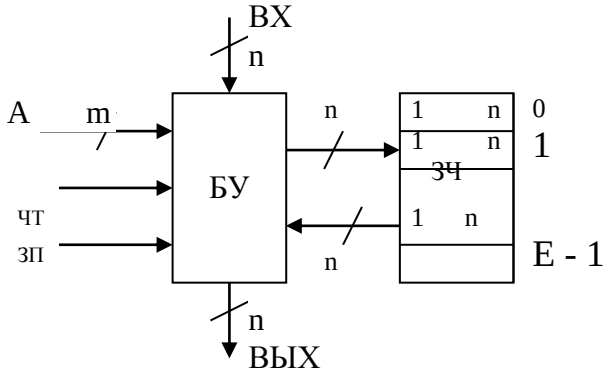


Рис. 8.2.1 – Структура ЗУ

Основные характеристики ЗУ: емкость, быстродействие, надежность, стоимость.

Емкость ЗУ определяется количеством ячеек, максимальным количеством информации, которая одновременно может храниться в ЗУ.

Быстродействие ЗУ определяется количеством операций обращения в единицу времени и зависит от продолжительности одной операции обращения: Vзу = 1/tобр.

В общем случае время обращения различно при выполнении операций ЧТ и ЗП:

$$t_{обр}^{чт} = \tau_д + \tau_{чт} + {}_{рег},$$

$$t_{обр}^{зп} = \tau_д + \tau_{подг} + \tau_{зп}.$$

Здесь $t_д$ – время доступа к информации (к ячейке ЗУ); $\tau_{чт}$ – время выдачи содержимого ячейки на выходную шину ЗУ; $\tau_{рег}$ – время регенерации – повторной записи в ячейку информации, разрушенной при чтении; $\tau_{подг}$ – время на подготовку ячейки к записи новой информации.; $\tau_{зп}$ – время собственно записи слова с входной шины ЗУ в выбранную схемой селекции ячейку ЗУ.

В зависимости от времени обращения (от быстродействия) ЗУ делятся на три класса: сверхоперативные, оперативные и внешние.

Надежность ЗУ обычно характеризуется временем наработки на отказ.

Память современных ЭВМ строится по трехуровневой иерархической схеме (рис. 8.2.2).

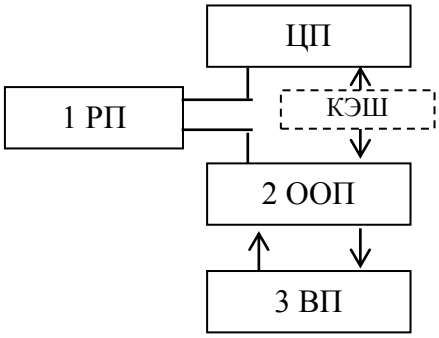


Рис. 8.2.2

Первый уровень памяти состоит из РОН и кэш-памяти и строится на базе сверхоперативных ЗУ (СОЗУ). Второй уровень памяти состоит из ячеек ООП и строится на базе ОЗУ. Третий уровень памяти составляют ЗУ, информация которых недоступна ЦП. (накопители на магнитных и оптических дисках, накопители на магнитных лентах (НМЛ) и др.

Память первого уровня (СОЗУ) предназначена для повышения быстродействия процессора, поэтому реализуется на основе СОЗУ типа SRAM небольшой емкости и сверхвысокого быстродействия.

Память второго уровня ОЗУ предназначена для хранения программ и данных: емкость большая, быстродействие высокое, удельная стоимость умеренная. Строится на основе ОЗУ динамического типа (SRAM).

Память третьего уровня (ВЗУ) предназначена для хранения больших объемов информации. Во ВЗУ размещаются базы данных, пакеты прикладных программ (ППП) и т.п. Она строится на базе различных внешних ЗУ типа НМД, МОД, НМЛ и др. очень большой емкости, но небольшого быстродействия и удельной стоимости.

8.3 Принципы и организация кэш-памяти

Кэш-память находится на верхних уровнях иерархии памяти, играет роль своего рода буфера между процессором и оперативной памятью, обеспечивая ускорение доступа к последней.

Кэш прямого отображения простой вариант быстрого определения того, имеется ли в данный момент в кэш-памяти информация, затребованная очередным обращением к оперативной памяти. Это обеспечивается посредством жесткой привязки физических адресов оперативной памяти к адресам кэш-памяти. Для этого сравнивают старшие разряды адреса, по которому выполняется обращение. Если номер страницы и тэг совпадут, то иско-

мая информация находится в кэш-памяти (это называют попаданием в кэш), в противном случае – в оперативной памяти - это кэш-промах.

Достоинствами кэш-памяти прямого отображения является высокая скорость определения наличия запрашиваемой информации и простота ее организации. Основной недостаток – это невысокая эффективность использования кэш-памяти, так как нельзя разместить одноименные строки (группы строк) различных страниц.

Алгоритм обратной записи WB быстродействующий, так как не требует при каждой записи обращаться к оперативной памяти. Запись информации в оперативную память производится только тогда, когда на место данной строки кэша вводится строка из другой страницы ОЗУ, или при выполнении команды обновления содержимого кэша. Этот алгоритм сложного управления, так как копии одной и той же информации различны в кэше и ОП.

Устранить неэффективное использование места в кэш-памяти прямого отображения можно, применив ассоциативный доступ, но он дорогой. Поэтому лучше сочетание механизма прямого отображения и ассоциативного поиска. Именно так и организован наборно-ассоциативный кэш, двухканальный (рис. 8.3.2).

Это сдвоенный кэш прямого отображения. Каждый банк кэш-памяти в паре со связанным с ним одним блоком тэговой памяти работает по схеме кэша прямого отображения. Однако наличие двух банков позволяет размещать в двухканальной наборно-ассоциативной кэш-памяти сразу две строки, расположенные одинаково по отношению к границам двух различных страниц кэшируемой памяти.

При обращении к памяти поиск нужной строки в кэше выполняется так же, как и в кэш-памяти прямого отображения, только этот поиск производится сразу для двух банков. Это повышает вероятность нахождения в кэше нужной информации и производительность памяти в целом. Могут быть четырех- и восьмиканальные наборно-ассоциативные кэш-памяти второго уровня.

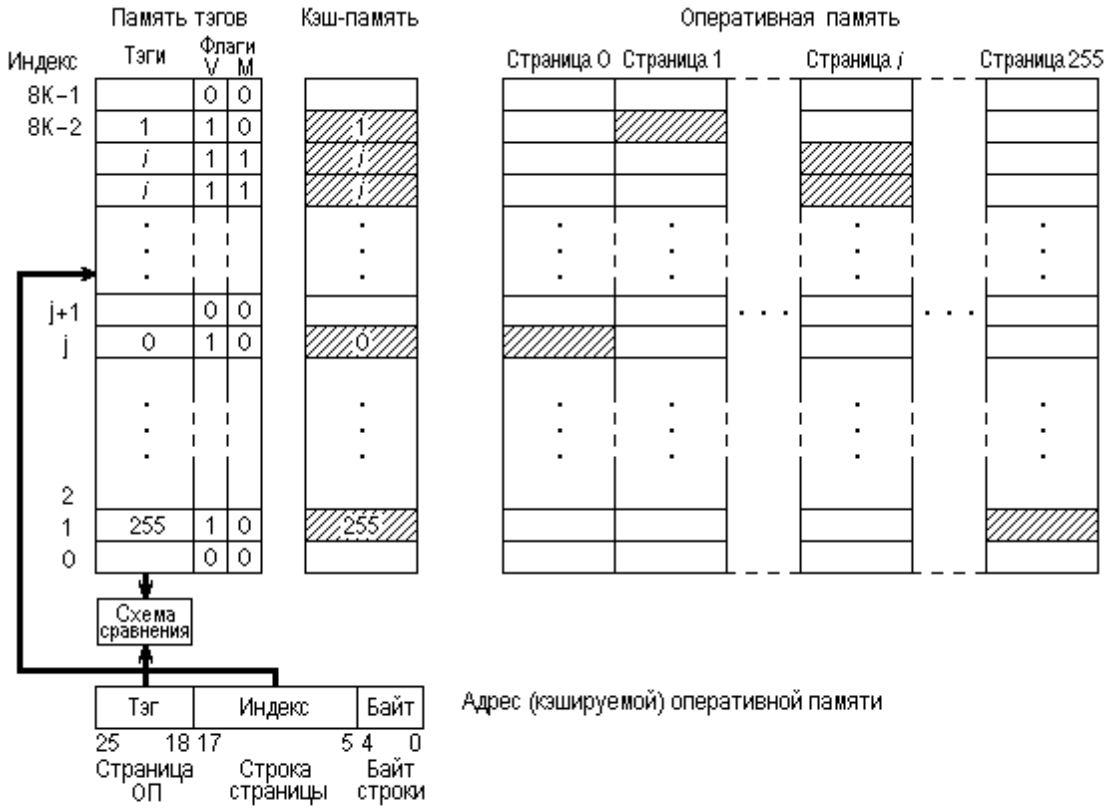


Рис. 8.3.2 – Двухканальный наборно-ассоциативный кэш

Обязательно нужно поддерживать соответствие (целостности, когерентности) между копиями одной и той же информацией в кэш- и оперативной памяти.

8.4 Оперативные запоминающие устройства (ОЗУ)

Оперативные запоминающие устройства выполнены на запоминающих элементах, содержащих биполярные или униполярные транзисторы, обеспечивающие высокое быстродействие и плотность размещения

Запоминающий элемент (триггер) на биполярных транзисторах с непосредственными связями представлен на рис. 8.4.1. В ЗЭ записан 0, если транзистор VT1 открыт, а транзистор VT2 закрыт. От $+E_k$ к $-E_д$ (к «земле») через открытый транзистор триггера ток может проходить через цепи эмиттеров. Верхние по схеме эмиттеры подключены к разрядным шинам, нижние соединены и выведены на адресную шину. При выборе ячейки, в которую входят рассматриваемые ЗУ, данная адресная шина возбуж-

дается – ее потенциал возрастает, и цепь тока через нижние эмиттеры обрывается.

В режиме хранения ЗЭ не выбраны, адресная шина имеет низкий потенциал, а разрядные – наиболее высокий. Поэтому ток открытого транзистора протекает через нижний эмиттер на адресную шину и по ней на «землю».

В режиме записи происходит выборка ЗЭ ячейки – данная адресная шина приобретает высокий потенциал, и ток отпертого транзистора ЗЭ может протекать только на разрядную шину. При появлении на входе двоичной переменной, записываемой в данный ЗЭ, и сигнала разрешения записи одна разрядная шина получает высокий потенциал, а другая – благодаря инвертору – низкий. Если перед этим триггер находился в состоянии 0 (VT1 отперт), то он переключается в состояние 1 (отперт VT2).

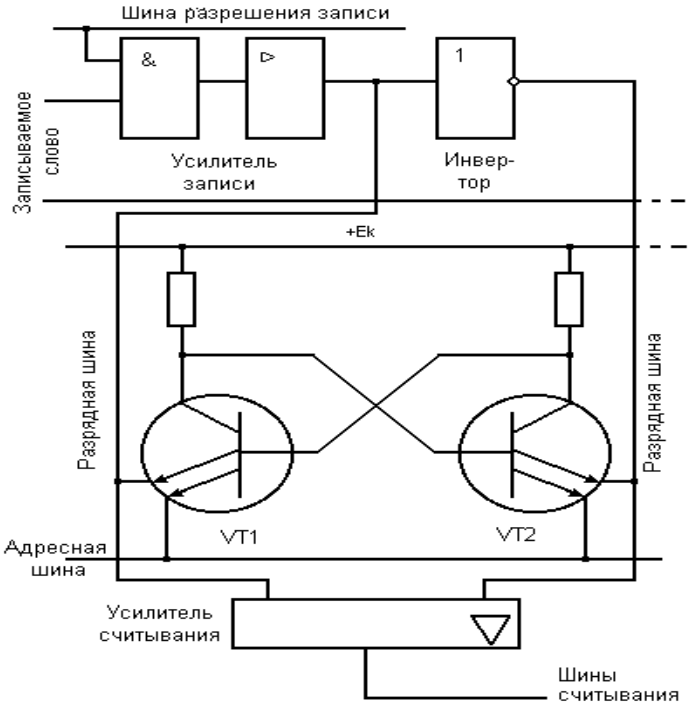


Рис. 8.4.1 – Запоминающий элемент (триггер) на биполярных транзисторах

В режиме считывания вновь происходит выборка ЗЭ ячейки – адресная шина получает высокий потенциал, и ток открытого транзистора протекает по разрядной шине на один из входов усилителя считывания. На его выходе появляется «лог. 1», если отперт VT2, и «лог. 0»,если отперт VT1.

В случае необходимости микросхемы ЗУ можно объединить, увеличивая тем самым емкость памяти. Для этого они имеют специальный вывод – «Выбор корпуса» (ВК).

В зависимости от типа ЗЭ на основе транзисторов реализуются ЗУ статические или динамические. В первом случае в качестве ЗЭ служит статический триггер, во втором случае информация запоминается на паразитной емкости затвора МОП-транзистора. Запоминающее устройство на МОП-транзисторах, так же как и на биполярных транзисторах, может быть с пословной и двухкоординатной произвольной выборкой.

ЗЭ-триггера на МОП-транзисторах для ЗУ с пословной выборкой приведен на рис. 8.4.2,*а*.

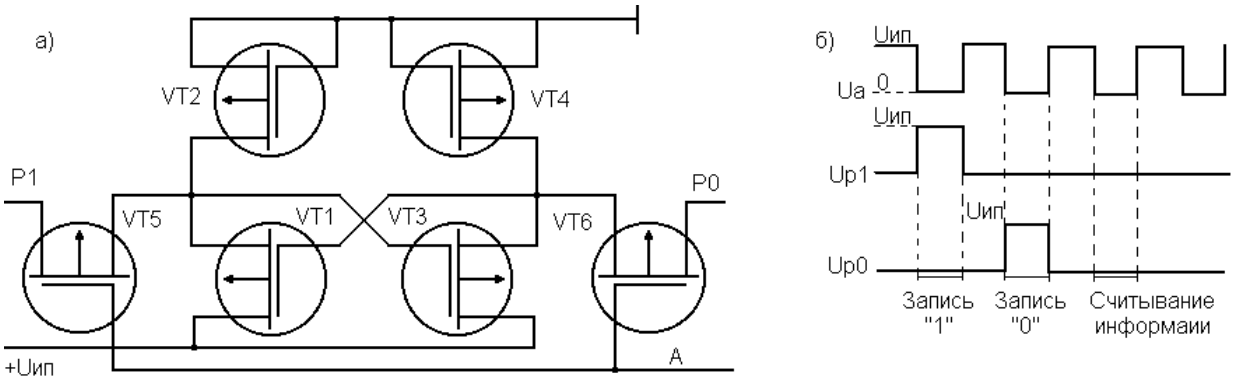


Рис. 8.4.2 – Запоминающий элемент – триггер на основе МОП структуры:
а – схема; *б* – временная диаграмма работы

Триггер образован транзисторами VT1-VT4. Управление триггером для записи и считывания осуществляется переключением транзисторов VT5 и VT6. Временные диаграммы работы такого ЗЭ представлены на рис. 8.4.2, б. В исходном состоянии напряжение на обоих разрядных шинах P_1 и P_0 равно нулю, а на шине слова А потенциал равен напряжению питания схемы. При этом транзисторы VT5 и VT6 закрыты, так как разность потенциалов между затворами и истоками по абсолютной величине меньше порогового напряжения. Триггер находится в одном из устойчивых состояний (0 или 1).

Высокое быстродействие полупроводниковых ЗУ обусловлено организацией с произвольным доступом. В статических ЗУ,

(*Static Random Access Memory* – SRAM) в качестве элемента памяти используется триггер. Это сложнее, чем конденсатор с транзисторным ключем динамического ЗУ. Поэтому статические ЗУ обладают меньшей плотностью хранения информации: В каждой ячейке памяти хранится один бит информации. На рис 8.4.3 показано, как может быть организован такой массив.

Память представляет матрицу запоминающих элементов, каждый из которых может быть установлен в одно из двух устойчивых состояний. Из ЗЭ строится матрица памяти, построение (организация) которой определяется способом выборки (опроса) ЗЭ при записи или считывании.

В структурной схеме матрицы с пословной выборкой и одной ступенью дешифрации (рис. 8.4.3, а) одна строка образует слово из m разрядов. На схеме символами A_1, A_2,…, A_n обозначены адресные, а P_1, P_2,…, Pm – разрядные шины. Адресные шины связаны с каждым ЗЭ одного слова, в то время как разрядные шины имеют связь с ЗЭ одноименного разряда всех слов. Состояние каждого из ЗЭ в этом слове может быть считано по разрядным шинам P_1 – P_m. Если необходимо записать информацию по выбранному адресу A_i, на разрядные шины P_1, P_2,…, P_m подаются электрические сигналы «1» и «0», которые попадут на каждый из ЗЭ i-ой строки: $ЗЭ_{i1}$, $ЗЭ_{i2}$, $ЗЭ_{i3}$,…, $ЗЭ_{im}$.

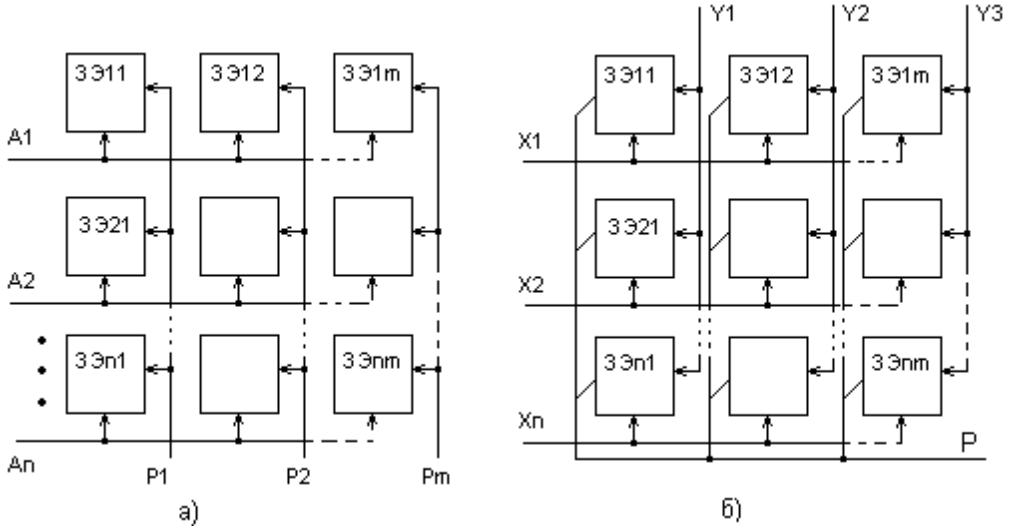


Рис. 8.4.3 – Структурные схемы матриц (накопителей информации):
а – с пословной выборкой и одной ступенью дешифрации;
б – двухкоординатная с двумя ступенями дешифрации

В структурной схеме двухкоординатной матрицы с двумя ступенями дешифрации (рис. 8.4.3,*б*) ЗЭ информация выбирается с помощью двух адресных шин. При наличии сигнала, соответствующего высокому уровню, на адресных шинах будет выбран только $ЗЭ_1$. Его состояние считывается по общей для всех элементов разрядной шине Р. Чтобы записать «1» в выбранный ЗЭ, по разрядной шине необходимо подать сигнал, соответствующий высокому уровню. Эта организация матрицы позволяет оперировать mn одноразрядными словами.

Основным недостатком статической памяти являются ее высокие стоимость и энергопотребление.

Основными разновидностями статической памяти (SRAM) с точки зрения организации ее функционирования являются асинхронная, синхронная пакетная (*Synchronous Burst*) и синхронная конвейерно-пакетная (*Pipeline Burst*) память.

Время доступа t_{AC} у типовых микросхем составляет порядка 10 нс, и реально такие микросхемы работают на частотах, близких к частоте системной шины.

