## 1.3 Понятия алгоритма

Понятие алгоритма является одним из основных понятий современной информатики. Термин алгоритм происходит от *algorithmi*– латинской формы написания имени выдающегося математика IX века аль-Хорезми, который сформулировал правила выполнения арифметических операций.

Вплоть до 30-х годов прошлого столетия понятие алгоритма носило сугубо интуитивный характер. Под алгоритмом понимали: *конечный набор точных и понятных предписаний (правил, инструкций, команд), позволяющих механически решать конкретную задачу из определенного класса однотипных задач*. Основными свойствами такого «интуитивного» понятия алгоритма являются:

**Массовость**. Означает, что алгоритм применим к целому классу задач, а при решении конкретной задачи из класса исходные данные могут меняться в определенных пределах.

**Детерминированность**. Процесс применения правил к исходным данным (путь решения задачи) определен однозначно.

**Дискретность**. Означает, что путь решения задачи определен в виде последовательности шагов – четко разделенных друг от друга предписаний. Только выполнив одно предписание, можно приступить к выполнению следующего.

**Результативность**. На каждом шаге процесса применения правил известно, что считать результатом этого процесса, а сам процесс должен закончиться за конечное число шагов.

**Понятность**. Означает, что алгоритм создается в расчете на определенного исполнителя, т.е. необходимо, чтобы он мог понять и выполнить каждый шаг предписания.

Для задач, имеющих положительное решение, этого определения достаточно. Другое дело, когда задача или класс задач не имеют решения. В этом случае требуется строго формализованное понятие алгоритма, чтобы иметь возможность доказать его отсутствие. Определение такого понятия алгоритма стала одной из центральных математических проблем. Решение было получено в середине 30-х годов в работах известных математиков, в двух эквивалентных формулировках: на основе особого класса арифметических функций, называемых *рекурсивными* (Д. Гильберт, К. Гедель, А. Черч, С. Клини), и на основе *абстрактных автоматов* (Э. Пост и А. Тьюринг). Появилось целое математическое направление – *теория алгоритмов,* в которой в основу определения алгоритма было поставлено *особое соответствие между словами в том или ином абстрактном алфавите* (А. Марков, Л. Калужнин). Теория алгоритмов оказалась тесно связанной не только с теоретической математикой (математической логикой, алгеброй, геометрией, анализом), но и с рядом областей лингвистики, экономики, физиологии мозга, философии, естествознания. Примером задачи

этой области может служить описание алгоритмов, реализуемых человеком в процессе умственной деятельности.

С появлением ЭВМ возникла область теории алгоритмов, тесно связанная с информатикой, которая стала теоретической основой таких ее составных частей, как теория программирования, построение алгоритмических языков и ЭВМ, разработка трансляторов, анализа алгоритмов с целью выбора наиболее рационального для решения на ЭВМ и т.д.

Практически любой сложный алгоритм можно представить собой комбинацией трех типов базовых структур: линейного, разветвляющегося и циклического.

**Линейный алгоритм** (следование) состоит из последовательности операций, выполняющихся только один раз в порядке их следования.


рис. 1.3.1 Линейная структура алгоритма

**Разветвляющийся алгоритм** (ветвление) обеспечивает выбор между двумя альтернативами. Выполняется проверка, а затем выбирается один из путей.

Подобная структура называется также «ЕСЛИ – ТО – ИНАЧЕ», или «развилка». Каждый из путей (ТО или ИНАЧЕ) ведет к общей точке слияния, так что выполнение программы продолжается независимо от того, какой путь был выбран.

Может оказаться, что для одного из результатов проверки ничего предпринимать не надо. В этом случае можно применять только один обрабатывающий блок.


рис. 1.3.2 Разветвляющаяся структура алгоритма


рис. 1.3.3 Структура «неполное ветвление»

*Циклический алгоритм*(Цикл) содержит некоторую последовательность операций, выполняемую многократно. Основной блок цикла, тело цикла, производит требуемые вычисления. Остальные блоки организуют циклический процесс: устанавливают начальные и новые значения данных, проверяют условия окончания или продолжения циклического процесса.

Различают два типа структур цикла: *цикл с параметром* или *с повторением и цикл с условием.* Циклический алгоритм позволяет компактно описать большое число одинаковых вычислений над разными данными для получения необходимого результата. *Циклы с параметром* используют тогда, когда количество повторов тела цикла заранее известно.