Синхронная пакетная статическая память (**SBSRAM**) ориентирована на выполнение пакетного обмена информацией, который характерен для кэш-памяти. Эта память включает в себя внутренний счетчик адреса, предназначенный для перебора адресов пакета, и использует сигналы синхронизации ***CLK,*** как и синхронная DRAM память.

Пакетный цикл предусматривает передачу четырех элементов, так как внутренний счетчик имеет всего 2 бита, причем перебор адресов в пределах пакета может быть последовательным или с расслоением (чередованием) по банкам.

Конвейерно-пакетная память **PBSRAM** обеспечивает большее быстродействие, чем SBSRAM. В нее введены дополнительные внутренние буферные регистры данных адреса, а в ряде модификаций предусмотрена возможность передачи данных на двойной скорости по переднему и заднему фронтам синхросигнала и используются сдвоенные внутренние тракты записи и чтения. Время обращения порядка 2–3 нс, и передача данных пакета без задержек на частотах шины более 400 Мгц.

Динамические ЗУ с произвольным доступом (**DRAM**) часто используются, несмотря на недостатки, связанные с необходимостью регенерации информации в таких ЗУ и относительно невысоким их быстродействием, компенсируются другими показателями: малыми размерами элементов памяти, большим объемом микросхем этих ЗУ, низкой стоимостью.

ЗУ этого типа имеет разновидности: асинхронное, синхронное, RAMBUS и другие.

Асинхронными динамическими ОЗУ выполняют операции чтения и записи, получив лишь запускающий сигнал (обычно, сигнал строба адреса), независимо от каких-либо внешних синхронизирующих сигналов. Адрес на шины адреса поступает двумя частями: адрес строки (обозначенный как R_1 или R_2) и адрес столбца (C_1 и C_2).

До 90 процентов обращений процессора к памяти удовлетворяются кэш-памятью. Обращения, не выполненные кэшем, вызывают обмен информацией между ОЗУ и КЭШем, выполняемый блоками. Обмен информацией между оперативной памятью и внешними устройствами также выполняется блоками, и предполагает обращения по последовательным адресам.

Модификацией асинхронной динамической памяти стала память EDO (*Extended Data Output* – растянутый выход данных). В микросхеме EDO памяти на выходе установлен буфер-защелка, фиксирующий данные после их извлечения из матрицы памяти при подъеме сигнала строба и удерживающий их на выходе до следующего его спада. Это позволило сократить длительность сигнала строба и соответственно цикла памяти.

Затем появилась модификация асинхронной DRAM – BEDO (*Burst EDO* – пакетная EDO память), в которой не только адрес строки, но и адрес столбца подавался лишь в первом цикле пакета, а в последующих циклах адреса столбцов формировались с помощью внутреннего счетчика. Это позволило еще повысить производительность памяти.

Синхронная динамическая память обеспечивает большее быстродействие, чем асинхронная, при использовании аналогичных элементов памяти. Это позволяет реализовать пакетный цикл при частоте системной шины 100 МГц и выше.

Следующим шагом в развитии SDRAM стала память DDR SDRAM, обеспечивающая двойную скорость передачи данных (*DDR – Double* или *Dual Data Rate*), в которой за один такт осуществляются две передачи данных – по переднему и заднему фронтам каждого синхроимпульса. Во всем остальном эта память работает аналогично обычной SDRAM памяти (которую стали иногда называть SDR SDRAM – *Single Data Rate*). Времена задержек *CAS Latency* для DDR SDRAM могут быть 2 и 2,5 такта.

Дальнейшим развитием SDRAM является стандарт DDR2. В нем обеспечивается учетверенная скорость передачи данных по отношению к частоте работы самих элементов памяти.

Сокращение SIMM означает *Single In-Line Memory Module* – модуль памяти с одним рядом контактов, так как контакты краевого разъема модуля, расположенные в одинаковых позициях с двух сторон платы, электрически соединены. Соответственно DIMM значит *Dual In-Line Memory Module* – модуль памяти с двумя рядами контактов. RIMM означает *Rambus Memory Module* – модуль памяти типа Rambus.

Организация ОЗУ

ОЗУ компьютера представляет совокупность ЗЭ, охваченных общей схемой управления. Емкость ОЗУ можно наращивать без дополнительного проектирования БИС и без уменьшения быстродействия, т.к. время обращения к памяти будет практически таким же, как и время обращения к отдельным её блокам.

Основная память ЭВМ обычно организуется на основе микросхем памяти динамического типа (БИС ОЗУ типа DRAM), время обращения к которым в 3…5 раз больше, чем к ОЗУ статического типа (SRAM).

Следует отметить, что часть ОЗУ строится на базе БИС ПЗУ. Обычно для этих целей используются перепрограммируемые (репрограммируемые) микросхемы (БИС ППЗУ) с ультрафиолетовым или электрическим стиранием старой информации.

Основной особенностью динамических ОЗУ является разрушение информации при чтении и, следовательно, необходимость ее регенерации, что увеличивает время обращения при чтении и, следовательно, уменьшает быстродействие памяти.

Однако основной особенностью ОЗУ является её многоблочная организация. Связано это с тем, что ёмкость отдельных БИС ЗУ ограничена, поэтому для построения ОЗУ необходимого объёма приходится использовать несколько ЗУ, организованных в единое целое.

Структура многоблочной памяти представлена на рис. 8.4.4.

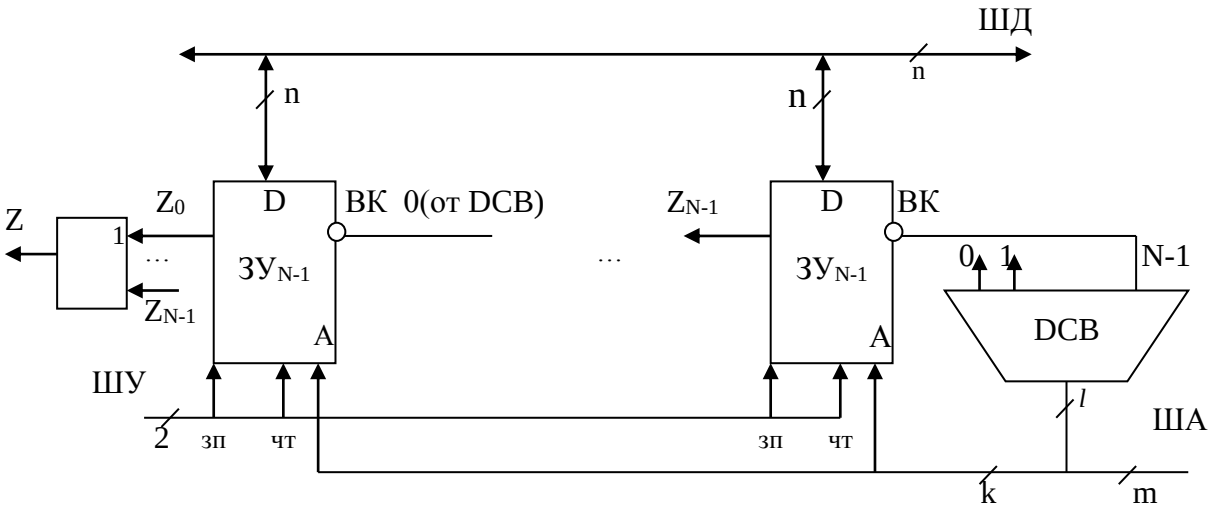


Рис. 8.4.4. Структура многоблочной памяти

$E_{ОЗУ}=2^m$, $E_{бл}=2^k$. Количество блоков 2^l, m=k+l.

Формат адреса: 1 m

1	B	l	1	D	k

Выбор блока обеспечивается дешифратором DCB.

Z – сигнал занятости памяти (блока ЗУ).

Функционирование многоблочной памяти осуществляется следующим образом:

В каждый момент времени схема обеспечивает обслуживание только одного обращения, т. к. одновременно в ней может быть выбран только один из блоков. Поскольку в каждом блоке есть свой автономный блок управления (с дешифратором адреса), то потенциально многоблочная память может обслуживать N обращений одновременно – за один цикл обращения – при условии, естественно, что каждый блок будет обслуживать свое обращение. В этом случае быстродействие многоблочной памяти может возрасти в N раз.

Такая память организуется через расслоение обращений. Обслуживание обращений к ОЗУ организуется, чтобы очередное по времени обращение к ней обслуживалось другим (свободным) блоком: первое обращение – i-м блоком, второе – j-м блоком, третье – k-м блоком и т.д. Необходимым условием такой организации является наличие в каждом блоке ЗУ двух буферных регистров – адреса РА и данных РД. Интерфейс памяти с расслоением обращений, конечно, будет сложнее, чем структура с одной шиной адреса и одной шиной данных,

8.5 Организация внешней памяти (ВЗУ)

Внешняя память ЭВМ – это сложная система, работающая под управлением операционной системы.

Внешняя память строится на основе ВЗУ различных типов: НМД, НМЛ, НОД и др. Особенности ВЗУ: ЗУ большой емкости, невысокого быстродействия и низкой удельной стоимости хранения информации. Простейшая структура внешней памяти представлена на рис. 8.5.1.

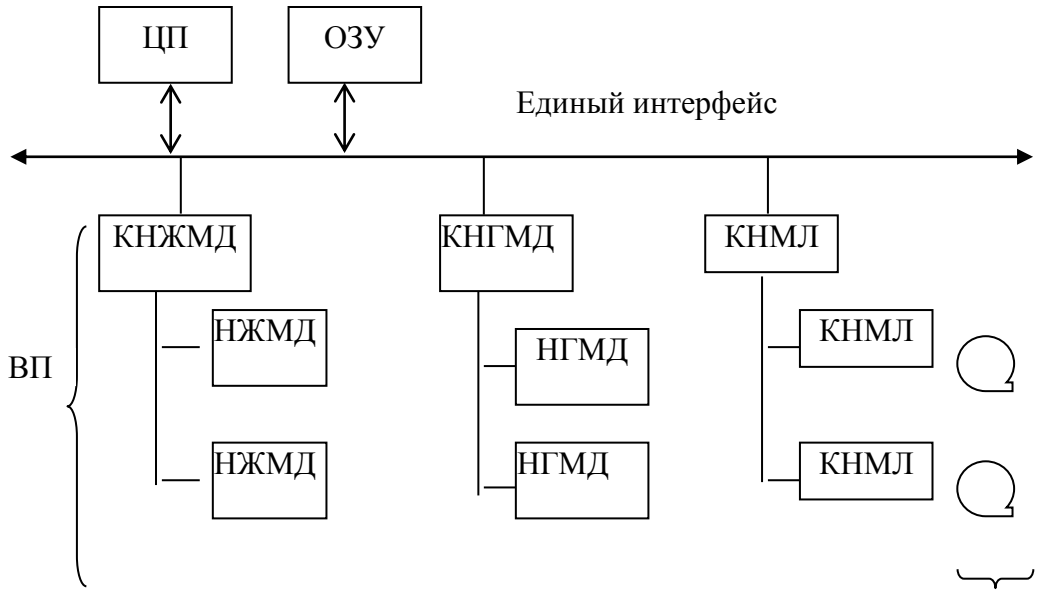


Рис. 8.5.1 – Структура внешней памяти

В состав ВЗУ входят:

1) носитель информации (съемный или стационарный);

2) привод – дисковод (для дисков), лентопротяжный механизм (ЛПМ) – для магнитных лент;

3) контроллер – блок управления накопителем.

Следует отметить, что НМД, НМЛ и др. обладают одним общим недостатком – движущимся носителем информации. Отсюда – низкое быстродействие (большое время доступа к информации) и низкая надежность.

По типу носителя информации ВЗУ делятся на следующие классы: МЛ – магнитная лента, МД – магнитный диск, ОД – оптический диск, МОД – магнитооптический диск.

Основные характеристики: емкость, время доступа к информации, пропускная способность (скорость чтения/записи), надежность, стоимость.

Емкость зависит от размеров носителя и плотности записи – количества информации на единицу площади носителя. Различают поперечную и продольную плотность записи информации на носителе.

ВЗУ на основе МД относятся к ЗУ с последовательным доступом к информации, а с логической точки зрения – к ЗУ с производительным (прямым) доступом информации.

ВЗУ на основе МД состоит из двух частей: собственно накопитель на МД (НМД) и блок управления накопителем – контроллер НМД (КНМД). Носитель состоит из привода МД (дисковода) и магнитного носителя информации в виде одного или нескольких (пакета) МД. Диск (пакет дисков) приводится в движение от специального привода, который обеспечивает их вращение с постоянной скоростью (рис. 8.5.2). Здесь П – привод, МП – механизм позиционирования, МГ – магнитная головка.

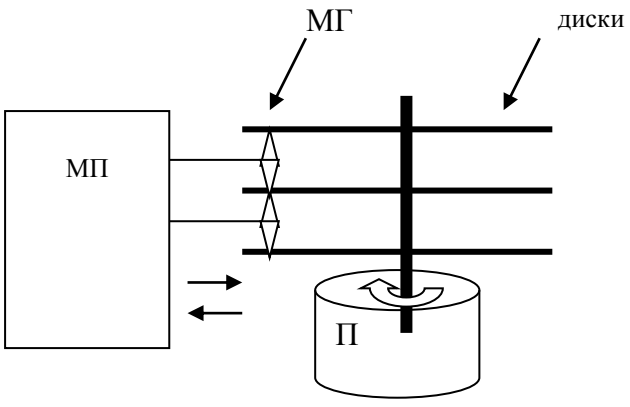


Рис. 8.5.2 – Организация НМД

Каждой рабочей поверхности диска соответствует одна МГ чтения/записи информации. Запись информации на носитель осуществляется при помощи МГ записи, чтение информации – при помощи МГ чтения.

При записи информации на носитель при помощи МГ создаются магнитные отпечатки, следы. Магнитные отпечатки, оставляемые на поверхности диска при фиксированном положении МГ, образуют дорожку – трек (рис. 8.5.4).

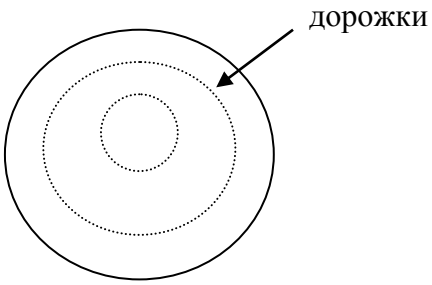


Рис. 8.5.4 – Вид диска

Перемещение блока МГ с одной позиции (дорожки) на другую вдоль радиуса диска обеспечивается механизмом позиционирования. В результате на одну поверхность диска помещается несколько дорожек – от нескольких десятков до нескольких сотен и даже тысяч в зависимости от радиуса диска и поперечной плотности записи.

Дорожки на поверхности нумеруются: 1, 2, …, C. Множество дорожек с одинаковыми номерами, расположенными на различных поверхностях пакета дисков, образует цилиндр, количество дорожек в котором определяется количеством рабочих поверхностей в пакете дисков.

Обмен информацией (чтение/запись) с дисками осуществляется блоками фиксированной длины – секторами. Количество секторов на дорожке зависит от емкости сектора и плотности записи. Размер сектора обычно 512 В.

В результате НМД можно представить в виде геометрической модели – совокупности вложенных цилиндров (рис. 8.5.5). Здесь C – номер цилиндра, H – номер поверхности, R – номер блока (сектора) на дорожке. Поскольку обмен информацией с НМД осуществляется с помощью одной МГ (чтения или записи),

т.е. в последовательном коде, то в цилиндре с номером С работает только одна поверхность с номером Н, обращение к которой обеспечивается МГ с номером Н.

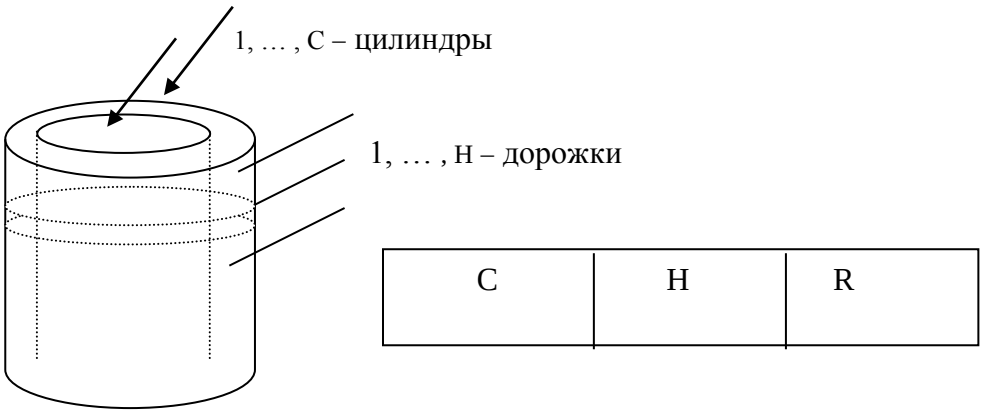


Рис. 8.5.5 – Организация носителя на МД

Доступ к цилиндру с номером С обеспечивается механизмом позиционирования. Время доступа к цилиндру зависит от быстродействия механизма позиционирования и разности $\Delta C = C - C'$, где С – номер цилиндра, к которому производится обращение, С’ – номер цилиндра, к которому производилось последнее обращение: $\tau_{поз} = \Delta C \tau_{поз}$, где $\tau_{поз}$ – время позиционирования на соседнюю дорожку. Знак ΔС задаёт направление движения МП.

Доступ к дорожке Н в цилиндре С осуществляется путём подключения (коммутации) МГ к единственному усилителю чтения-записи. Время коммутации незначительно, поскольку осуществляется электронным коммутатором.

Таким образом, дорожка, к которой производится обращение, задаётся парой значений: С, Н.

Доступ к сектору R на дорожке С, Н обеспечивается (неподвижной) МГ и сводится к ожиданию момента, когда адресуемый сектор R окажется под МГ. Время ожидания $\tau_{ож}$ – случайная величина, лежит в пределах от 0 до $T = 1/\omega$, ω – угловая скорость вращения диска.

Общее время доступа: $\tau_д = \tau_{поз} + \tau_{ком} + \tau_{ож}$.

Доступ к дорожке С – последовательный апериодический, к поверхности Н – произвольный (прямой), к сектору R – последовательный периодический (циклический).

С логической точки зрения НМД относятся к ЗУ с произвольным доступом к файлам. Прямой доступ к файлам на дисках обеспечивается каталогом файлов, хранимых на дисках. Каталог файлов устанавливает соответствие имён файлов и их физических адресов: C, H, R, т.е. каталог – это таблица, в которой и указывается это соответствие.

Накопители на магнитной ленте получили новое развитие в связи с появлением способа потоковой записи (stream – поток). Отсюда название такого рода НМЛ – стример.

Основная особенность стримеров: вся поверхность МЛ делится на дорожки. Количество дорожек k от 20 и выше.

Запись информации осуществляется сначала на первую дорожку непрерывно (в последовательном коде, потоком) при помощи неподвижной (стационарной) МГ с номером 1. Когда дорожка кончается, осуществляется реверс МЛ и запись продолжается на вторую дорожку и т.д. «серпантином», «змейкой». Недостатки стримеров – плохая совместимость, которая объясняется отсутствием стандартов записи. Пропускная способность стримеров от 2 до 18 МВ/с. Область применения – хранение архивов.

В ЗУ на основе оптических методов используется способность некоторых материалов изменять свойства отражения света на тех участках ОД, которые подверглись, например, тепловому воздействию от лазерного источника света.

Один из первых способов оптической записи основан на способности лазерного луча прожигать отверстия в тонком слое металла. Прожженное отверстие является оптическим отпечатком записываемой информации. Считывание информации производится также лазерным лучом, но меньшей, чем при записи, интенсивности, и фотодетектором ФД (рис.8.5.6).

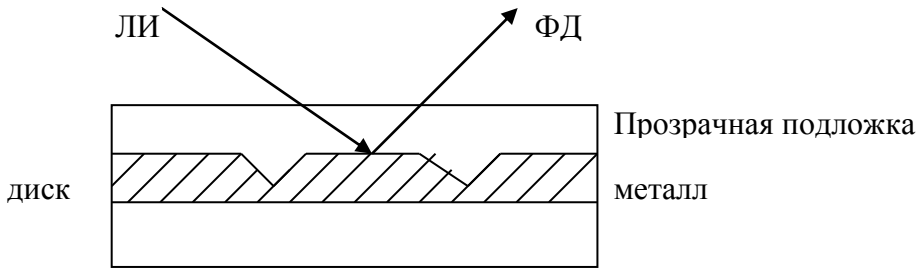


Рис.8.5.6

Отраженный от поверхности диска луч (от лазерного источника ЛИ) попадает на фотодетектор ФД и преобразуется в электрический сигнал, который интерпретируется либо как 0, либо как 1 – в зависимости от интенсивности отраженного сигнала.

Особенности ОД: 1) информационные дорожки в виде спирали, т.е. дорожка фактически одна, 2) переменная угловая скорость вращения диска, 3) высокая надежность хранения – десятки лет, 4) высокая плотность записи – в десятки раз выше, чем у ЖМД. И, как следствие, низкая удельная стоимость хранения информации. Основной недостаток ОД – большое время доступа: измеряется десятками, сотнями мс, может достигать секунд.

8.6 Виртуальная память

В современных компьютерах возможности формирования адресного пространства опережают объем физической памяти. Поэтому часть сегментов программы, которые редко используются, хранятся во внешней памяти и по мере необходимости переписываются в ОЗУ. Эту технологию принято называть виртуальной памятью. За ее организацию отвечает ОС. Программы в ходе выполнения используют логические или виртуальные адреса. Затем эти адреса преобразуются в физические. Если виртуальный адрес указывает на данные и программы, находящиеся в не основной памяти, то этот сегмент должен быть перемещен в основную память (рис. 8.6.1).

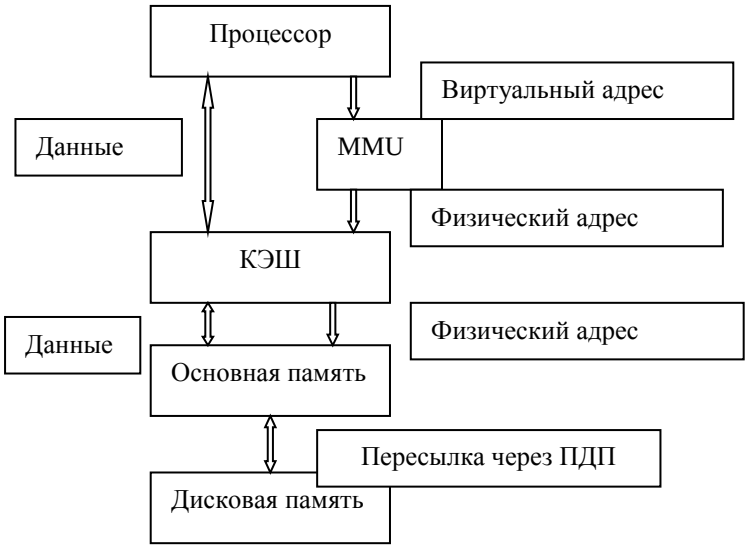


Рис. 8.6.1 – Организация виртуальной памяти

8.7 Постоянные запоминающие устройства

Постоянные запоминающие устройства (ПЗУ или Read Only Memory – ROM), называемые энергонезависимыми (или Non Volatile Storage), обеспечивают сохранение записанной в них информации и при отсутствии напряжения питания. Постоянные ЗУ – это устройства памяти с произвольным адресным доступом, основанные на различных физических принципах и обладающие различными характеристиками по емкости и времени обращения к ним, возможности замены записанной в них информации.

Процесс записи для полупроводниковых постоянных ЗУ получил также название «прожига» или программирования, первое из которых связано со способом записи, сводящимся к разрушению (расплавлению, прожигу) соединительных перемычек в чистом ЗУ.

В полупроводниковых ПЗУ в качестве элементов памяти используются транзисторы, объединенные в матрицу, выборка данных из которой производится по строкам и столбцам, соответствующим указанному адресу, так же, как и в других ЗУ с произвольным доступом.

Различают две большие группы ПЗУ: программируемые изготовителем *и* программируемые пользователем.

ЗУ первой группы, называемые иначе масочными, обычно выпускаются большими партиями. Информация в них заносится в процессе изготовления этих ЗУ с помощью специальной маски в конце технологического цикла, и на кристалле формируется соответствующая конфигурация соединений. Такие ЗУ оказываются наиболее дешевыми при массовом изготовлении.

В ПЗУ, программируемые пользователем, информация записывается самими пользователями. При этом существуют два основных типа таких ЗУ: однократно программируемые *и* перепрограммируемые.

Наиболее простыми являются однократно программируемые ПЗУ. В этих ЗУ запись производится посредством разрушения соединительных перемычек между выводами транзисторов и шинами матрицы.

Перепрограммируемые ПЗУ позволяют производить в них запись информации многократно. Распространенные технологические варианты используют МОП-транзисторы со сложным затвором (составным или «плавающим»), который способен накапливать заряд, снижающий пороговое напряжение отпирания транзистора, и сохранять этот заряд при выключенном питании. Программирование таких ПЗУ и состоит в создании зарядов на затворах тех транзисторов, где должны быть записаны данные (обычно «0», так как в исходном состоянии в таких микросхемах записаны все «1»).

Флэш-память, появившаяся в конце 1980-х годов (*Intel*), является представителем класса перепрограммируемых постоянных ЗУ с электрическим стиранием. Однако стирание в ней осуществляется сразу целой области ячеек: блока или всей микросхемы. Флэш-память строится на однотранзисторных элементах памяти (с «плавающим» затвором), что обеспечивает плотность хранения информации даже несколько выше, чем в динамической оперативной памяти. Наиболее широко известны NOR и NAND типы флэш-памяти, запоминающие транзисторы в которых подключены к разрядным шинам, соответственно, параллельно и последовательно.

Первый тип имеет относительно большие размеры ячеек и быстрый произвольный доступ (порядка 70 нс), что позволяет выполнять программы непосредственно из этой памяти.

Второй тип имеет меньшие размеры ячеек и быстрый последовательный доступ (обеспечивая скорость передачи до 16 Мбайт/с), что более пригодно для построения устройств блочного типа, например «твердотельных дисков».

Элементы памяти флэш-ЗУ организованы в матрицы, как и в других видах полупроводниковой памяти. Операция чтения из флэш-памяти выполняется как в обычных ЗУ с произвольным доступом (оперативных ЗУ или кэш). Однако запись сохраняет в себе некоторые особенности, аналогичные свойствам постоянных ЗУ.

В процессе записи информации соответствующие элементы памяти переключаются в нулевое состояние. Фактически при операции записи производятся два действия: запись и считыва-

ние, но управление этими операциями производится внутренним автоматом и «прозрачно» для процессора.

Флэш-память используется для различных целей. Непосредственно в самой ЭВМ эту память применяют для хранения BIOS (базовой системы ввода-вывода), что позволяет при необходимости производить обновление последней прямо на рабочей машине.

Другим применением флэш-памяти являются «твердотельные диски», эмулирующие работу внешних винчестеров. Такое устройство имеет габариты порядка 70×20×10 мм, подключается обычно к шине USB и состоит из собственно флэш-памяти, эмулятора контроллера дисковода и контроллера шины USB. При включении его в систему (допускается «горячее» подключение и отключение) устройство с точки зрения пользователя ведет себя как обычный (съемный) жесткий диск. Производительность его меньше, чем у жесткого диска: скорость передачи при записи и чтении составляет менее одного мегабайта в секунду.

8.8. RAID-массивы дисков

RAID (Redundant Array of Independent Disks. RAID-массив представляет объединение физических HDD дисков в один логический. Создание RAID-массивов необходимо для повышения скорости чтения и записи данных, для повышения безопасности и отказоустойчивости вычислительной системы. RAID-массивы бывают двух типов, аппаратные и программные:

- аппаратные RAID-массивы создаются до того, как производится загрузка операционной системы при помощи специализированных утилит. После такой обработки, при подключении RAID-массива операционная система на стадии инсталляции видит HDD диски как один.
- программные RAID-массивы создаются посредством подключения HDD дисков к операционной системе, которая определяет несколько физических дисков, и затем HDD диски объединяются в один массив. ОС устанавливается до создания массива.

Виды RAID-массивов.

RAID-0. Более одного HDD диска объединяются в один посредством последовательного соединения, после чего происходит суммирование объемов. После слияние дисков в один массив, скорость чтения и записи у накопителя будет больше, чем у дис-

ков по раздельности. Массив «RAID-0» предоставит возможность выполнять чтение и запись параллельно. В массиве RAID-0 отсутствует отказоустойчивость.

Массив «RAID-1» не увеличит производительность, зато отказоустойчивость обеспечивает, в случае если выйдет из строя один из HDD дисков, на втором HDD диске будет резервная копия информации. В случае удаления данных с массива целенаправленно, то удаление происходит с обоих дисков одновременно.

Самый отказоустойчивый массив RAID-5 Рис.8.8.1. Заполнение массива информацией описывается выражением «(N - 1) * V, где N число - это количество HDD дисков находящихся в массиве, а V - это объем каждого установленного HDD диска, т.е. при создании массива версии «RAID-5» из 3-х HDD дисков, емкостью каждый из которых по 500Гб, у нас получится массив объемом памяти в 1000Gb 1терабайт.

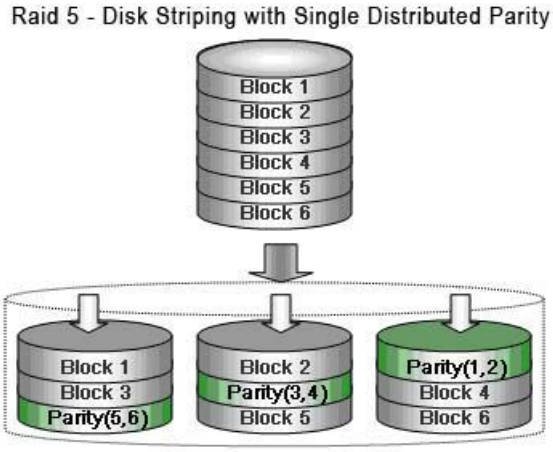


Рис. 8.8.1 – массив дисков RAID-5

Суть RAID-5 массива заключается в следующем - несколько HDD дисков объединяются в RAID-0, а на третьем HDD диске (который не учитывается) будет храниться "контрольная сумма" в виде информации, предназначенной для восстановления дисков массива, в случае отказа. У массива RAID-5 скорость записи ниже из-за небольших затрат времени на расчет и запись полученной суммы на дополнительный диск, а скорость чтения остается такой же, как у массив RAID-0. Если один из HDD дисков массива RAID-5 выйдет из строя, сразу резко понизится скорость чтения и записи, так как все происходящие операции сопровождаются дополнительным действиями.

Параллельно с массивом RAID-5 есть возможность использовать запасной диск (Spaire). Во время стабильного режима работы RAID-массива, диск Spare не используется и находится в режиме простоя. В случае возникновения критической ситуации, резервное восстановление RAID-массива начнется в автоматическом режиме. На запасной HDD диск будет происходить восстановление информации с поврежденного HDD при помощи контрольно-вычислительных сумм, нахождение которых находится на Spaire диске. Массив RAID-5 обычно создается минимум из трех HDD дисков и поможет спасти данные только от одиночных возникших ошибок.

RAID-6 обладает улучшенными возможностями по сравнению с RAID-5 массивом. Работа такая же как с массивом RAID-5, только вычисление контрольных сумм будет происходить не на один HDD диск, а на два HDD диска, причем весь подсчет контрольных сумм выполняются разными алгоритмами, что способствует существенному повышению отказоустойчивости всего RAID-массива в целом.

Массив RAID-10 представляет объединение RAID-0 и RAID-1. Данный массив обычно создается минимум из четырех HDD дисков. На первом разделе RAID-0 и на втором RAID-0 для того, чтобы повысить скорость чтения и записи, между собой они будут находится в зеркале массива RAID-1, это нужно для повышения отказоустойчивости. Массив RAID-10 смог совместить в себе плюсы двух первых вариантов - что последовало его повышению быстродействия и отказоустойчивости.

Массив RAID-50 является объединением RAID-0 и RAID-5 и собирается как массив RAID-5, и состоит из массивов RAID-0. Массив RAID-50 обеспечивает при работе высокую скорость чтения с записью и устойчивость и надежность.

Существуют и другие комбинации массивов, таких как - RAID 5+1 / RAID 6+1 - по сути, они схожи с RAID-50 / RAID-60 с той разницей, что базой их элементов массива обладают не «RAID-0» как у других, а зеркала массива «RAID-1».

Контрольные вопросы

1. В чем достоинства и недостатки запоминающих элементов на биполярных транзисторах и униполярных?

2. Почему не используется КЭШ-память только с ассоциативной организацией и какая организация предпочтительней?

3. Чем обусловлена иерархия памяти в компьютерах?

4. В чем отличие модификации TLB в виртуальной памяти и в КЭШ-памяти?

5. К какому виду памяти можно отнести видеопамять?

6. В чем суть пакетной передачи в оперативной памяти?

Глава 9. МНОГОПРОЦЕССОРНЫЕ СИСТЕМЫ

9.1 Принципы многопроцессорной обработки

Достоинство многопроцессорной вычислительной системы заключается в ее производительности, т.е. количество операций, выполняемой системой за единицу времени, а также в архитектурных решениях, направленных на повышение производительности (работа с векторными операциями, организация быстрого обмена сообщениями между процессорами или организация глобальной памяти в многопроцессорных системах и др.).

В 1966 году **М. Флинн** разработал классификацию архитектур вычислительных систем, в основу которой положено понятие потока. Под потоком понимается последовательность элементов, команд или данных, обрабатываемая процессором (рис. 9.1.1).

SISD (single instruction stream / single data stream) – одиночный поток команд и одиночный поток данных. К этому классу относятся последовательные компьютерные системы, которые имеют один центральный процессор, способный обрабатывать только один поток последовательно исполняемых инструкций (классическая машина фон Неймана).

MISD (multiple instruction stream / single data stream) – множественный поток команд и одиночный поток данных. Множество инструкций выполняется над единственным потоком данных. Пока таких компьютеров не создано.

SIMD (single instruction stream / multiple data stream) – одиночный поток команд и множественный поток данных. Эти си-

стемы имеют множество процессоров, обладающих собственной памятью, в пределах от 1024 до 16384 шт., выполняющих одну и ту же инструкцию относительно разных данных.

MIMD (multiple instruction stream / multiple data stream) – множественный поток команд и множественный поток данных. Такие системы параллельно выполняют несколько потоков инструкций над различными потоками данных.

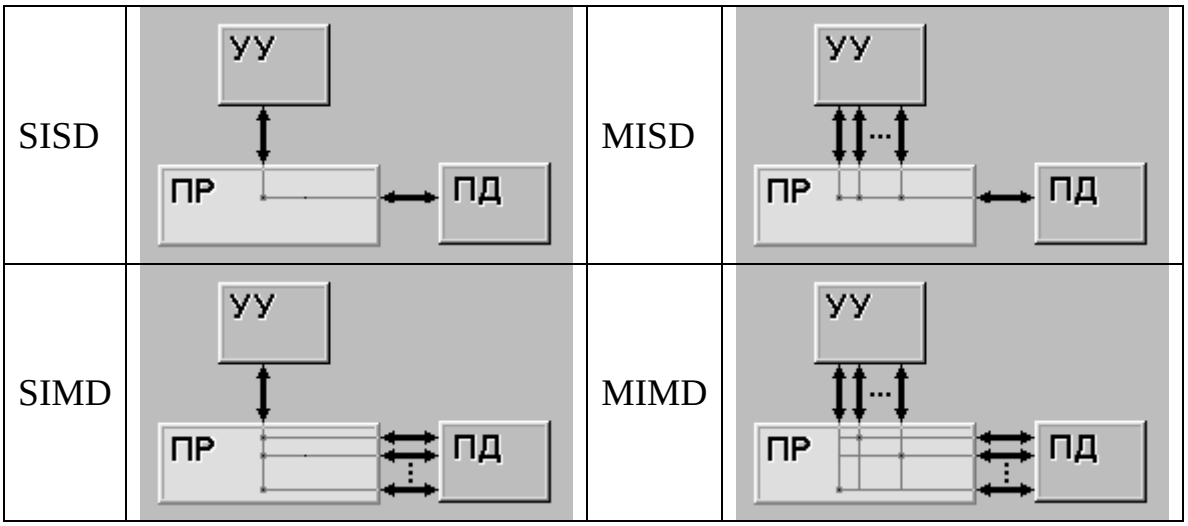


Рис. 9.1.1 – Классификация Флинна

Современные многопроцессорные систем в основном принадлежат одному классу MIMD, и параметром классификации многопроцессорных систем является наличие **общей** (SMP-архитектура) или **распределенной** (MMP-архитектура) памяти. Нечто среднее между этими классами представляют собой NUMA-архитектуры, где память физически распределена, но логически общедоступна, Рис. 9.1.3.

- **Многопроцессорные системы** с общей UMA (разделяемой) памятью, где каждый процессор компьютера обладает возможностью прямого доступа к общей памяти, используя общую шину (возможно, реализованную на основе высокоскоростной сети). В таких компьютерах сложно увеличивать число процессоров, поскольку при этом возрастает числа конфликтов доступа к шине.

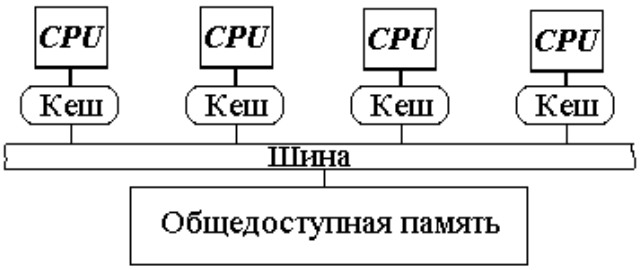


Рис. 9.1.2 – Общая память

- Многопроцессорные системы с **распределенной** памятью, где каждый процессор имеет доступ только к локальной собственной памяти. Процессоры объединены в сеть. Доступ к удаленной памяти возможен только с помощью системы обмена сообщениями.

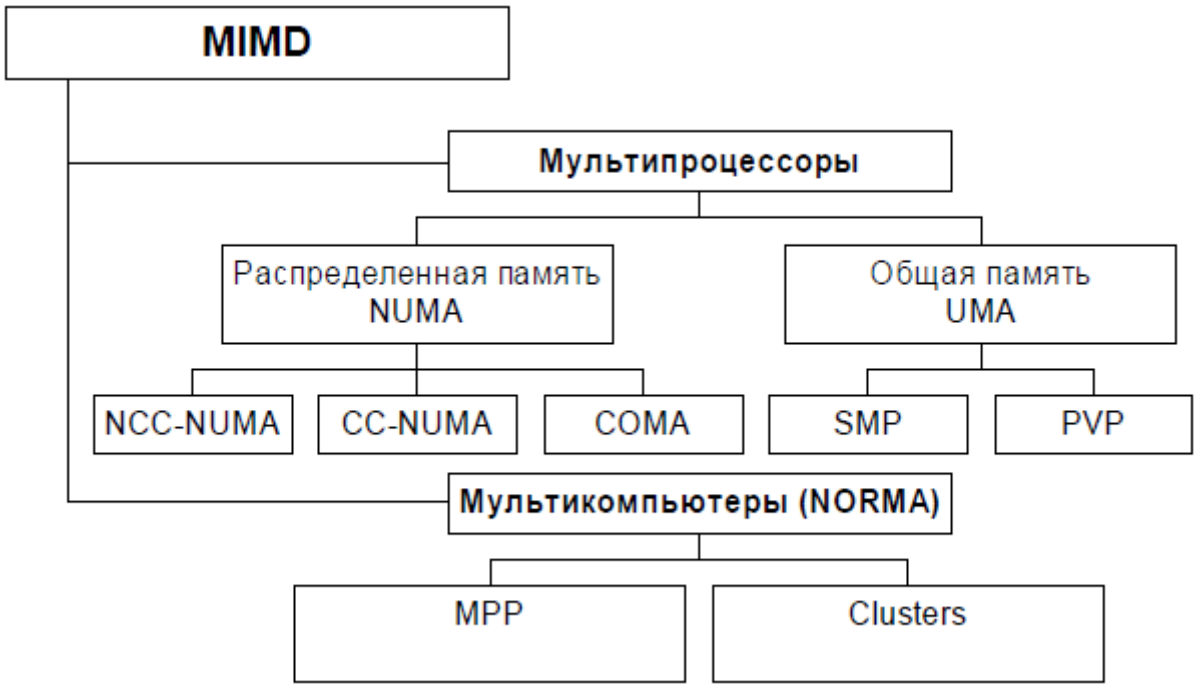


Рис.9.1.3. Классификация многопроцессорных вычислительных систем

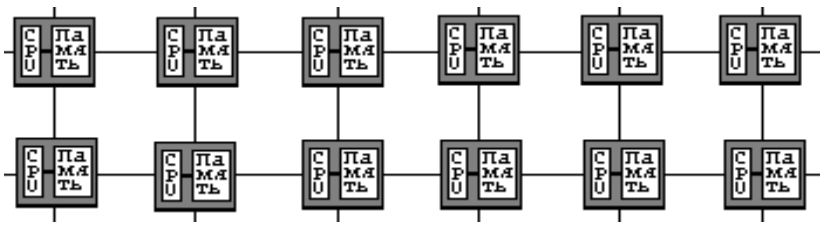


Рис. 9.1.4 – Распределенная память

- **Неоднородный доступ** к памяти **– NUMA –** (non uniform memory access).