*Циклы с условием* используются тогда, когда число повторений заранее неизвестно, но задано условие окончания цикла. Причем, если условие окончания цикла проверяется перед выполнением тела цикла, то такие циклические структуры называют циклами *с предусловием* («Выполнять пока»), а если проверка условия происходит после выполнения тела цикла – циклами *с постусловием*(«Выполнять до тех пор пока не»).

### 1.4 Понятия понятие процесса и процессора

**Процессом** называется некоторая деятельность, выполняемая на процессоре. **Процессором** в широком смысле называется любое устройство в составе ЭВМ, способное автоматически выполнять определенную последовательность действий по программе, хранимой в памяти ЭВМ. Можно утверждать, что архитектура современной ЭВМ является многопроцессорной.

Процесс, выполняемый на центральном процессоре (внутренний процесс), представляет собой программу. При исполнении программ на центральном процессоре выделяются следующие допустимые состояния:

1. *Порождение* – подготавливаются условия для первого исполнения программы на процессоре;
2. *Активность* – программа исполняется на процессоре;
3. *Ожидание* – программа не исполняется на процессоре из-за отсутствия какого-либо требуемого ресурса;
4. *Готовность* – программа не исполняется, имея все необходимые ресурсы, кроме ЦП;
5. *Окончание* – нормальное или аварийное завершение программы, после чего ЦП и другие ресурсы ей не предоставляются.

Интервал времени между порождением и завершением называется *интервалом существования процесса*.

удобно изображать в виде *графа существования процесса*(рис. 1).

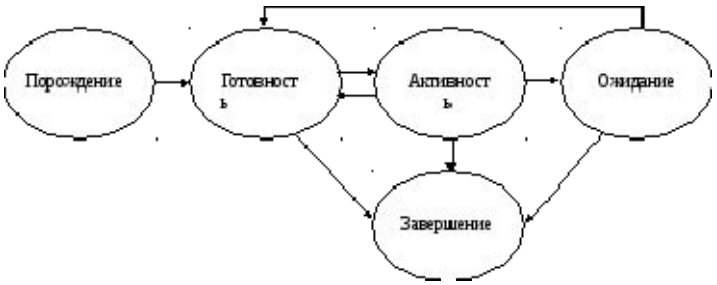


рис.1.4.1 Граф существования процесса

Рассмотрим классификацию процессов по ряду признаков (рис. 1.4.2).

1. Процессы, которые необходимо завершить до наступления конкретного момента времени, называются **процессами реального времени.**

2. Процессы, время существования которых не превышает времени допустимой реакции на запросы пользователя, называются **интерактивными**.

3. Остальные процессы называются **пакетными**.