Таких архитектур много. В некоторых архитектурах каждый процессор имеет как прямой доступ к общей памяти, так и собственную локальную память.

Рис. 9.1.5. Система с неоднородной памятью

Сеть обмена, с помощью которой процессоры соединяются друг с другом или с памятью, является важным элементом многопроцессорной вычислительной системы. Способы межпроцессорного обмена связаны с архитектурой оперативной памяти системы.

Существуют две основные модели межпроцессорного обмена:

- с использованием общей памяти.
- с передачей сообщений,

В многопроцессорной системе с общей памятью один процессор осуществляет запись в конкретную ячейку, а другой процессор производит считывание из этой ячейки памяти. Чтобы обеспечить согласованность данных и синхронизацию процессов, обмен часто реализуется по принципу взаимоисключающего доступа к общей памяти.

9.2 Организация многопроцессорных систем

Многопроцессорная система ILLIAC IV (Иллинойский университет) включала 64 (и до 256) процессорных элемента (ПЭ), работающих по единой программе, применяемой к локальной оперативной памяти каждого ПЭ. Обмен данными между процессорами осуществлялся через матрицу коммуникационных каналов. Это многопроцессорная системой с массовым параллелизмом (MPP – Massively Parallel Processing). Каждый из процессорных элементов

MPP системы является универсальным процессором, функционирующим по собственной программе

Затем мультипроцессорные системы появились на базе векторно-конвейерных компьютеров (фирмы Cray). Симметричные мультипроцессорные системы (Symmetric Multi-Processing – SMP) объединяли от 2 до 16 процессоров, имеющих равноправный (симметричный) доступ к общей оперативной памяти. Ограничения из-за конфликтов на шине, привели к необходимости обеспечить каждый процессор собственной оперативной памятью, превращая компьютер в объединение независимых вычислительных узлов. Это увеличивает степень масштабируемости многопроцессорных систем, но требует способа обмена данными между вычислительными узлами передачи сообщений.. Компьютеры с такой архитектурой (MPP системы) перспективны в развитии суперкомпьютерных технологий.

Два принципа заложены в архитектуре векторно-конвейерных процессоров:

- конвейерная организация обработки потока команд,
- введение в систему команд набора векторных операций, позволяющих оперировать с целыми массивами данных.

Конвейерная обработка эффективна при загрузке конвейера, близкой к полной, а скорость подачи новых операндов соответствует максимальной производительности конвейера.

Главный принцип вычислений на векторной машине заключается в выполнении элементарной операции или комбинации из элементарных операций, которые должны повторно применяться к некоторому блоку данных. Таким операциям в исходной программе соответствуют небольшие компактные циклы. Длина одновременно обрабатываемых векторов в современных векторных компьютерах составляет, как правило, 128 или 256 элементов, поэтому векторные процессоры имеют сложную структуру и содержать множество арифметических устройств. Основное назначение векторных операций состоит в распараллеливании выполнения операторов цикла, в которых сосредоточена большая часть вычислительной работы. Для этого циклы подвергаются процедуре векторизации для реализации векторных команд. Это выполняется автоматически компиляторами при выполнении ис-

полнимого кода программы. Исходя из этого векторно-конвейерные компьютеры не требуют специальной технологии программирования. Несколько векторно-конвейерных процессоров (2–16) работают в режиме с общей памятью (SMP), образуя вычислительный узел, а несколько таких узлов объединяются с помощью коммутаторов, образуя либо NUMA-, либо MPP-систему.

SMP архитектура (symmetric multiprocessing) – симметричная многопроцессорная архитектура с общей физической памятью, разделяемой всеми процессорами.

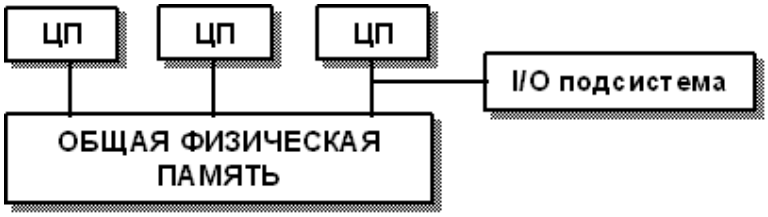


Рис. 9.2.1 – Схематический вид SMP-архитектуры

Память выполняет передачу сообщений между процессорами, при этом все вычислительные устройства при обращении к ней имеют равные права и одну и ту же адресацию для всех ячеек памяти. Поэтому SMP-архитектура называется симметричной. Последнее обстоятельство позволяет эффективно обмениваться данными с другими вычислительными устройствами. SMP-система строится на основе высокоскоростной системной шины, к слотам которой подключаются функциональные блоки трех типов: процессоры (ЦП), операционная система (ОС) и подсистема ввода/вывода (I/O). Для подсоединения к модулям I/O используются медленные шины (PCI, VME64).

Данные, находящиеся в кэш-памяти любого процессора, недоступны для других процессоров. Поэтому после каждой модификации переменной, находящейся в кэш-памяти процессора, необходимо производить синхронную модификацию этой же переменной, расположенной в основной памяти.

Основные преимущества SMP-систем:

- Простое программирование. Использование общей памяти увеличивает скорость обмена, пользователь имеет доступ ко всему объему памяти. Для SMP-систем существуют эффективные средства автоматического распараллеливания;

• легкость в эксплуатации. SMP-системы используют систему охлаждения, основанную на воздушном кондиционировании, облегчающую их техническое обслуживание;

Недостатки:

• системы с общей памятью, построенные на системной шине, плохо масштабируемы, потому что шина способна обрабатывать только одну транзакцию, вследствие чего возникают проблемы разрешения конфликтов при одновременном обращении нескольких процессоров к одним и тем же областям общей физической памяти.

• системная шина имеет ограниченную (хоть и высокую) пропускную способность (ПС) и ограниченное число слотов. Все это с очевидностью препятствует увеличению производительности при увеличении числа процессоров и числа подключаемых пользователей. В реальных системах можно использовать не более 32 процессоров.

Для построения масштабируемых систем на базе SMP используются кластерные или NUMA-архитектуры. При работе с SMP-системами используют так называемую парадигму программирования с разделяемой памятью (shared memory paradigm). Представители этого класса: HP 9000 (Exemplar), Sun HPC 10000 (StarFire), Sun Fire 15K, Fujitsu PrimePower 2000, AlphaServer GS/ES.

MPP-архитектура (massive parallel processing) – массивно-параллельная архитектура с массовым параллелизмом с физически разделенной памятью. Система строится из отдельных модулей, содержащих один или несколько процессоров, локальный банк операционной памяти (ОП), два коммуникационных процессора, жесткие диски и/или другие устройства ввода/вывода.

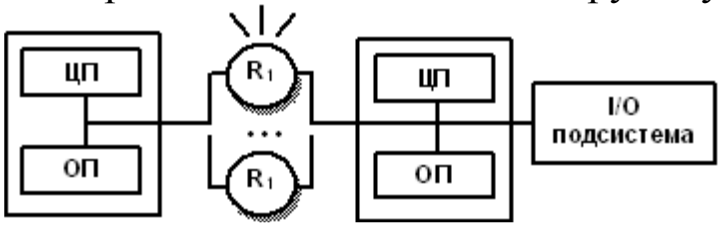


Рис. 9.2.2 – Схема модуля MPP-системы

Один маршрутизатор используется для передачи команд, другой – для передачи данных. Доступ к банку ОП из данного модуля имеют только процессоры (ЦП) из этого же модуля. Оперативная память в MPP-системах имеет 3-уровневую структуру:

- кэш-память процессора;
- локальная оперативная память узла;
- оперативная память других узлов.

При этом на каждом узле может функционировать либо полноценная операционная система, либо урезанный вариант, поддерживающий только базовые функции ядра. Модули соединяются коммуникационными каналами через высокоскоростную сеть.

Процессоры в таких системах имеют прямой доступ только к своей локальной памяти. Доступ к памяти других узлов реализуется обычно с помощью механизма передачи сообщений. Такая архитектура вычислительной системы устраняет одновременно как проблему конфликтов при обращении к памяти, так и проблему когерентности кэш-памяти. Это дает возможность практически неограниченного наращивания числа процессоров в системе, увеличивая тем самым ее производительность. Успешно функционируют MPP-системы с сотнями и тысячами процессоров. Производительность наиболее мощных систем достигает 10 триллионов оп/сек (10 Tflops). Важным свойством MPP систем является их высокая степень масштабируемости, для достижения нужной производительности требуется просто собрать систему с определенным числом узлов.

Достоинство:

- масштабируемость: каждый процессор имеет доступ только к своей локальной памяти, в связи с чем не возникает необходимости в потактовой синхронизации процессоров и производительности.

Недостатки:

- отсутствие общей памяти снижает скорость межпроцессорного обмена, поскольку нет общей среды для хранения данных.
- каждый процессор использует ограниченный объем локального банка памяти;

• требуются значительные усилия для использования системных ресурсов.

Представители этого класса: IBM RS/6000 SP, Cray T3E, AlphaServer SC, Hitachi SR8000, системы Parsytec.

Гибридная архитектура (NUMA)

Архитектура NUMA является MPP-архитектурой, где в качестве отдельных вычислительных элементов используются SMP-узлы. Неоднородный доступ к памяти имеет достоинства с общей памятью и простоту систем с раздельной памятью. Память является физически распределенной по различным частям системы, но логически разделяемой, и пользователь видит единое адресное пространство. Система состоит из однородных базовых модулей, состоящих из небольшого числа процессоров и блока памяти. Модули объединены с помощью высокоскоростного коммутатора. Поддерживается единое адресное пространство, аппаратно поддерживается доступ к удаленной памяти, т.е. к памяти других модулей. При этом доступ к локальной памяти осуществляется в несколько раз быстрее, чем к удаленной..

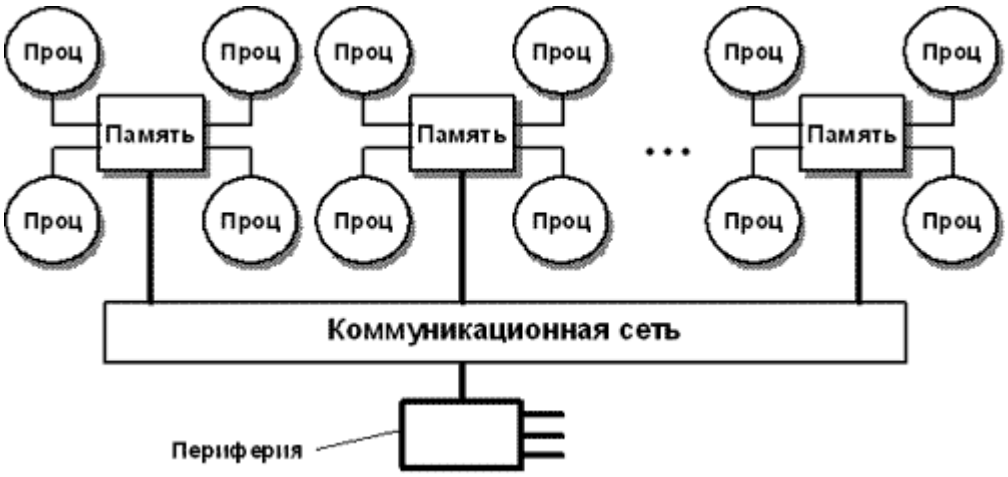


Рис. 9.2.3 – Структурная схема компьютера с гибридной сетью: четыре процессора связываются между собой при помощи коммутатора в рамках одного SMP-узла. Узлы связаны сетью типа «бабочка»

Представители этого класса: SGI Origin2000, SGI Altix3000, NUMA-Q 2000, системы Exemplar.

9.3. Кластерные системы

Кластерные технологии являются продолжением развития архитектуры MPP систем. Кластер – это связанный набор компьютеров, используемый в качестве единого вычислительного ресурса. Для создания кластеров обычно применяются либо простые однопроцессорные персональные компьютеры, либо двух- или четырехпроцессорные SMP-серверы. При этом не накладывается никаких ограничений на состав и архитектуру узлов. Каждый из узлов может функционировать под управлением своей собственной операционной системы. Чаще всего используются стандартные ОС: Linux, FreeBSD, Solaris, Tru64 Unix.

В качестве узлов используются обычные компьютеры. Развитие коммуникационных технологий, появление высокоскоростного сетевого оборудования и программного обеспечения, реализующего механизм передачи сообщений над стандартными сетевыми протоколами, сделали кластерные технологии общедоступными.

Кластерные технологии для достижения необходимой производительности объединяют различные компьютеры от персональных до суперкомпьютеров.

Производительность систем с распределенной памятью зависит от производительности коммуникационной среды, которая представляется двумя параметрами: временем задержки и скоростью передачи информации.

Разработаны технологии соединения компьютеров в кластер с использованием технология Fast Etherne, что обусловлено простотой ее использования и низкой стоимостью коммуникационного оборудования, но при этом недостаточна скорость обмена. Это оборудование обеспечивает максимальную скорость обмена между узлами 100 Мб/сек, тогда как скорость обмена с оперативной памятью составляет 250 Мб/сек и выше. Предлагают специализированные кластерные решения на основе более скоростных сетей.

9.4 Суперкомпьютеры

Векторно-конвейерный суперкомпьютер NEC SX-6 линейки SX компании NEC состоит из центрального процессора, подсистемы оперативной памяти и подсистемы ввода-вывода (рис.в со-

временных микропроцессорах). Данные компоненты объединяются в узлы SMP-архитектуры, связанных через коммуникационные межсоединение. При этом вся память всех узлов является общей; поэтому многоузловые модели SX обладают архитектурой NUMA.

Каждый центральный процессор в NEC SX состоит из двух основных блоков: векторного и скалярного устройств. В архитектуре SX имеются операционные векторные регистры команд и векторные регистры данных. Исполнительные блоки векторного устройства конвейеризованы. Основные конвейеры в SX – блоки сложения/сдвига, умножения, деления и логических операций.

Скалярное устройство в SX содержит кэш данных и кэш команд, а также 64-разрядные регистры общего назначения. Так, для SX-5 размеры кэш-памяти обоих типов составляют по 64 Кбайт, а число регистров общего назначения равно 128. Скалярное устройство выдает на выполнение все команды и способно декодировать до 4 команд за такт.

Подсистема памяти SMP-узлов SX доступна процессорам через неблокирующийся коммутатор, и каждая плата памяти SX-5 может иметь емкость 4 Гбайт, а весь 16-процессорный узел – до 128 Гбайт. Вся оперативная память разбита на банки.

В SX используется страничная адресация оперативной памяти, поэтому программные модули могут загружаться в несмежные области физической оперативной памяти – тем самым устраняются проблемы фрагментации.

Основные блоки подсистемы ввода-вывода в SX разгружают центральный процессор от непосредственного управления вводом-выводом. Каждое из них способно поддерживать работу многих каналов ввода-вывода при наличии соответствующих канальных плат.

Суперкомпьютеры семейства SX-6 представляют собой параллельные векторные системы с пиковой производительностью всей системы до 8 TFLOPS. В систему может входить до 128 узлов, каждый из которых включает от 2 до 8 процессоров и общую память до 64 Гбайт. Пиковая производительность одного процессора достигает 8 GFLOPS.

Архитектура S2MP состоит из узлов и маршрутизаторов, и носит название «фабрика межсоединений», связывающая узлы друг с

другом. Многие процессоры одновременно могут соединяться с другими узлами, в то время как ресурсы шины SMP-компьютера использует только один процессор (ЦП). Системная шина работает обычно в полудуплексном режиме, в то время как все «пути» являются дуплексными. ОП и порты ввода-вывода в S2MP являются глобально адресуемыми всех узлов.

Основным строительным блоком в архитектуре SMP является узел. В компьютерах Origin 2000 узлы реализованы в виде отдельных плат (рис. 9.4.3), каждая из которых содержит 1 или 2 64-разрядных ЦП.

Центральная часть каждого узла – это хаб. Он представляет собой полузаказную микросхему-коммутатор с 4 двунаправленными портами, имеющими пиковую производительность 780 Мбайт/с. Эти 4 порта связывают хаб с процессорами, ОП, подсистемой ввода-вывода и маршрутизатором. Хаб не только объединяет воедино все блоки узла – через него осуществляется также взаимодействие с другими узлами. Хаб преобразует внутренние сообщения, использующие

В системах SMP необходима высокопроизводительная подсистема памяти в виде КЭШа для обеспечения скоростных RISC-процессоров данными и командами. В архитектуре PowerScale применена новая организация управления кэш-памятью и доступа к памяти.

Выполнение прикладных систем связано с необходимостью обработки больших объемов данных и разделения доступа к этим данным между многими пользователями или программами.

Поэтому нужно обеспечивать широкополосный доступ к памяти из-за низкой вероятности нахождения соответствующих данных в кэш-памяти и возникающего интенсивного трафика между системной памятью и кэшами центрального процессора.

При увеличении числа процессоров и процессов, вероятность перемещения процессов с одного процессора на другой также возрастает, что приводит к увеличению трафика между кэшами. Поэтому нужно обеспечивать когерентность кэш-памяти.

В архитектуре PowerScale эффективность доступа к разделяемой основной памяти обеспечивается использованием сложной системной шины, представляющей комбинацию шины адре-

са/управления, реализованной классическим способом, и набора магистралей данных, соединенных посредством высокоскоростного матричного коммутатора. Преимущество состоит в том, что каждый процессор имеет прямой доступ к подсистеме памяти и использование расслоения памяти дает возможность многим процессорам обращаться к памяти одновременно.

Каждый процессорный модуль имеет свой собственный выделенный порт памяти для пересылки данных. При этом общая шина адреса и управления гарантирует, что на уровне системы все адреса являются когерентными.

Используются большие кэши второго уровня (L2), дополняющие кэши первого уровня (L1), интегрированные в процессорах PowerPC.

9.5. Многоядерные системы

Причиной для создания многоядерных процессоров стала необходимость повышения производительности компьютеров. Возможности повышения тактовой частоты уже практически исчерпаны. Проблема может быть решена за счет максимального распараллеливания вычислений Многоядерный процессор — это центральный процессор, содержащий два и более вычислительных ядра на одном процессорном кристалле или в одном корпусе. Под ядром принято понимать процессор и кэш-память (обычно первого уровня — L1). При наличии нескольких ядер необходимо обеспечить возможность их взаимодействия с основной памятью и между собой. Эта задача решается либо путем подключения ядер и памяти к общей шине, либо с помощью коммуникационной сети (рис. 9.5.1). Выбор варианта зависит от способа организации основной памяти. Если она является совместно используемой, то применяется вариант с шиной. В случае распределенной основной памяти коммуникации может обеспечить лишь сеть. Шинная организация обусловливает ограничение на число ядер. Пропускная способность шины ограничивает число ядер величиной 32, поскольку дальнейшее увеличение количества ядер ведет к снижению производительности многоядерного процессора.

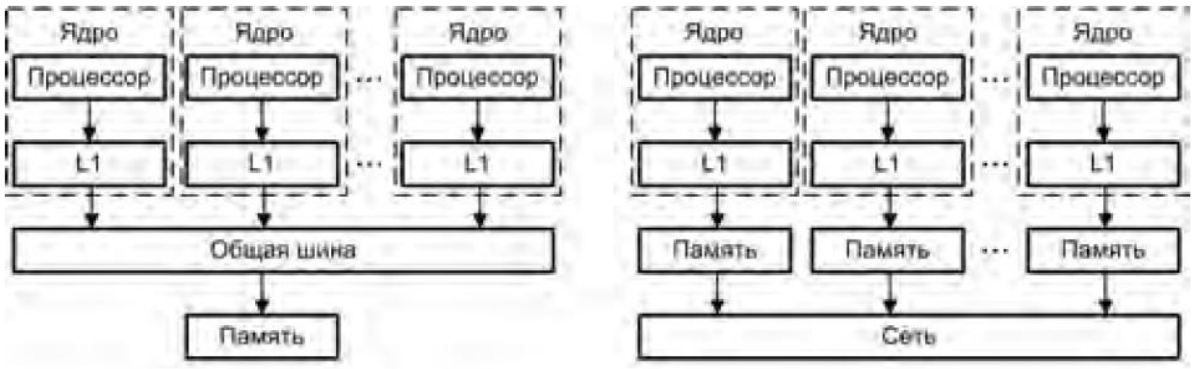


Рис. 9.5.1. Варианты многоядерных процессоров: а — с совместно используемой памятью; б — с разделенной памятью

В плане реализации многоядерные архитектуры различных производителей существенно различаются (рис. 9.5.2). Это касается не только системы коммуникаций, но и организации иерархии кэш-памяти. Во всех вариантах каждый процессор обладает кэш-памятью первого уровня. Кэш-память второго уровня может быть либо индивидуальной для каждого ядра, либо общей для группы ядер (обычно одна память L2 на пару ядер). В новейших многоядерных процессорах присутствует кэш-память третьего уровня L3. В известных микросхемах L3 является общей для всех ядер.

Каждый производитель отдает предпочтение своим архитектурным решениям.. Так, в процессорах фирмы Intel ядра поддерживают технологию гиперпотоков . В процессорах фирмы AMD связь между ядрами организована непосредственно на процессорном кристалле, а не через материнскую плату. Это решение существенно повышает эффективность взаимодействия ядер. Архитектура процессоров

В рамках одного ядра такое распараллеливание достигается, например, за счет гиперпотоковой обработки (ядром одновременно обслуживаются два потока), 4 ядра позволяют обслуживать 8 программных потоков. Такая возможность обеспечивается, главным образом, операционной системой, поскольку в плане аппаратуры ядра достаточно самостоятельны. Особые средства для взаимодействия ядер, помимо системы коммуникаций и, возможно, общей кэш-памяти третьего уровня, обычно отсутствуют.

Применение многоядерной технологии позволяет повысить производительность компьютеров и избежать роста потребления энергии, накладывающего ограничения на развитие одноядерных процессоров.

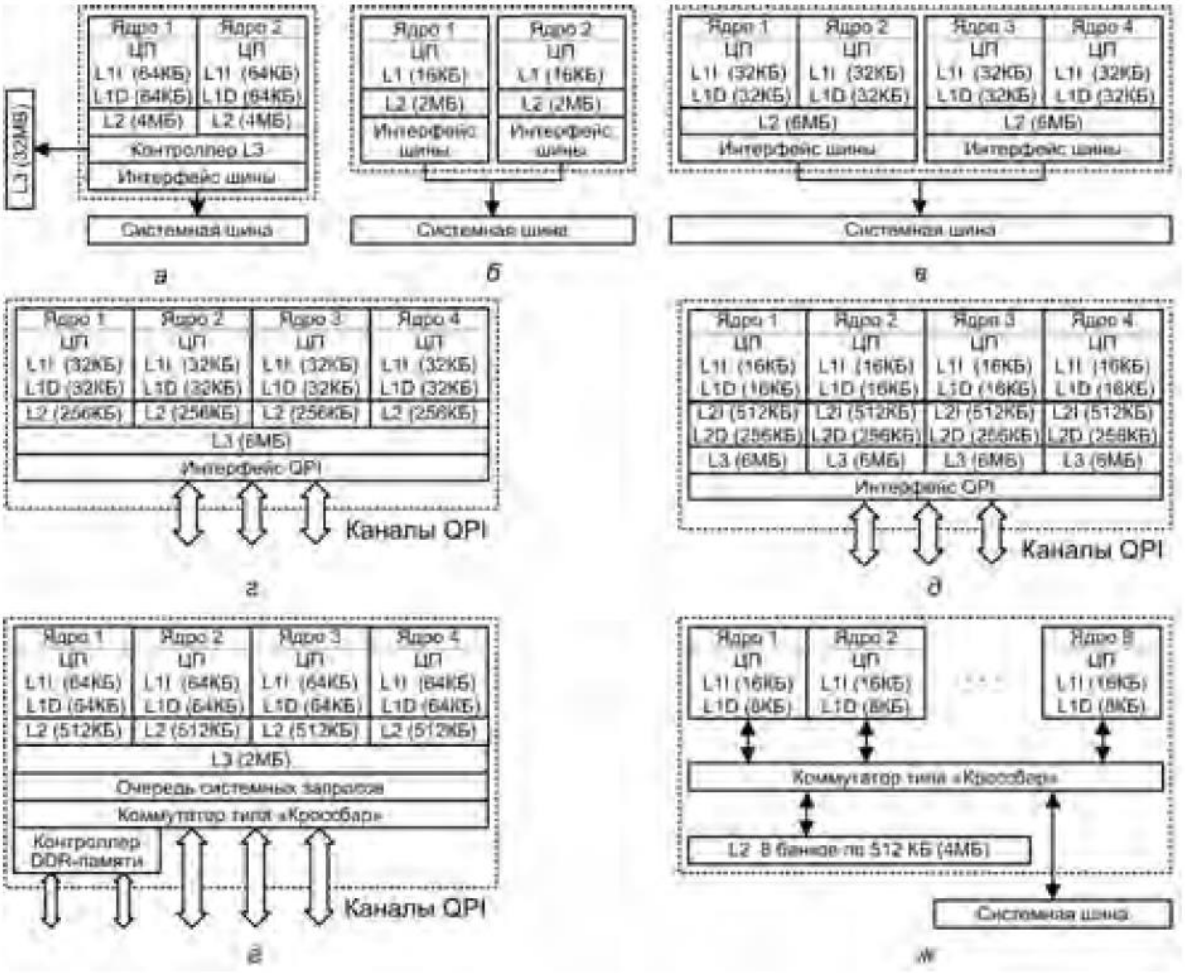


Рис. 9.5.2. Структура многоядерных процессоров различных производителей: а — IBM Power 6; б — Intel Pentium D; в — Intel Core 2 Quad; г — Intel Nehalem; д — Intel Itanium 3 Tukwila; е — AMD Phenom X4; ж — Sun UltraSPARC T2

. Также, чем выше частота процессора, тем большее время он простаивает при обращении к памяти. Поскольку частота памяти растет медленнее, чем частота процессоров, прирост производительности за счет нескольких ядер более предпочтителен.

9.6. Метрики определения степени ускорения

В параллельных вычислительных системах повышение скорости вычислений осуществляется за счет распределения вычислительной нагрузки по множеству параллельно работающих

процессоров. Система из n процессоров должна ускорить вычисления в n раз. В реальности достичь этого показателя не удается. Основная проблема состоит в невозможности полного распараллеливания вычислений, потому что в каждой программе всегда имеется часть кода, выполняемая последовательно и только одним из процессоров. Это часть программы, отвечающая за запуск задачи, распределение распараллеленного кода по процессорам, фрагмент программы, обеспечивающий операции ввода/вывода. Даже для задач, которые поддается распараллеливанию, нужно чтобы параллельные ветви программы постоянно загружали бы все процессоры системы при одинаковой нагрузке на каждый процессор, что практически трудно реализовать.Закон Амдала позволяет оценить степень реального ускорения

Джин Амдал (разработчик IBM 360) предложил формулу ускорения вычислений, достигаемого на многопроцессорной вычислительной системе, от числа процессоров и соотношения между последовательной и параллельной частями программы. Показателем сокращения времени вычислений служит такая метрика, как степень ускорения. В формуле степень ускорения S — это отношение времени T_s, затрачиваемого на проведение вычислений на однопроцессорном компьютере, ко времени T_p решения этой же задачи на параллельной системе:

$$S = \frac{T_s}{T_p}.$$

При оценке степени ускорения обязательно нужно исходить именно из наилучших алгоритмов последовательной и параллельной обработки информации.

Программный код задачи состоит из двух частей: последовательной и параллельной. Долю последовательных операций одного из процессоров обозначим через f, где $0 \le f \le 1$ (доля выражаетсяпо числу реально выполняемых операций), а доля параллельной части программы,составит $1 - f$. Крайние случаи в этих значенияхсоответствуют полностью параллельным ($f = 0$) и полностью последовательным ($f = 1$) программам. Параллельная

часть программы равномерно распределяется по всем процессорам.

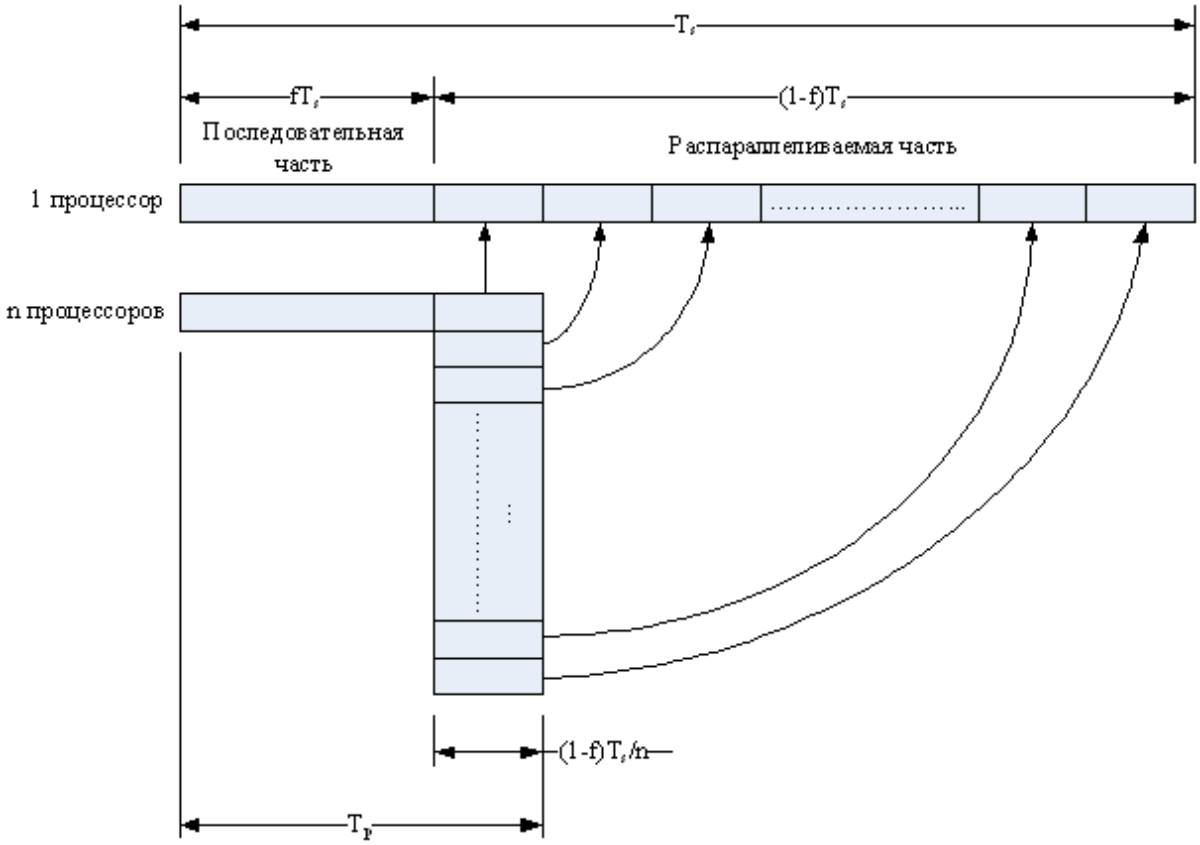


Рис.9.6.1. Представление закона Амдала.

Исходя из этого, имеем:

$$T_p = f \times T_s + \frac{(1-f) \times T_s}{n}.$$

Формула Амдала, выражающаястепень ускорения, для вычислительной системы из n процессоров, имеет вид

$$S = \frac{T_s}{T_p} = \frac{n}{1 + (n-1) \times f}.$$

Формула выражает простую и обладающую большой общностью зависимость. При устремлении числа процессоров к бесконечность, в пределе получаем:

$$\lim_{n \to \infty} s = \frac{1}{f}.$$

Это означает, что если в программе 10% последовательных операций (то есть f=0,1), то, сколько бы процессоров ни использовалось, убыстрения работы программы более чем в десять раз никак ни получить, причем, 10 — это теоретическая верхняя оценка лучшего случая, когда никаких других отрицательных факторов нет.

Джон Густафсон из NASA решал на вычислительной системе из 1024 процессоров три больших задачи, для которых доля последовательного кода лежала в пределах от 0,4 до 0,8%. Значения ускорения он получил по сравнению с однопроцессорным вариантом, равные соответственно 1021,1020 и 1016. По закону Амдала для данного числа процессоров и диапазона f, ускорение не должно было превысить величины порядка 201.

Причина заключалась в исходной предпосылке, лежащей в основе закона Амдала: увеличение числа процессоров не сопровождается увеличением объема решаемой задачи. Поведение пользователей реально отличается от такого представления. Пользовательполучая в свое распоряжение более мощную систему, не стремится сократить время вычислений, а, сохраняя его практически неизменным, старается пропорционально мощности вычислительной системы увеличить объем решаемой задачи. При этом оказывается, что увеличение общего объема программы касается только параллельной части программы, чтоприводит к сокращению значения f. На рис.9.6.2 видно, что, оставаясь практически неизменной, последовательная часть в общем объеме увеличенной программы имеет уже меньший удельный вес.

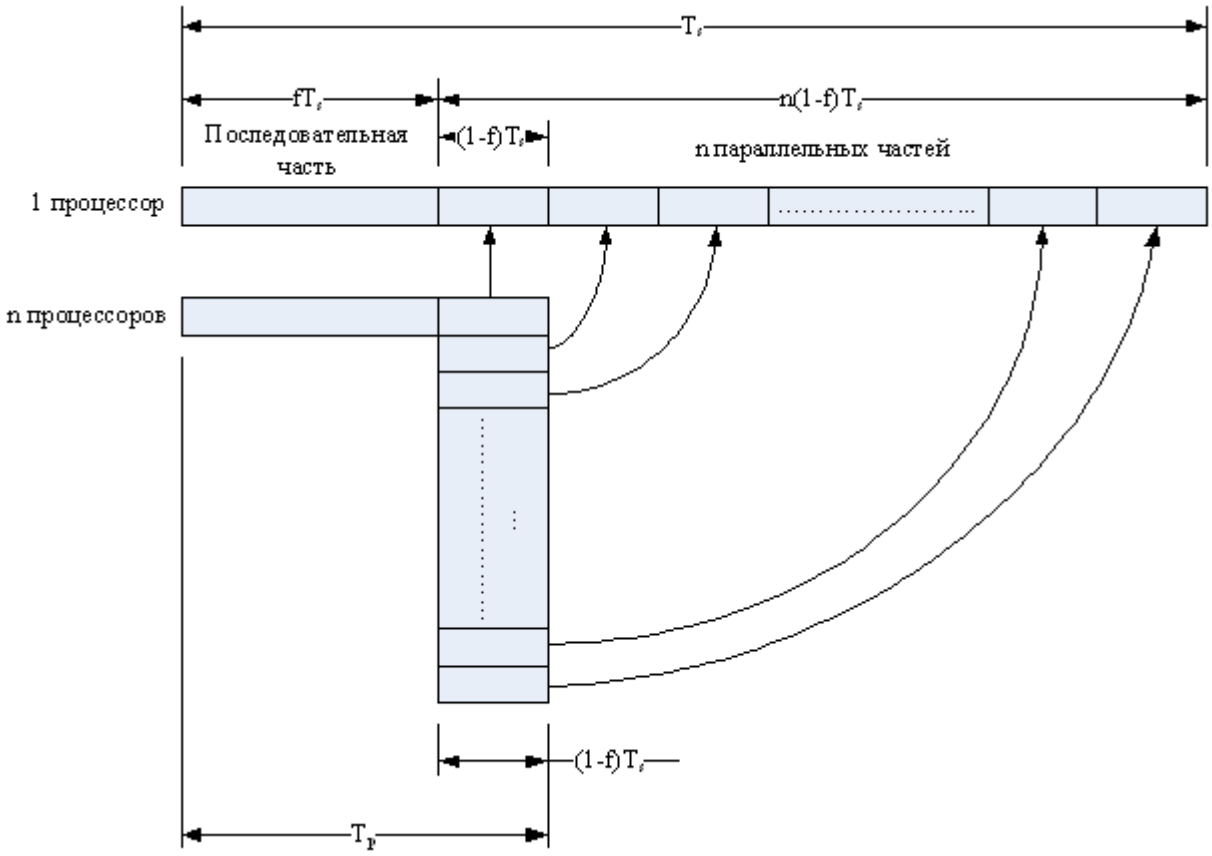


Рис. 9.6.2. К постановке задачи в законе Густафсона.

В первом приближении объем работы, выполняемый параллельно, возрастает линейно с ростом числа процессоров в системе. Для оценки степени ускорения вычислений, когда объем последних увеличивается с ростом количества процессоров при постоянстве общего времени вычислений, Густафсон использует выражение,

$$S = \frac{T_S}{T_P} = \frac{f \times T_S + n \times (1-f) \times T_S}{f \times T_S + (1-f) \times T_S} = n + (1-n) \times f .$$

Это закон масштабируемой степени ускорения или закон Густафсона (закон Густафсона-Барсиса). Закон Густафсона не противоречит закону Амдала, аразличие состоит лишь в форме использования дополнительной мощности вычислительной системы, возникающей при увеличении числа процессоров.

ГЛАВА 10. НЕЙРОКОМПЬЮТЕРНЫЕ СИСТЕМЫ

10.1 Классификация нейросетевых архитектур. Практическая реализация нейрокомпьютерных систем обычно представляет разработку программного продукта в различных средах моделирования. Аппаратные реализации до сих пор не имеют распространения. Тем не менее, уже разработаны аппаратные нейросетевые архитектуры, классификации, которых представлена на рис. 10.1.1.