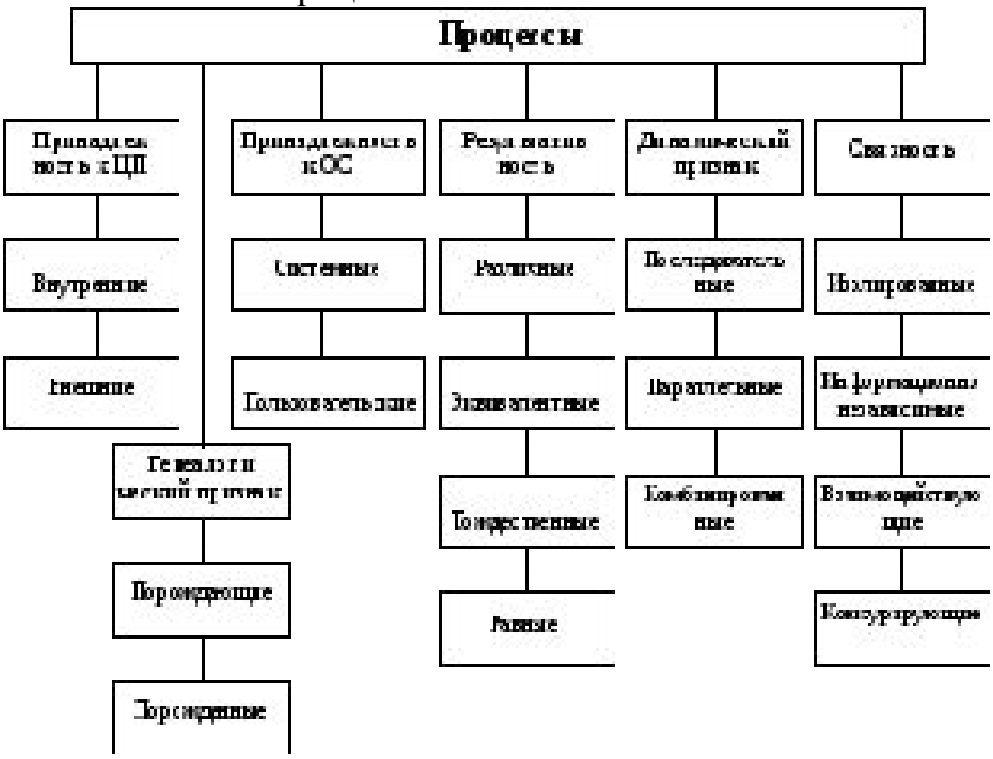


Рис.1.4.2 Схема классификации процессов

Процессы, развивающиеся на ЦП, называется **внутренними**, или **программными**. Процессы, развивающиеся на других процессорах, отличных от ЦП, называются **внешними** (процессы ввода-вывода).

Программные процессы, в свою очередь, делятся на **системные** (исполняется программа из состава ОС) и **пользовательские** (исполняется прикладная программа).

Два процесса, имеющие одинаковый конечный результат при одинаковых исходных данных называются **эквивалентными**. Если при этом используется одна и та же программа, то процессы называются **тождественными**. Если совпадают трассы процессов, то они **равные**. Во всех остальных случаях процессы **различные**.

Если интервалы существования двух процессов не пересекаются во времени, то это **последовательные** процессы. В противном случае – **параллельные**. Если на части рассматриваемого интервала процессы развиваются одновременно, то они **комбинированные**.

Если между процессами возникают связи всевозможных типов (функциональные, пространственно-временные, управляющие, информационные и т.д.), то процессы являются **взаимосвязанными**.

Процесс, по требованию которого образуется другой процесс, называется **порождающим**. Образованный процесс называется **порожденным**. Управляющий тип связи определяется отношением порождающего к порождаемому.

Если два процесса совместно используют некоторые ресурсы, но не обмениваются информацией, они называются **информационно-независимыми**. При наличии информационной связи, они являются взаимодействующими.

Если необходимо акцентировать внимание на совместном использовании двумя процессами одного и того же ресурса, их называют **конкурирующими**.

Допустимые отношения между взаимосвязанными процессами определяются тремя синхронизирующими правилами:

1. Отношение **предшествования**. Первый из двух процессов должен переходить в активное состояние всегда раньше второго.
2. Отношение **приоритетности**. Процесс, имеющий приоритет **P**, активизируется при соблюдении двух условий:
   - в состоянии готовности нет процессов с большим приоритетом;
   - процессор либо свободен, либо используется менее приоритетным, чем **P** процессом.
3. Отношение **взаимного исключения**. Если два процесса используют общий ресурс, допускающий только последовательное использование (**критический ресурс**), то действия одного процесса над этим ресурсом не должны выполняться одновременно с действиями другого процесса. Сами эти действия процесса над критическим ресурсом называются **критической областью**.

## 1.5 Обобщенная структура ЦВМ

Архитектурой компьютера считается его представление на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, системы адресации, организации памяти и т. д. Архитектура определяет принципы действия, информационные связи и взаимное соединение основных логических узлов компьютера: процессора, оперативного запоминаю-щего устройства (ОЗУ, ОП), внешних ЗУ и периферийных устройств. Общность архитектуры разных компьютеров обеспечивает их совместимость с точки зрения пользователя.

Структура компьютера — это совокупность его функциональных элементов и связей между ними. Элементами могут быть самые различные устройства — от основных логических узлов компьютера до простейших схем. Структура компьютера графически представляется в виде структурных схем, с помощью которых можно дать описание компьютера на любом уровне детализации.

В 40-х годах известный математик Джон фон Нейман пришёл к выводу, что вычислительные машины, в которых программы задаются буквально вручную, переключением рычагов и проводов, чрезмерно сложны для практического использования. Он создаёт концепцию, по которой исполняемые коды хранятся в памяти так же, как и обрабатываемые данные. Отделение процессорной части от накопителя данных и принципиально одинаковый подход к хранению программ и информации стали краеугольными камнями архитектуры фон Неймана. Эта компьютерная архитектура до сих пор является самой распространённой. Именно от первых устройств, построенных на архитектуре фон Неймана, отсчитываются поколения ЭВМ. Итак принципы фон Неймана:

1. **Принцип программного управления**. Из него следует, что программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности.