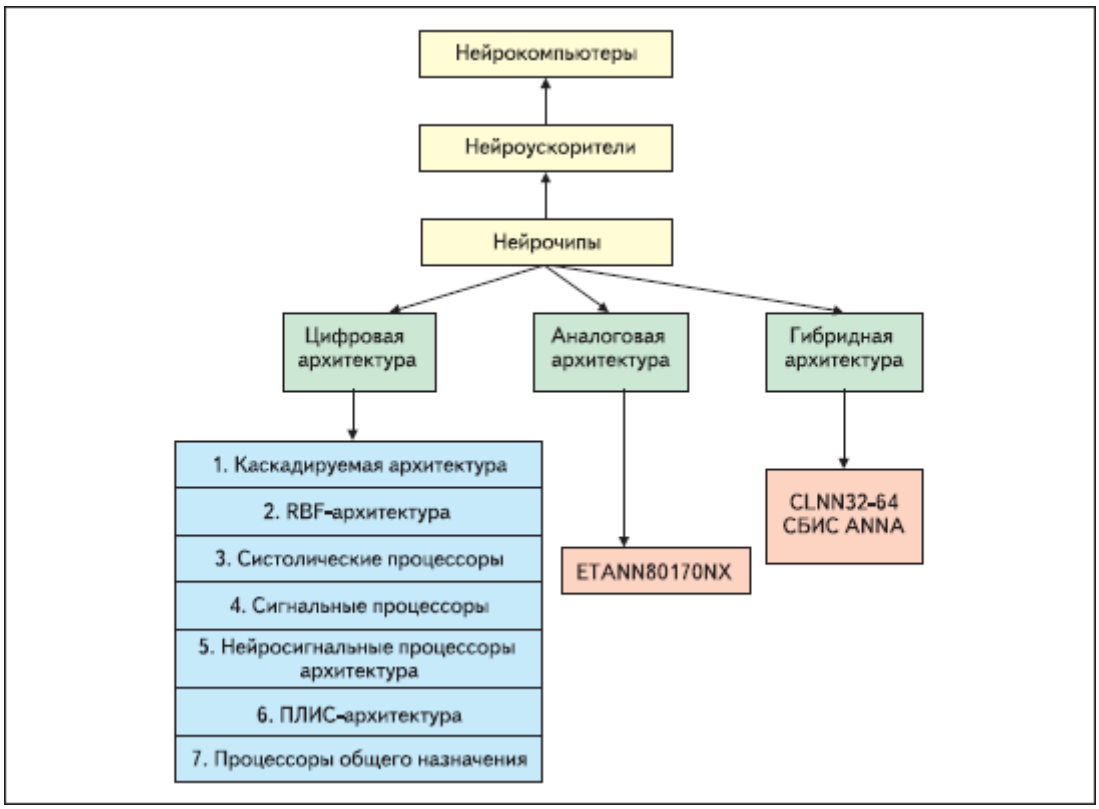


Рис. 10.1.1- Классификация нейросетевых архитектур по типу элементной базы

10.2 Понятие нейронной сети

Искусственная нейронная сеть (ИНС) — математическая модель, а также её программная или аппаратная реализации построены по принципу организации и функционирования биологических нейронных сетей. Понятие ИНС возникло в 1953 г. при изучении процессов, протекающих в мозге, и при желании смоделировать эти процессы. ИНС представляют систе-

му взаимодействующих между собой простых процессоров (искусственных нейронов). Каждый нейрон подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим нейронам. Простые процессоры (нейроны), соединённые в большую сеть с управляемым взаимодействием, способны выполнять сложные задачи обработки трудноформализуемой информации. Существуют различные точки зрения на феномен нейронных сетей.

- С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п.
- С математической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации.
- С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники.
- С точки зрения развития вычислительной техники и программирования, нейронная сеть обеспечивает эффективный параллелизм.
- С точки зрения философии и искусственного интеллекта, ИНС представляет философский подход к коннективизму и основным направлениям по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Базовым элементом нейронной сети является формальный нейрон, имеющий некоторое множество однонаправленных входных каналов аналоговой или булевой информации и один или несколько выходных каналов. По аналогии с биологическими нейронами входные каналы называются синапсами, а выходные - аксонами. Нейрон может быть цифровым, то есть оперирующим с битовыми сигналами, или аналоговым. При работе с аналоговым нейроном обычно производится нормирование сигналов таким образом, чтобы значения входов и выходов были в интервале (0,1). Каждому входному каналу с номером j формального нейрона ставится в соот-

ветствие число w_j, называемое весом этого канала. Вектор весов является внутренней характеристикой формального нейрона с n входами, рис. 10.2.1.

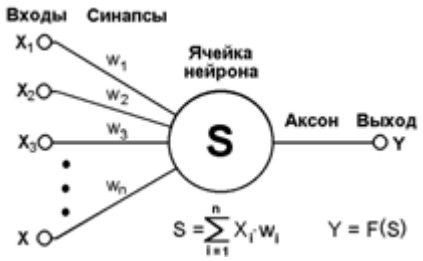


Рис. 10.2.1 -. Искусственный нейрон

Текущее состояние искусственного нейрона с n входными каналами определяется как взвешенная сумма его входов: $s = \sum_{j=1}^{n} w_j x_j$, где x есть входной сигнал.

Выходной сигнал нейрона есть функция его состояния $y = f(s)$, являющаяся его внутренней характеристикой и называемая активационной функцией. В случае булевых искусственных нейронов используется пороговая функция. Для обработки сигнала внутреннего возбуждения в случае аналогового нейрона используется логистическая функция или сигмоид с внутренней нелинейной характеристикой f(s), рис.10.2.2.

При уменьшении значений функции f сигмоид становится более пологим и при f = 0 вырождаясь в горизонтальную линию на уровне 0.5, при увеличении f сигмоид приближается по внешнему виду к функции единичного скачка с порогом в точке $s = 0$. Эта функция используется благодаря таким свойствам, как дифференцируемость на всей оси абсцисс, причем её производная имеет простой вид $F(s) = F(s)(1-F(s))$ и гладкость функции.

Сеть состоит из произвольного количества нейронов, разбитых на слои. Нейроны каждого слоя соединяются только с нейронами предыдущего и последующего слоев (по принципу каждый с каждым), выделяют

входной и выходной слои, остальные слои называются внутренними или скрытыми. Каждый слой рассчитывает нелинейное преобразование от линейной комбинации сигналов предыдущего слоя. Многослойная сеть может формировать на выходе произвольную многомерную функцию при соответствующем выборе количества слоев, диапазона изменения сигналов и параметров нейронов.

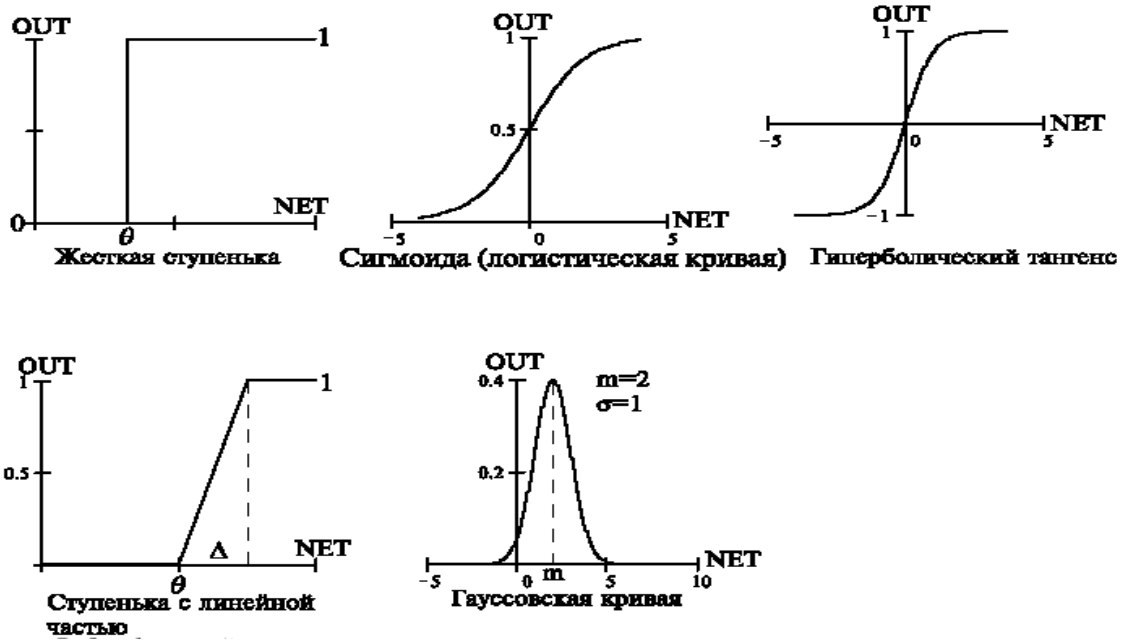


Рис.10.2.2- Виды активационных функций

Искусственные нейроны объединяются в сети различными способами, но самым распространенным видом нейросети стал многослойный перцептрон, рис10.2.3..

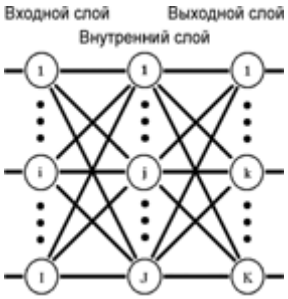


Рис. 10.2.3 - Многослойный перцептрон

1. **Необходимость обучения сети.** Решение, которое формирует нейросеть, определяется весами входных каналов и видом активационной

функции нейронов. Расчет вектора весов обычно выполняется один раз и в процессе функционирования нейросети этот вектор фиксирован. Но в некоторых случаях необходимо использовать самообучающуюся нейронную сеть, тогда выполняется динамический пересчет весов, а иногда и изменение структуры сети.

Обучение проводят по обучающему множеству, для которого должны быть известны выходы нейросети. В зависимости от разницы между ожидаемым и фактическим выходом нейросети проводится подстройка весов каждого нейрона. После того, как обучение закончено, достигнуто минимальное значение ошибки на всем обучающем множестве, веса фиксируются, и нейронная сеть готова к функционированию.

Другой вариант, когда выполняется создание нейронной сети, для которой не известны ее выходные значения, но есть некоторые условия которые нужно удовлетворять.В этом случае выполняется обучение без учителя.

В возможности обучения (а не программирования) заключается одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Поэтому в случае успешного обучения сеть способна сформировать правильный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искажённых данных.

Стоит заметить, что возможность объяснить принятие решений сетью отсутствует.

2. **Однородность выполняемых операций.** Нейронная сеть может содержать множество слоев нейронов, но независимо от сложности задачи выполняемые операции фиксированы.

Работа каждого слоя нейронной сети описывается одной формулой

$$Y = f(XW),$$

где X - вектор входных значений, Y - вектор выходных значений, W - матрица весовых коэффициентов, а f() - функция активации, применяемая к компонентам вектора.

Фактически вычислительных операций две: перемножение матриц XW и вычисление нелинейной функции f для каждого элемента вектора. Эта особенность позволяет, применив соответствующие решения, увеличить скорость работы нейронной сети.

3. **Независимость работы нейронов.** Выход нейрона зависит только от выходных сигналов нейронов предыдущего слоя, что обеспечивает параллельную обработку всех нейронов одного слоя, и для этого нужно сохранять только выходной вектор непосредственно предшествующего слоя. При аппаратной реализации, целесообразно применить конвейерный подход, и обрабатывать следующий входной вектор сразу после освобождения слоя нейронов, не ожидая расчета всех слоев сети.

4. **Устойчивость решения.** Выходной сигнал зависит от работы множества нейронов, причем в сложных нейросетях ни один из нейронов не оказывает сильного влияния на выход. Алгоритмы обучения сети подразумевают, что малые отклонения весов и выходов нейронов вызывают соответственно малые отклонения выходного сигнала нейронной сети, и функционирование обученной нейронной сети не нарушается малыми изменениями весов и выходов нейронов. При такой устойчивости нейронная сеть, незначительно реагирует на частичную потерю данных, аппаратные проблеммы или шумы входных данных.

5. Свойство устойчивости нейросети применяется при выборе представления чисел в нейросети. Возможно уменьшать точность величин без ухудшения качества работы нейросети. Поэтому в нейросистемах часто

применяют числа с фиксированной запятой малой разрядности, вместо чисел с плавающей запятой, занимающих много места в памяти и замедляющих вычисления.

С точки зрения построения нейрокомпьютерных систем представляет интерес процессорная реализация нейронных сетей, согласно классификации на рис.10.1.1.

10.3. Нейропроцессоры и архитектуры нейрокомпьютерных систем

Каскадируемая архитектура

Примером каскадируемой архитектуры может служить нейрочип NNP фирмы Accurate Automation Corp, представляющий несколько 16-разрядных процессоров, снабженных памятью для хранения весовых коэффициентов. Процессоры связаны локальной внутренней шиной и имеют всего 9 команд. В комплект поставки входит библиотека подпрограмм с реализованными нейросетевыми алгоритмами. Чип поставляется на платах под шины ISA и VME. Производительность составляет от 140 MIPS до 1,4 GIPS, в зависимости от количества нейрочипов на плате. Их количество может быть от одного до десяти.

Нейрокомпьютеры на процессорах общего назначения

Это распространенная структура для построения нейросистем, обычно реализованых в виде программных продуктов. Реализация нейроалгоритмов на рабочих станциях осуществляется языками высокого уровня, для которых имеются библиотеки, реализующие различные функции, используемые в нейроприложениях. Разработаны визуальные средства разработки нейроприложений, для пользования которыми даже не обязательны навыки программирования. Эти средства содержат множество типовых решений и удобный графический интерфейс, позволяющий производить настройку на конкретную задачу (Матлаб, Маткад).

Процессоры общего назначения мало подходят для создания про-

мышленных устройств мз-за архитектурной избыточности. Разработано множество нейроускорителей на этой базе, но и они не имеют преимуществ перед другими подходами.

К достоинства процессоров общего назначения можно отнести: высокая производительность, наличие языков высокого уровня, возможность интеграции нейроприложений с другими програмными средствами, большой объем доступной памяти.

Недостатки процессоров общего назначения следующие: аппаратная избыточность, сложность в реализации промышленных устройств и расширяемых многопроцессорных систем;

Применение DSP для построения нейрокомпьютеров

Цифровые сигнальные процессоры (DSP) часто применяются для реализации нейросетевых сред вследствии:

1. Рынок DSP сложился давно, отработаны технологии производства и сбыта продукции. Для DSP не нужно дорогостоящее окружение (система ввода/вывода, система внешней памяти, система синхронизации и т.д.).

2. Производители DSP сопровождают свои системы развитыми средствами разработки и отладки, разработаны компиляторы языка С, эмуляторы, средства внутрисистемной отладки и программирования.

3. DSP не накладывают ограничений на вид реализуемой нейросети, разработчик может построить сеть практически любой топологии и размеров.

4. Малое энергопотребление и тепловыделение. Поэтому нейропроцессоры уже применяются в промышленности для создания сложных датчиков и организации систем управления, где большое значение имеет низкое потребление энергии и малое тепловыделение. Например, энергопотребление DSP компании Analog Devices семейства SHARC составляет 1.5Вт при напряжении 3.3В, а у чипов С6201 компании Texas Instruments -

7Вт при 2.5В[ ]

DSP-процессоры в нейросистемах могут выполнять две функции: реализацию самой нейронной сети или реализацию контура управления нейрокомпьютерной системой.

Наиболее разработаны цифровые сигнальные микропроцессоры ADSP-2116x, тактовая частота которых повышена до 100 МГц, позволяющее увеличить производительность до 600 MFLOPS. Архитектура этого семейства на примере ADSP-21161, его основные функциональные блоки приведены на рис.10.3.1. .

В данном процессоре используется модифицированная Гарвардская архитектура (SHARC - Super Harvard ARChitecture), предполагающая разделение памяти и шин команд и данных (по шине команд могут передаваться как инструкции, так и данные). Имея два генератора адреса данных процессор может одновременно генерировать два адреса для выборки двух операндов.

Работа нейросети включает операции с двумя операндами (часто используемая операция - умножение значения на входе нейрона и соответствующего ему весового коэффициента), и выборка двух операндов за один машинный цикл позволяет значительно увеличить скорость, нейросети. Так как операции с весами проводятся отдельно от операций с входными величинами, то распределение перемножаемых чисел по разным банкам внутренней памяти достаточно просто выполняется.

Для выборки очередной инструкции предусмотрен кэш команд на 32 48-ми битных инструкции. Если текущая инструкция находится в кэше, то выборки из памяти команд не происходит и шины свободны для выбора второго операнда в этом же цикле.

Система команд ADSP-21161 содержит арифметическую команду умножения с накоплением ($C = A \cdot B + C$), с указанием числа выполнений в цикле и правил изменения индексов для адресации A и B. Эта команда

должна быть выполнена для каждого входа нейрона при просчете функционирования сети.

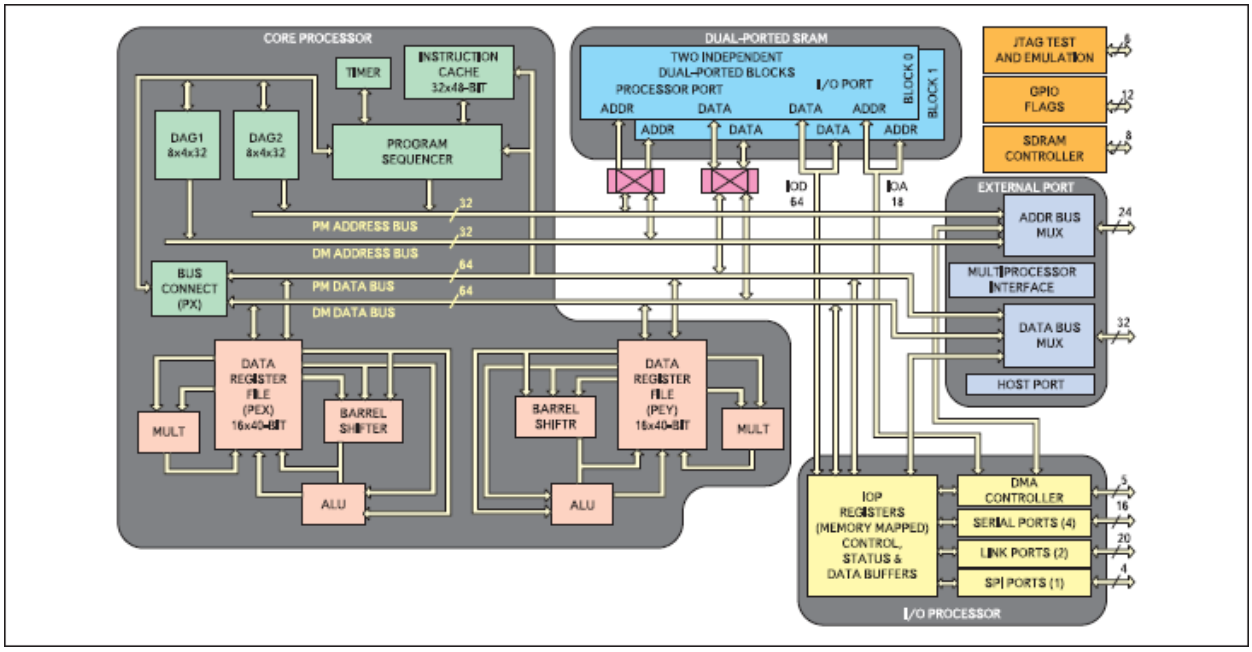


Рис. 10.3.1 - Блоки ADSP-21161 (SIMD-режим)

DSP архитектуры SHARC организовывают мультимикропроцессорные системы двумя способами: через соединения ‘точка-точка” (Рис10.3.2)- и кластерное соединение (Рис. 10.3.3).

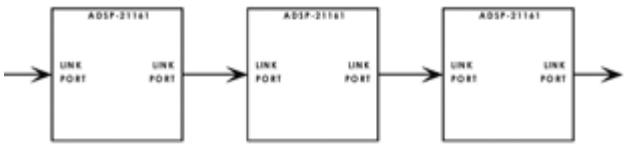


Рис. 10.3.2 - ‘точка-точка”

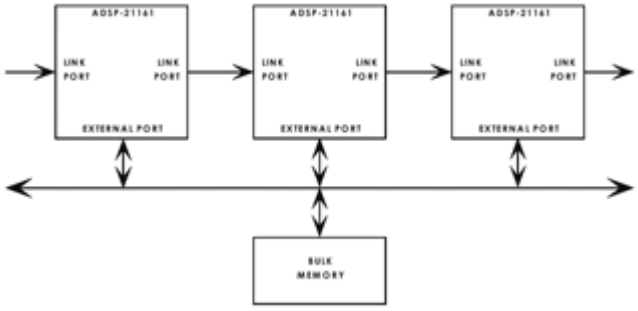


Рис10.3.3 - Кластерное соединение

Для первого типа обмен информацией идет через два имеющихся 8-ми битных двунаправленных порта связи. Прохождение потока данных через все микропроцессоры, ускоряет обработку, распределяя слои сети по отдельным узлам. После расчета своих слоев узел, передает данные следующему процессору и принимает следующие входные данные. Этот тип мультимикропроцессорных систем наиболее прост для построения и наиболее компактен при размещении на плате.

Кластерная система более гибкая, так как наращивание числа DSP и внешней памяти возможно без существенного изменения алгоритмов. Кластерная система включает несколько DSP, подключенных к общим шинам, с поддержкой межпроцессорного доступа к памяти и обращения к общей разделяемой памяти.

В кластерной конфигурации узким местом является общая шина, так как только два DSP могут участвовать в передаче информации в течение каждого цикла - остальные DSP ожидают освобождения шины. Поэтому в кластерной конфигурации задействованы порты связи для передачи данных без блокирования общей шины.

Обрабатывающие элементы PEx и PEy (Processing Elements) производят независимую вычислительную обработку. Каждый PE содержит регистровый файл и три вычислительных блока: арифметико-логическое устройство, умножитель и устройство сдвига. Вычислительные инструкции для этих элементов могут выполняться за один цикл, вычисления могут проводиться с фиксированной и плавающей запятой.

В обычном режиме SISD DSP использует только PEx. В режиме SIMD DSP параллельно выполняет следующие операции:

- Направляет единственную инструкцию в оба обрабатывающих элемента

- Загружает два набора данных из памяти в каждый элемент
- Выполняет одну и ту же команду одновременно в двух элементах
- Сохраняет результаты совместного выполнения в памяти

DSP, используя SIMD-команды, может удвоить скорость обработки данных. В нейроприложениях это важно, потому что выполняются одинаковые операции для большого набора однородных данных.

Применение ПЛИС для построения нейросред

ПЛИС (Программируемые Логические Интегральные Схемы) применяются для построения устройств управления и контроля и в том числе для построения нейрокомпьютерных систем. ПЛИС, выполненные по современной 0,22-микронной технологии, работают на частотах до 300 МГц и реализуют до 3 млн. эквивалентных логических вентилей. Компания Xilinx, уже объявила о выпуске ПЛИС в 10 млн. логических вентилей.

Исходя из принципа формирования структуры ПЛИС для нейрокомпьютерных систем, подразделяются на две группы:

- ПЛИС CPLD (Complex Programmable Logic Devices). Это устройства, в которых нужная структура формируется программированием связей коммутирующих матриц с использованием технологий перепрограммируемых постоянных запоминающих устройств: FLASH или EPROM. CPLD содержат несколько логических блоков, объединенных коммутационной матрицей. Каждый логический блок представляет программируемую матрицу И и фиксированную матрицу ИЛИ. Для изменения структуры необходимо выполнить операции стирания и программирования новой структуры. Эта вид ПЛИС энергонезависим.
- ПЛИС FPGA (Field Programmable Gate Array), которые управляются битовой последовательностью, хранящейся во внутреннем статическом ОЗУ. ПЛИС должна восстанавливаться (записываться во внутреннее статическое ОЗУ) после каждого включения питания, и поэтому нужно со-

хранять и восстанавливать конфигурацию.

ПЛИС семейства AT40K, выпускается компанией Atmel со следующими. характеристиками:

- системная частота до 100 Мгц
- буфер на 3 состояния для каждой ячейки
- до 2Кбайт внутренней 10 нс SRAM
- модификации для напряжения питания 3.3В и 5В
- модификации для разных диапазонов температур

Концепция ПЛИС предполагает соединения вычислительных ячеек. Основным архитектурным блоком является матрица одинаковых ячеек (рис. 10.3.4). Массив однороден по всей площади, кроме шинных повторителей, отделяющих каждые четыре ячейки.

Рис. 10.3.4- Массив ячеек ПЛИС AT40K

На пересечении вертикальных и горизонтальных линеек повторителей расположен блок RAM 32x4, доступный для смежных ячеек и сконфигурированный для синхронного и асинхронного доступа и для одно- и двухпортового режима работы.

Основой ячейки FPGA является логическая таблица (LUT - look-up table), представляющая собой однобитное ОЗУ на 8 ячеек. Различные варианты соединения логических таблиц и программируемого D-триггера

позволяют создавать на базе ячеек различные устройства: регистры, сумматоры, умножители, счетчики.

.Преимущества ПЛИС для реализации нейросистем:

- Высокое быстродействие до 300МГц, чипы выполняются по технологии до 0.09 мкм на девяти слоях металла.
- Реализации параллельных алгоритмов из-за расположения на одном кристалле множества блоков с интенсивным обменом информацией между ними.
- Возможность перепрограммирования в системе, что обеспечивает гибкость системы в целом.
- Совместимость при переносе алгоритмов на уровне языков описания аппаратуры VHDL, AHDL, Verilog и др..
- Возможность реализации стандартного интерфейса. Любые FPGA-чипы имеют в своем составе блоки ввода/вывода, настроеваемые под стандарты входов микросхем.
- Наличие библиотек мегафункций, описывающих сложные алгоритмы.
- Быстрота создания нейросистем.

Реализуется нейросистема на ПЛИС двумя способами:

- Непосредственная построение нейроархитектуры. Различные участки ПЛИС реализуют отдельные нейроны, между которыми программируются соединения, в том числе с обратными связями, с разными типами нейронов, с неоднородностями в топологии межсоединений.

Это решение имеет следующие достоинства:

- возможно обеспечить высокую параллельность обработки данных
- возможно реализовать любую топологию нейросети и вид нейронов
- возможно менять структуру сети на аппаратном уровне

Недостатки решения на ПЛИС:

- отсутствие готовых решений
- высокая схемотехническая сложность реализации отдельного нейрона.

На ПЛИС с интеграцией в 40 тысяч эквивалентных вентилей можно реализовать до 15 параллельно работающих нейронов, а доступны ПЛИС с интеграцией в 2.67 миллиона вентилей (EP20K1000 фирмы Altera), что дает возможность реализовать, соответственно, 1000 параллельных нейронов. Такое число нейронов уже достаточно для серьезных нейроприложений.

Применение аналоговых и гибридных систем для построения нейросред

Гибридные и аналоговые решения также могут обеспечивать достаточное быстродействие при эмуляции нейронной сети. Например, аналоговая СБИС ETANN 80170NX фирмы Intel и гибридное решение СБИС CLNN32/CLNN64 фирмы Bellcore.

СБИС ETANN 80170NX.

СБИС содержит 64 входа и 64 нейрона (пороговый усилитель с сигмоидной передаточной функцией). Каждый вход соединен с 64 синапсами. Активационная функция нейрона в СБИС близка к сигмоиде, и параметр наклона может изменяться.

Усиление передаточной функции определяет чувствительность нейрона. Низкое значение усиления позволяет интерпретировать выход нейрона как аналоговый, а высокое - как цифровой. Максимальное значение выхода нейрона также задается. Веса ограничены интервалом (-2.5,2.5). Скорость прохождения сигнала по одному слою зависит от усиления и примерно равна 1.5 мкс, что и определяет быстродействие. Точность выполнения операций примерно эквивалентна 6 бит, быстродействие -

1,3... 109 переключений/с.

Обучение выполняется методом обратного распространения ошибок с помощью Intel Neural Network Training System (INNTS), работающей на компьютере i486. Применяемое системное окружение представляет собой специальную версию пакета DynaMind. Обучение выполняется до получения приемлемого уровня ошибки выхода сети, и после достижения удовлетворительной работы веса загружаются в СБИС.

Надежная работа СБИС обеспечивается модулем с низкой пульсацией источника питания и температурной стабильностью/

СБИС CLNN32/CLNN64 фирмы Bellcore.

Гибридный нейрочип CLNN32 состоит из 32 нейронов и 496 двунаправленных адаптивных синапсов. CLNN64 содержит 1024 адаптивных синапсов. В наборе CLNN32/CLNN64 все нейроны взаимосвязаны, так что любая топология сети отображается подбором синапсов. Динамика сети полностью аналоговая, но значения синапсов хранятся/обновляются в цифровом виде с точностью 5 бит. На аппаратном уровне реализовано обучение сети - подбор весов происходит по алгоритму обучения машины Больцмана..

Производительность CLNN32 достигает 108 переключений/с (при работе с CLNN64 удваивается). Для CLNN32 это означает, что примерно 105 32 битных образцов/с или 32 аналоговых канала (с полосой пропускания 50 кГц) могут быть использованы для быстрого распознавания/обучения. Время распространения для одного слоя нейронов <1мкс. «Охлаждение>> (по методу Больцмана) или MF обучение требует 10... 20мкс.

По сравнению с ETANN СБИС CLNN32 имеет следующие очевидные преимущества:

- быстрое обучение (микросекунды по сравнению с часами при CIL-процессе;

- эффективный алгоритм обучения Больцмана, обеспечивающий быстрое нахождение хорошего решения (особенно хорош для задач распознавания изображений);
- простые и быстрые процедуры чтения/записи весов, выполняемые в цифровом виде
- легкая каскадируемость.

Применение систолических процессоров для построения нейросред

Систолические процессоры используют специальные обрабатывающие элементы, простые по своим функциям и структуре. Эти элементы образуют процессорную матрицу, через которую идет поток данных, изменяемых каждым элементом. Высокая степень параллельности обработки данных достигается, если отработавший элемент сразу же считывает следующую порцию данных для обработки. Систолы больше всего напоминают систему сосудов и сердце, которое постоянно посылает кровь во все артерии, сосуды и капилляры тела.

Базовые принципы построения систолических архитектур:

1. Систола представляет собой сеть связанных простых вычислительных ячеек;
2. Каждая ячейка содержит в себе буферный входной регистр, защёлкивающий данные и вычислитель, оперирующий с содержимым этого регистра. Выход вычислителя может подаваться на входы других ячеек;
3. Операции в систоле производятся по типу конвейерной обработки;
4. Вычисления в систоле регулируются с помощью общего тактового сигнала;
5. реализуются практически все виды параллелизма;
6. модульная структура систол позволяет расширять систолы;
7. вычислительная эффективность систол определяется правильным

построением конвейера,

8. систолы эффективно используют полосу пропускания каналов доступа к памяти, отличаясь предсказуемым обращением к памяти.

На рис. 10.3.5 приведена схема типичной систолы:

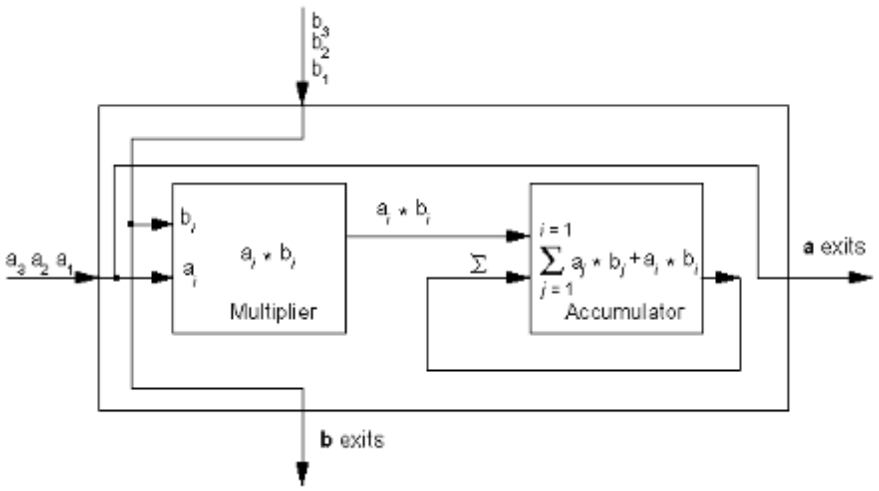


Рис. 10.3.5 - Систола, дающая на выходе скалярное произведение массивов чисел

Структура систолы имеет ряд недостатков:

Общая тактовая частота должна быть такой, чтобы за время одного такта успевали полностью отработать все вычислители, на данном рисунке умножитель явно отрабатывает за в несколько раз большее время, чем сумматор.

Чип SAND (Simple Applicable Neural Device) способен реализовывать нейросети с максимальным числом входов 512, Рис.10.3.6

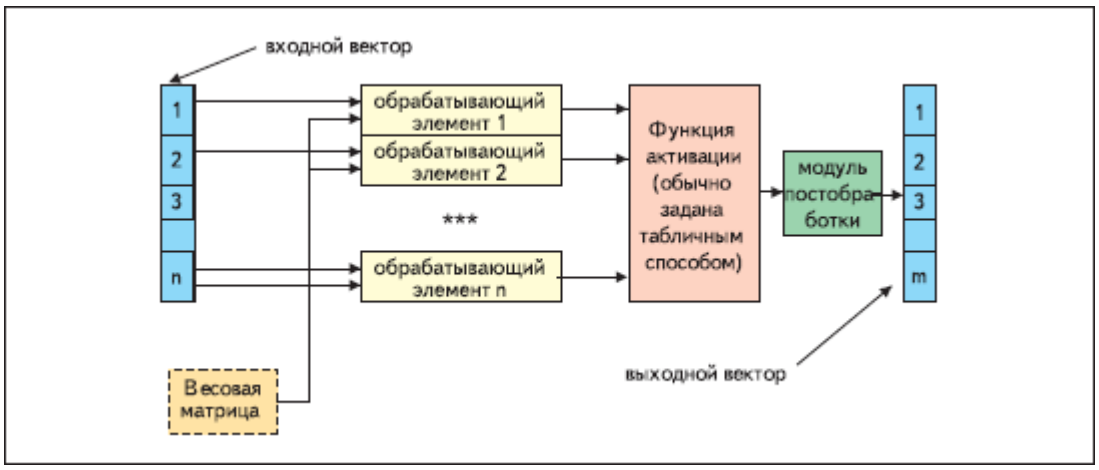


Рис. 10.3.6 - Структура чипа SAND

Архитектура SAND показана на рис. , содержит четыре параллельных обрабатывающих элемента PE (Processing Elements), каждый из которых снабжен АЛУ и блоками отсечения (auto-cut). АЛУ используется для умножения векторов. Так как АЛУ накапливает входные значения, выходная шина имеет разрядность 40 бит (это ограничивает число входных нейронов до 512). Блок отсечения снижает разрядность до 16 с контролем переполнения и потери точности. Окно, выбирающее 16 бит из 40 может быть смещено пользователем по его усмотрению.

Систолический процессор CNAPS

Данная архитектура разработана компанией Adaptive Solutions, представляет собой SIMD мультипроцессор общего назначения, разработанный для нейроприложений. Чип реализует алгоритмы обучения и функционирования нейросети. Чип имеет большую емкость памяти весов, позволяя хранить 2M 1 битных весов, или 256K 8 битных весов, или 128K 16 битных весов, поровну распределенных между 64 процессорами. Чипы могут образовывать мультимикропроцессорные системы различной конфигурации.

Базовым элементом системы CNAPS является чип N6400, состоящий из 64 обрабатывающих элементов, соединенных широковещательной шиной в режиме SIMD (рис. 10.3.7) Нейросистема содержит управляющий чип (Sequencer) и четыре процессора N64000.

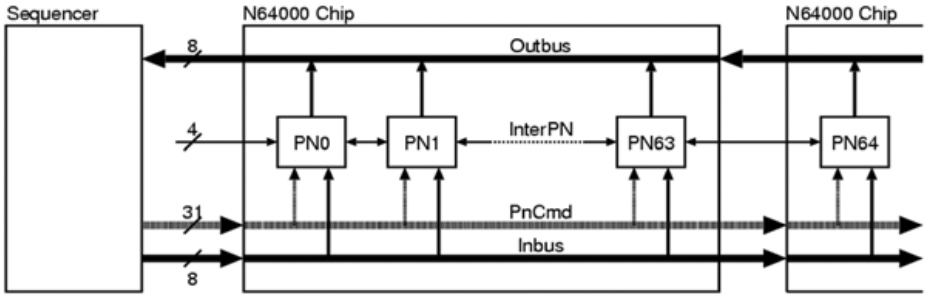


Рис. 10.3.7 - Структура системы CNAPS

Так как N64000 работает как микропроцессор общего назначения, то CNAPS может реализовать широкий класс нейросетевых алгоритмов. Для этой архитектуры создан инструментальный набор, включающий С-компилятор с надстройками, позволяющими получать код, использующий все преимущества параллельной архитектуры.

Нейросигнальные процессоры для построения нейросред

Нейросигнальные процессоры в настоящее время являются наиболее быстродействующим средством построения нейрокомпьютерных систем. Ядро нейросигнальных процессоров представляет DSP, а реализованная на кристалле дополнительная логика обеспечивает выполнение нейросетевых операций.

Структура нейропроцессора NM6403

Ядро NM6403 состоит из двух базовых блоков: 32-битного RISC процессора и 64 битного векторного процессора, обеспечивающего выполнение векторных операций над данными переменной разрядности. Для построения нейрокомпьютерных систем имеются два одинаковых программируемых интерфейса для работы с внешней памятью различного типа и два коммуникационных порта, аппаратно совместимых с портами DSP TMS320C4x. Общая структура нейропроцессора показана на рис. 10.3.8.

- Ядро нейропроцессора, выполняющее операции сдвига, арифметико-логические операции над 32 разрядными данными, формирующее адреса команд и данных, выполняющее управление работой нейропроцессора.
- VCP. Векторный сопроцессор, выполняющий арифметические и логические операции над 64 разрядными векторами упакованных данных переменной разрядности.
- LMI и GMI. Два идентичных блока программируемого интер-

фейса с локальной и глобальной 64 разрядными внешними шинами, к каждой из которых может быть подключен блок внешней памяти, содержащий до 2^{31} 32 разрядных ячеек.

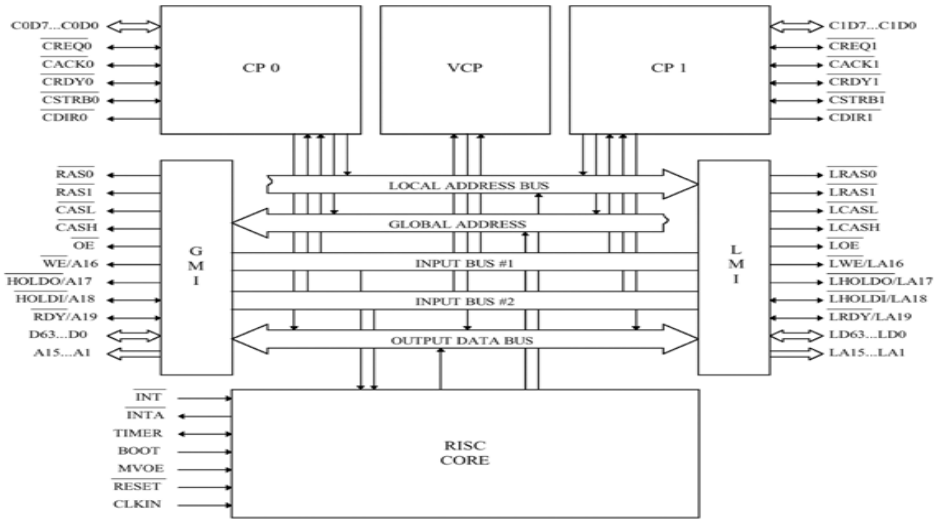


Рис. 10.3.8 - Общая структура нейропроцессора NM6403

- CP0 и CP1. Два идентичных коммуникационных порта, полностью совместимых с коммуникационным портом сигнального процессора TMS320C4x.

Нейропроцессор содержит пять внутренних шин.

На рис. 10.3.3 приведена реализуемая функция активации.