Выборка программы из памяти осуществляется с помощью счетчика команд. Этот регистр процессора последовательно увеличивает хранимый в нем адрес очередной команды на длину команды. Так как команды программы расположены в памяти друг за другом, то тем самым организуется выборка цепочки команд из последовательно расположенных ячеек памяти.

Если после выполнения команды следует перейти не к следующей, а к какой-то другой, используются команды условного или безусловного переходов (ветвления), которые заносят в счетчик команд номер ячейки памяти, содержащей следующую команду. Выборка команд из памяти прекращается после достижения и выполнения команды «стоп».

Таким образом, процессор исполняет программу автоматически, без вмешательства человека.

2. **Принцип однородности памяти**. Программы и данные хранятся в одной и той же памяти. Поэтому компьютер не различает, что хранится в данной ячейке памяти — число, текст или команда. Над командами можно выполнять такие же действия, как и над данными. Это открывает целый ряд возможностей. Например, программа в процессе своего выполнения также может подвергаться переработке, что позволяет задавать в самой программе правила получения некоторых ее частей (так в программе организуется выполнение циклов и подпрограмм). Более того, команды одной программы могут быть получены как результаты исполнения другой программы. На этом принципе основаны методы трансляции — перевода текста программы с языка программирования высокого уровня на язык конкретной машины.

3. **Принцип адресности**. Структурно основная память состоит из перенумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Отсюда следует возможность давать имена областям памяти, так, чтобы к запомненным в них значениям можно было впоследствии обращаться или менять их в процессе выполнения программ с использованием присвоенных имен.

Перечисленные принципы обеспечивают построение алгоритмически универсальных компьютеров простой архитектуры (рис. 1.1.5)


Рис. 1.5.1 Архитектура фон-неймановских (принстонских) компьютеров

Эти Компьютеры относятся к типу фон-неймановских или, как их еще иначе называют, **принстонских** и характеризуются использованием общей оперативной памяти для хранения программ, данных, а также для организации стека. Для обращения к этой памяти используется общая системная шина, по которой в процессор, по которой в процессор поступают и команда, и данные. Эта архитектура имеет ряд важных достоинств. Наличие общей памяти позволяет оперативно перераспределять ее объем для хранения отдельных массивов команд, данных и реализации стека в зависимости от решаемых задач. Таким образом, обеспечивается возможность более эффективного использования имеющегося объема оперативной памяти в каждом конкретном случае применения МП. Использование общей шины для передачи команд и данных значительно упрощает отладку, тестирование и текущий контроль функционирования системы, повышает ее надежность. Поэтом Принстонская архитектура в течение долгого времени доминировала в вычислительной технике.

Однако ей присущи и существенные недостатки. Основным из них является необходимость последовательной выборки команд и обрабатываемых данных по общей системной шине. При этом общая шина становится «узким местом», которое ограничивает производительность цифровой системы.

Данная проблема решается совершенствованием систем кэширования, что в свою очередь усложняет архитектуру систем и увеличивает риск возникновения побочных ошибок (например, в 2017 году были обнаружены уязвимости Meltdown и Spectre, присутствовавшие в современных процессорах в течение десятилетий, но не обнаруженные ранее из-за сложности современных вычислительных систем и, в частности, их взаимодействия с кэш-памятью).

Существуют и другие классы компьютеров, принципиально отличающиеся от фон-неймановских. Здесь, например, может не выполняться принцип программного управления, т. е. они могут работать без счетчика (регистра адреса) команд, указывающего на выполняемую команду программы. Для обращения к какой-либо переменной, хранящейся в памяти, этим компьютерам не обязательно давать ей имя. Такие компьютеры называются не-фон-неймановскими.

Одним из примеров не-фон-неймановскими компьютеров являются компьютеры, построенные на **гарвардской архитектуре**.