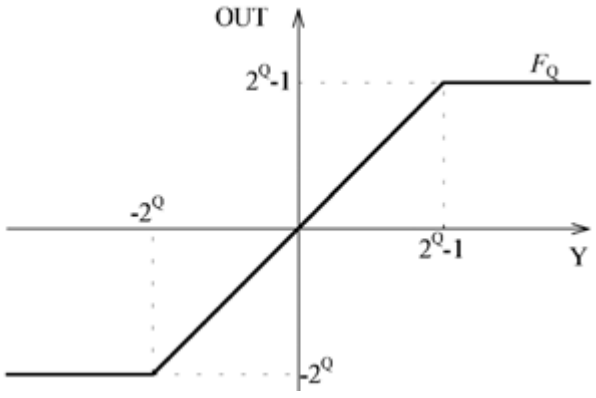


Рис. 10.3.9 - Реализуемая функция активации

Функция активации применяется к входному вектору до выполнения

операции.

Подключение к одной шине нескольких нейропроцессоров позволяет им обмениваться информацией через общую память, расположенную на этой же шине. Подключение к одной шине двух нейропроцессоров осуществляется без использования дополнительной управляющей аппаратуры. Скорость обмена данными через общую шину может достигать 400 Мбайт/с.

Возможности NM6403 ограничены, но все же они являются вполне достаточными для создания нейроприложений повышенной производительности. На рис. 10.3.10 показана реализация нейронной сети на нейропроцессоре NM6403

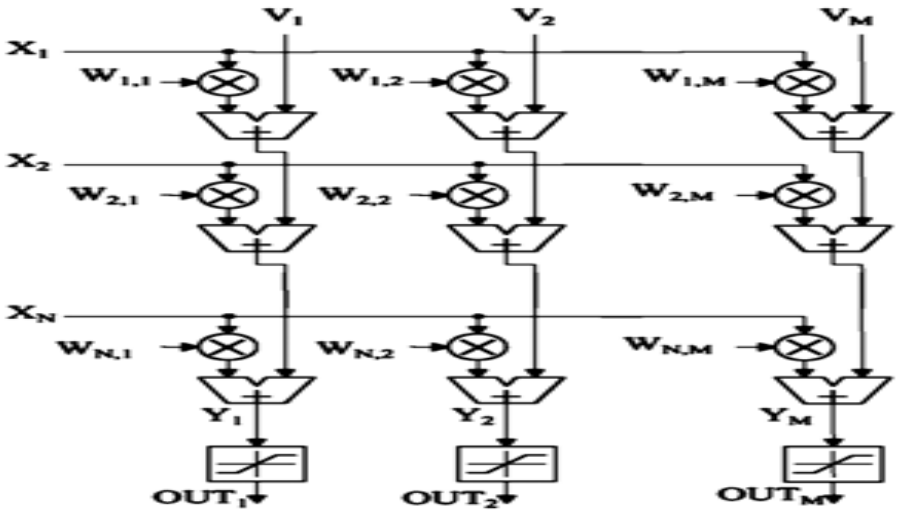


Рис. 10.3.10 - Реализация слоя сети NM6403

Обработка реакции сети на входное воздействие происходит в нейропроцессоре последовательно, слой за слоем. Каждый слой нейросети имеет N нейронных входов и состоит из M нейронов. При этом каждый нейрон выполняет взвешенное суммирование с учетом смещения данного нейрона:

Затем каждый нейрон вычисляет функцию насыщения от результата взвешенного суммирования:

$$OUT_m = F_{Qm}(Y_m).$$

Все входные данные, весовые коэффициенты, пороговые значения и результаты представляются в дополнительном коде.

Процессор NM6403 создавался как универсальное средство для реализации нейросред. При его использовании пользователь может программно задавать множество параметров нейронной сети:

- число слоев;
- число нейронов и нейронных входов в каждом слое;
- разрядность данных на каждом нейронном входе;
- разрядность каждого весового коэффициента;
- разрядность выходного значения каждого нейрона;
- параметр функции насыщения для каждого нейрона.

Многоядерные системы для построения нейросред

Технологии, связанные с использованием возможностей нейронных сетей развиваются на принципах «глубокого обучения» (Deep learning). Поэтому нейронные сети целесообразно реализовывать на многоядерных графических процессоров (GPU). Новый 168-ядерный чип, разработан для реализации алгоритмов искусственного интеллекта на основе нейронных сетей. Процессор продемонстрировал в 10 раз большую эффективность, что позволяет использовать пользовательское мобильное устройство для локального запуска мощных алгоритмов искусственного интеллекта без необходимости отправки данных для облачной обработки.

Нейронные сети, как правило, имеют многослойную структуру и каждый слой содержит большое количество обрабатывающих узлов. На начальном этапе обработки данные приходят и распределяются между узлами нижнего слоя. После обработки полученных данных каждым из узлов результат передается для обработки узлам следующего слоя. На выходе последнего слоя формируется результат решения поставленной задачи. Соответственно, для решения масштабных задач по описанному алгоритму потребуются значительные вычислительные ресурсы.

Чип Eyeriss представляет собой уже сформированную нейронную сеть, локализованную на уровне 168-ядерного процессора, который встроен в мобильные устройства.

Ключ к эффективности Eyeriss — минимизация частоты обмена данными между ядрами и внешними банками памяти, операции, связанной с большим энергопотреблением и временными затратами. В то время как ядра традиционных GPU завязаны на один общий банк памяти, каждое ядро Eyeriss располагает собственной памятью. Помимо этого, данные, перед оправкой на соседние ядра проходят процедуру сжатия.

Возможность ядер “общаться” друг с другом напрямую, минуя “посредника” в виде шины системной памяти. Это критически важная особенность для имитации работы «сверточной нейронной сети» (Convolutional Neural Network – CNN). Вся вычислительная работа, необходимая для распознавания образов и речи, выполняется в Eyeriss локально, без необходимости обращения к сетевым ресурсам, что позволяет обеспечить возможность эффективного функционирования устройства даже в условиях отсутствия внешней сети.

Эффективное распределения вычислительных задач между ядрами в реализовано также в Eyeriss В своей локальной памяти ядро хранит не только данные обрабатываемые узлами, но и данные, описывающие сами узлы. С целью обеспечения максимальной производительности процесса обработки данных, а также для загрузки Eyeriss максимальным объемом данных из основной памяти алгоритм распределения данных обоих типов оптимизируется специально предназначенной для этой цели микросхемой в режиме реального времени с учетом особенностей актуальной нейронной сети.

В 2014 году выпущен чип с названием TrueNorth. Рис. 10.14, Это 28-нм чип производства компании Samsung с одним миллионом цифровых нейронов, с каждым из которых связано 256 цифровых синапсов (всего —

256 млн синапсов). Процессор TrueNorth содержит 5,4 млрд транзисторов. Нейрокомпьютерная система на базе TrueNorth содержит 16 процессоров TrueNorth, связанных подобием нейронных сетей. Для функционирования нейрокомпьютерной системы создан алгоритм, позволяющий организовать высокоэффективные и компактные самообучающиеся системы. Алгоритм, на основе сверточных нейронных сетей в процессоре TrueNorth классифицирует визуальные образы со скоростью между 1200 и 2600 кадров в секунду, с потреблением от 25 до 275 мВт., с эффективностью работы 6000 FPS/Вт. Фактически процессор IBM TrueNorth способен в реальном режиме времени обрабатывать кадры одновременно с 50-100 камер со скоростью 24 кадра в секунду в матрице 32 х 32 пикселя.Рис.10.3.11.

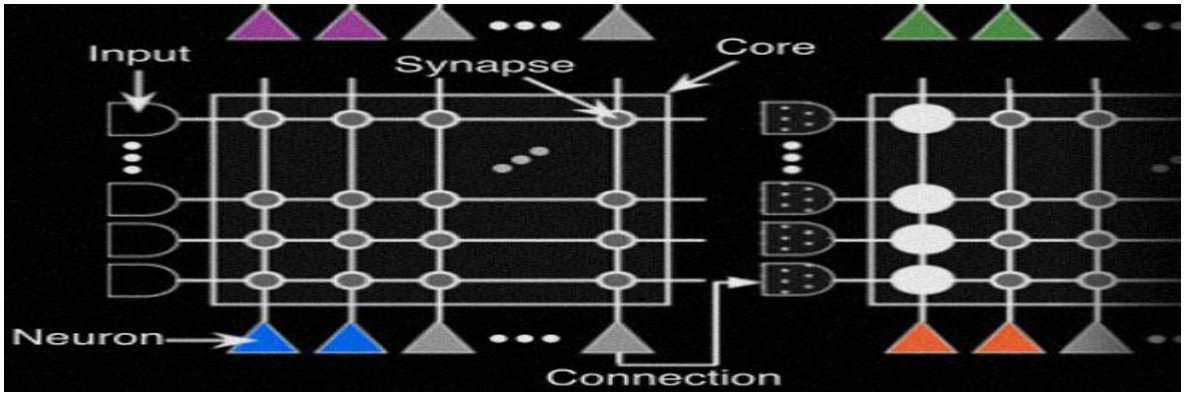


Рис. 10.3.11 - Фрагмент структуры процессора TrueNorth

Результат сравнения нейрокомпьютерных систем

Нейрокомпьютерные системы обработки данных оцениваются показателями быстродействия:

- CPS (connections per second). Число соединений (умножений с накоплением) в секунду.
- CPSPW (CPS per weight). CPPSPW = CPS/N_W, где N_W - число синапсов в нейроне. Фактически эта величина показывает число подсчетов выходов нейронов (количество вычислений функций активации) в секунду.
- CPPS. CPPS = CPSЧB_WЧB_S, где B_W, B_S - разрядность весов и

синапсов соответственно. Этот показатель учитывает разрядность и поэтому более правдоподобен для сравнения разнородных архитектурных решений.

- MMAC (multiplications and additions per second). Этот параметр обычно указывается для средств обработки сигналов (цифровых сигнальных процессоров и ПЛИС). Он не полностью соответствует показателю CPS.
- Время обработки. Этот показатель равен промежутку времени между поступлением входного сигнала и получением выходного сигнала нейросети.

Исходя из представленных параметров, особое внимание нужно обратить на ПЛИС, сигнальные процессоры и процессоры для каскадируемых архитектур. ПЛИС способны неплохо масштабироваться и достаточно дешевы, но затраты аппаратных ресурсов ПЛИС при проектировании достаточно велики и быстро растут с увеличением сложности сети.

Сигнальные процессоры масштабируются хуже, чем ПЛИС, но разработка нейронных сетей на них благодаря развитой технической поддержке со стороны производителей проста. При этом привязка к аппаратной части DSP не позволит с легкостью переносить нейронную сеть с одной элементной базы на другую. Сигнальные процессоры обладают не слишком хорошей масштабируемостью, что затрудняет построение больших многопроцессорных систем.

Систолические процессоры и процессоры с каскадной архитектурой обладают на порядок лучшей масштабируемостью, чем сигнальные, однако для них требуется достаточно много периферийных модулей. С увеличением числа систолических процессоров растут задержки в цепях прохождения сигнала.

Для нейросигнальных процессоров в целом характерно то же самое, что для сигнальных процессоров, отличия заключаются в более высокой

производительности за счет встроенного векторного сопроцессора и узкой специализации, в большинстве случаев ориентированной на конкретный вид нейронной сети.

Таким образом, выбор аппаратной реализации НКС определяется опытом разработчика, его предпочтениями, а также доступностью электронных компонентов.

Контрольные вопросы

1. В чем принципиальное отличие построения нейрокомпьютерных систем от многопроцессорных?
2. Каковы критерии выбора активационной функции?
3. В чем заключается понятие парадигмы нейросетевых реализаций?
4. В чем достоинства и недостатки обучения back propagation и машины Больцмана?
5. В чем различие АРТ-сети и сети Хемминга по применению и чем это обусловлено?
6. Какую роль играют синаптические коэффициенты при обучении нейронной сети?

Глава 11. ПЕРСПЕКТИВЫ РАЗВИТИЯ ПРЕОБРАЗОВАТЕЛЕЙ ИНФОРМАЦИИ

11.1 Перспективы развития компьютеров

Успехи, достигнутые при разработке современных компьютеров, впечатляют, но в то же время уже существуют ограничения по плотности размещения переключающих элементов, по производительности, объему памяти. Производительность компьютера пропорциональна количеству транзисторов на единице площади интегральной схемы. На процессорном чипе современного компьютера расположено до сотен миллионов транзисторов, и намного больше разместить уже вряд ли удастся, поскольку доведённые до совершенства технологии их производства достигли своего пика. Транзистор – это два электрода на кремниевой подложке, ток между которыми регулируется потенциалом, подавае-

мым на третий управляющий электрод – затвор. Критический элемент кремниевого транзистора, из-за которого нельзя сделать его намного меньше, – толщина изолирующего слоя оксида кремния между затвором и проводящим слоем. Современные технологии уже позволяют сделать его толщиной 0,13 микрон (130 нм), что соответствует примерно 1/1000 толщины человеческого волоса. В перспективе, лет через десять, может быть, удастся достичь толщины 0,09 микрон. Несмотря на то, что технологии производства изолирующего слоя оксида кремния совершенствуются и он становится тоньше, у него существует физический предел – не более 4–5 молекул (1,5–2 нм). В более тонких слоях начинаются неконтролируемые процессы туннелирования электронов и перегрева, которые нарушают работу транзисторов и вычислительной системы в целом. Более того, существует предел стабильной концентрации зарядов в проводящем слое, и само формирование интегральной схемы с меньшими размерами транзисторов невозможно на базе стандартной техники фотолитографии. В силу квантовых законов травление нельзя осуществить на меньшем масштабе, чем длина полуволны света, а уже сейчас используют жёсткое УФ и рентгеновское излучение, а также электронная литография, позволяющие достичь ширины проводящего слоя до 16 нм.. Такая проблема имеется и с носителями информации.

В настоящее время применяют магнитные и оптические носители памяти, которые основаны на принципе двумерной записи, что ограничивает объёмы записываемой информации. Стандартный диск CD-ROM диаметром 12 см может содержать примерно 0,5 гигабайт ($\sim 4 \cdot 10^9$ бит) данных. Теоретическая плотность оптической записи информации обратно пропорциональна квадрату длины волны, используемому для записи света, поэтому предел возможностей однослойного компакт-диска равен $3{,}5 \cdot 10^8$ бит/см^2 (для света с длиной волны 532 нм).

Третье ограничение – это применение двоичной арифметики. При этом увеличивается длина слова. Поэтому перспективней применение других систем счисления, но необходимо найти физические эффекты, позволяющие хранить много состояний.

Архитектура каждого компьютера включает три основных элемента: переключатели, память, соединяющие провода. Все

элементы в компьютерах будущего будут отличаться от их же аналогов в нынешних вычислительных устройствах.

Можно определить технологии построения и перспективы развития компьютеров будущего:

- оптические компьютеры;
- квантовые компьютеры;
- молекулярные компьютеры;
- нанокомпьютеры;
- биокомпьютеры.

Наиболее реальный вариант построения компьютера будущего – эволюционное использование комбинаций перспективных подходов: квантовых вычислений в структурах на основе молекулярных ключей с применением нанотехнологий, а также возбуждение этих активных элементов для считывания и записи информации оптическими технологиями. Бистабильные молекулы – переключатели будут управляться световыми и электрическими импульсами или электрохимическими реакциями. Память может работать на принципе «запоминания» оптических или магнитных эффектов, а проводниками могут стать нанотрубки или сопряжённые полимеры.

Результатом развития станут компактные быстродействующие и дешевые компьютеры. Появится возможность наделять любые промышленные продукты определенными интеллектуальными и коммуникационными способностями. К 2030 году может начаться распространение вживленных устройств с прямым доступом к нейронам. Ближе к середине столетия в мире киберпространства будут царить микро- и наноустройства (интеллектуальная пыль). К тому времени Интернет будет представлять собой отображение почти всего реального мира. Причем разрешение изображений, учитывая вероятные размеры емкости запоминающих устройств того времени, будет очень высоким. Надев на себя шлем виртуальной реальности, можно будет совершить полноценный круиз в любой уголок земного шара. Таким образом, грань между кибер- и реальным пространством начнет исчезать.

Исследования в области клетки приближают возможность замены тканей или органов, включая нейроны, которые раньше считались незаменимыми. Более того, клетки и ткани можно будет наделять способностями обработки и передачи данных. По-

добный контроль над живыми процессами дает надежду на увеличение продолжительности жизни: ученые не видят принципиальных препятствий к тому, чтобы люди жили по несколько сотен лет.

Контрольные вопросы

1. В чем причины радикального перехода на новые способы обработки информации?
2. Понятие квантовой суперпозиции?
3. Какие варианты представления состояний информации в квантовых компьютерах существуют?
4. Целесообразно и возможно ли применять другие системы счисления, кроме двоичной, в квантовых компьютерах?
5. Какова роль нанотехнологий в компьютерах будущего?
6. Какие достоинства оптической обработки информации?

12 ЗАКЛЮЧЕНИЕ

Происходит эволюция технических средств усиления деятельности человеческого мозга – компьютеров. Роль информации становится насущно необходимой для существования человека. Поэтому, взаимодействуя с энергетикой и технологией, информатика становится определяющей в жизни человека. Начиная от простейших элементов (ключей), используя аппарат дискретной математики, большие интегральные схемы и системы программирования, на основе законов диалектического материализма происходит переход к совершеннейшим системам обработки информации.

В настоящее время наблюдается следующая тенденция:

во-первых, переход от систем с простыми связями и сложными элементами (многопроцессорные структуры) к системам с простыми элементами (нейронами) и сложными связями (нейрокомпьютерные системы). Это – аналог человеческого мозга, и за такими системам будущее.

во-вторых, с уменьшением размеров активных элементов и повышением плотности их упаковки происходит переход от БИС к системам, в которых начинают действовать законы квантовой механики и реализуются принципы квантовых суперпозиций с гигантскими возможностями увеличения объемов обработки информации;

в третьих, переход к самоорганизации интеллектуальных систем с помощью микророботов на основе нано- и биотехнологий.

Термин «квантовый молекулярный биологический скачок» означает, что изменения происходят не только постепенно, но и скачками. Примерно 2020 год станет годом перехода от традиционных кремниевых полупроводников к более совершенным технологиям. Результатом станут намного компактные, быстродействующие и дешевые компьютеры. К 2030 году может начаться распространение вживленных устройств с прямым доступом к нейронам. Результаты исследований в биотехнологии дадут возможность замены тканей или органов, включая нейроны, которые раньше считались незаменимыми. Клетки и ткани будут обладать способностями обработки и передачи информации.

Ближе к середине столетия в мире киберпространства будут царить микро- и наноустройства, а второй половине века обрабатывающие мощности компьютеров превысят интеллектуальные способности человека.

Пока здравый смысл не приспособился к непонятному миру квантовой механики, генетики, молекулярным и биологическим представлениям. Это будущее кажется чуждым такому знакомому современному миру, основанному на электронике. Но придет время, и все это будет казаться естественным и необходимым для человека, да и сам человек возможно уже будет другим.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Майоров С.А., Новиков Г.И. Структура цифровых вычислительных машин. – Ленинград, 1970.

2. Коган Б.М. Электронные вычислительные машины и системы. – М., 1991.

3. Калабеков Б.А. Мамзелев И.А. Цифровые устройства и микропроцессорные системы. – М.: Радио и связь, 1987.

4. Калабеков Б.А. Микропроцессоры и их применение в системах передачи и обработки сигналов. – М.: Радио и связь, 1988.

5. Майоров С.А. Новиков Г.И. Структура цифровых вычислительных машин. – СПб.: Машиностроение, 1970.

6. Каган Б.М. Электронные вычислительные машины и системы. – М.: Энергоатомиздат, 1985.

7. Стрыгин В.В., Щарев Л.С. Основы вычислительной микропроцессорной техники и программирования. – М.: Высшая школа, 1989.

8. Журнал «Мир ПК».

9. Современные микропроцессоры / В.В. Корнеев, А.В. Киселев. – Нолидж, 1998.

10. Баранов С.И. Синтез микропрограммных автоматов. – Л.: Энергия, 1979.

11. Метлицкий Е.А., Каверзнев В.В. Системы параллельной памяти. Теория, проектирование, применение. – Л., 1989.