Так же в качестве недостатка архитектуры фон Неймана можно назвать возможность непреднамеренного нарушения работоспособности системы (программные ошибки) и преднамеренное уничтожение ее работы (вирусные атаки). В Гарвардской архитектуре принципиально различаются два вида памяти микропроцессора:

- Память программ (для хранения инструкций микропроцессора)
- Память данных (для временного хранения и обработки переменных)

В гарвардской архитектуре принципиально невозможно осуществить операцию записи в память программ, что исключает возможность случайного разрушения управляющей программы в случае ошибки программы при работе с данными или атаки третьих лиц. Кроме того, для работы с памятью программ и с памятью данных организуются отдельные шины обмена данными (системные шины), как это показано на структурной схеме, приведенной на рис. 1.5.2.

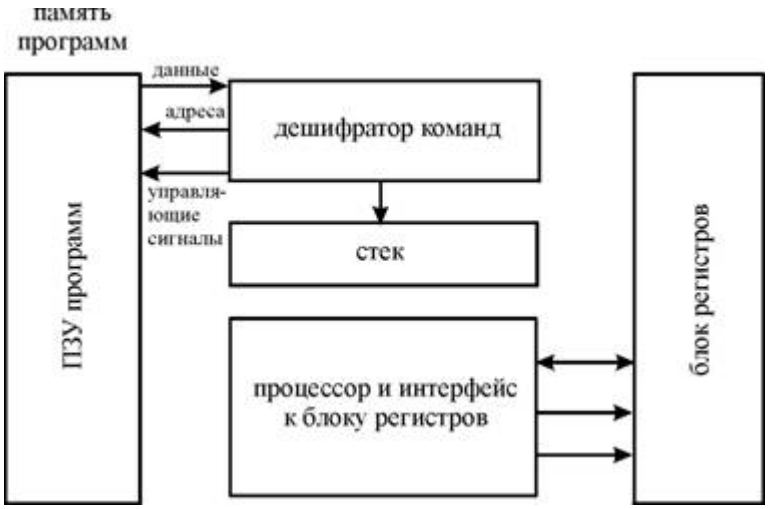


рис. 1.5.2 Структурная схема гарвардской архитектуры

Эти особенности определили области применения гарвардской архитектуры. Гарвардская архитектура применяется в микроконтролерах и в сигнальных процессорах , где требуется обеспечить высокую надёжность работы аппаратуры. В сигнальных процессорах Гарвардская архитектура дополняется применением трехшинного операционного блока микропроцессора. Трехшинная архитектура операционного блока позволяет совместить операции считывания двух операндов с записью результата выполнения команды в оперативную память микропроцессора. Это

значительно увеличивает производительность сигнального микропроцессора без увеличения его тактовой частоты.

В Гарвардской архитектуре характеристики устройств памяти программ и памяти данных не всегда выполняются одинаковыми. В памяти данных и команд могут различаться разрядность шины данных и распределение адресов памяти. Часто адресные пространства памяти программ и памяти данных выполняют различными. Это приводит к различию разрядности шины адреса для этих видов памяти. В микроконтроллерах память программ обычно реализуется в виде постоянного запоминающего устройства, а память данных — в виде ОЗУ. В сигнальных процессорах память программ вынуждены выполнять в виде ОЗУ. Это связано с более высоким быстродействием оперативного запоминающего устройства, однако при этом в процессе работы осуществляется защита от записи в эту область памяти.

Применение двух системных шин для обращения к памяти программ и памяти данных в гарвадской архитектуре имеет два недостатка — высокую стоимость и большое количество внешних выводов микропроцессора. При использованиии двух шин для передачи команд и данных, микропроцессор должен иметь почти вдвое больше выводов, так как шина адреса и шина данных составляют основную часть выводов микропроцессора. Для уменьшения количества выводов кристалла микропроцессора фирмы-производители микросхем объединили шины данных и шины адреса для внешней памяти данных и программ, оставив только различные сигналы управления (WR, RD, IRQ) а внутри микропроцессора сохранили классическую гарвардскую архитектуру. Такое решение получило название **модифицированная гарвардская архитектура**.

Модифицированная гарвардская структура применяется в современных микросхемах сигнальных процессоров. Ещё дальше по пути уменьшения стоимости кристалла за счет уменьшения площади, занимаемой системными шинами пошли производители однокристалльных ЭВМ — микроконтроллеров. В этих микросхемах применяется одна системная шина для передачи команд и данных (модифицированная гарвардская архитектура) и внутри кристалла.

В сигнальных процессорах для реализации таких алгоритмах как быстрое преобразование Фурье и цифровая фильтрация часто требуется еще большее количество внутренних шин. Обычно применяются две шины для чтения данных, одна шина для записи данных и одна шина для чтения инструкций. Подобная структура микропроцессора получила название расширенной гарвардской архитектуры.

Архитектура постнеймановских компьютеров

В настоящее время различают архитектуру больших универсальных компьютеров-мейнфреймов (наиболее типичными представителями являются компьютеры серий IBM 360/370 и их «потомков» ES9000) и архитектуру мини-, микро- и персональных компьютеров.

Особенностью универсальных компьютеров является *параллельная и асинхронная работа процессора и специализированных процессоров ввода-вывода - каналов ввода-вывода.* Каналы ввода-вывода полностью управляют всеми периферийными устройствами. Взаимодействие периферийных устройств с каналами и каналов с процессором обеспечивается системой прерывания. Если при выполнении программы возникает необходимость в работе периферийного устройства, то процессор инициализирует канал на выполнение данной операции, после чего продолжает выполнять основную программу. О завершении своей работы канал сообщает процессору прерыванием. Такая архитектура наиболее эффективная в понимании быстродействия, но требует больших аппаратных затрат (каналы ввода-вывода по своей архитектуре более сложные, чем процессор), сложного управления и имеют более низкую архитектурную надежность.

Универсальные компьютеры типа IBM 360/370 используются в режиме мультипрограммной обработки информации для многих пользователей и имеют широкий набор периферийных устройств. Типовая архитектура изображена на рис 1.5.3.

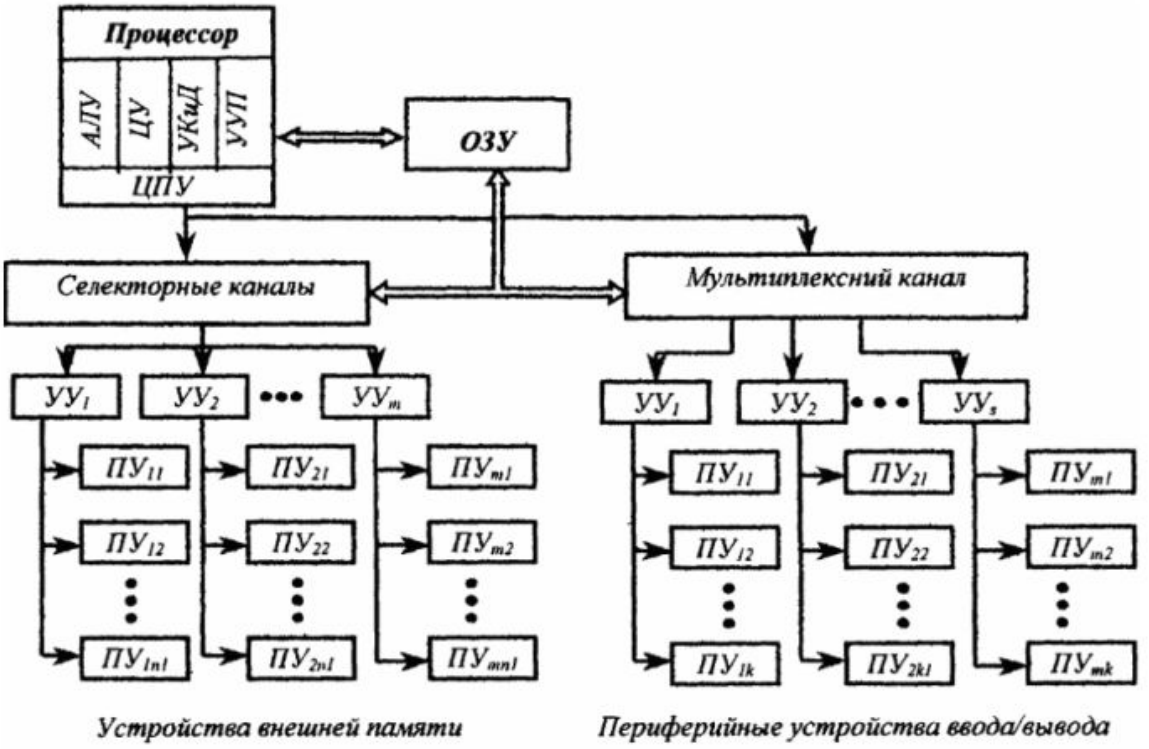


рис. 1.5.3 Типовая архитектура универсальных компьютеров IBM 360/370

Процессор имеет арифметико-логическое устройство (АЛУ), устройство центрального управления (ЦУ), устройство управления памятью (УУП) и устройство контроля и диагностики (УКиД). *Арифметико-логическое устройство* (АЛУ) выполняет арифметические и логические операции над двоичными и двоично-десятичными числами. *Устройство центрального управления* (ЦУ) обеспечивает микропрограммное управление всего процессора, обработку прерываний и отсчет времени. *Устройство управления памятью* (УУП) обеспечивает связь процессора и каналов ввода-

вывода с оперативным запоминающим устройством (ОЗУ), решения конфликтов при обращении к памяти и буферизацию информации, которая передается. *Устройство контроля и диагностики* (УКиД) обеспечивает текущий контроль функционирования компьютера при инициализации системы.

*Мультиплексный канал* является специализированным процессором ввода-вывода и обеспечивает ввод/вывод информации из медленно-действующих *периферийных устройств* (ПУ). Он работает в мультиплексном режиме, то есть после чтения/записи одного байта информации из одного периферийного устройства возможный обмен байтом информации с другим более приоритетным устройством (если канал получи запрос на обмен);

*Селекторные каналы* также являются специализированными процессорами ввода-вывода, но они предназначены для работы с периферийными быстродействующими устройствами, например, с устройствами внешней памяти, накопителями на дисках. Селекторный канал работает в селекторном режиме, то есть, если начался обмен информацией с одним устройством, то он не может быть прерван другим, даже более приоритетным, устройством.

Все периферийные устройства подключаются к каналу через свои *устройства управления* (УУ), что обеспечивают стандартное подключение разнотипных устройств к каналам.

Отличительной особенностью мини-, микро- и персональных компьютеров является простота и надежность в управлении. Поэтому все эти компьютеры имеют магистральную архитектуру, при которой процессор связан со всеми другими блоками компьютера (блоки ОЗУ, ПЗУ, периферийные устройства) путем единого интерфейса типа *общей шины* (рис. 1.5.4). При такой архитектуре в данный момент времени возможный обмен информацией только между двумя блоками, один из которых является *задатчнком* (обычно процессор или контроллер прямого доступа к памяти) и управляет процессом передачи информации по обшей шине, которая состоит из трех подшин - *шины адреса (А), шины данных (Д)* и *шины управления (У).* Общее управление системным интерфейсом выполняет контроллер шины.

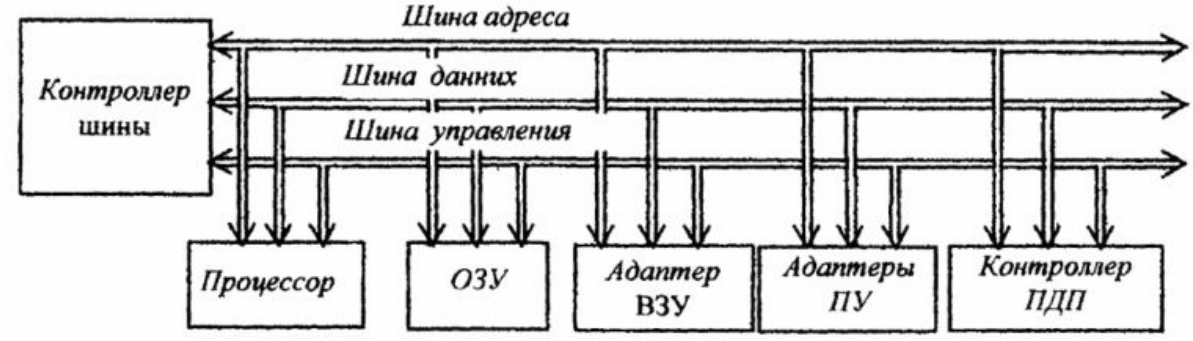


рис. 1.5.4 Типовая архитектура мини- и микро

Архитектура персональных компьютеров (ПК) берет начало от магистральной архитектуры, но в процессе своего развития архитектура ПК стала базироваться на *системы шин,* которая включает: *локальную шину*

*процессора* (L-local bus), *шину оперативной памяти* (M - Memory bus), *системную шину* (S - System bus), которая связывает работу всех модулей компьютера в единое целое, и *внешнюю (периферийную) шину (X*- external bus), связанную с периферийными модулями. Типовая схема персонального компьютера IBM PC показана на рис. 1.5.5.

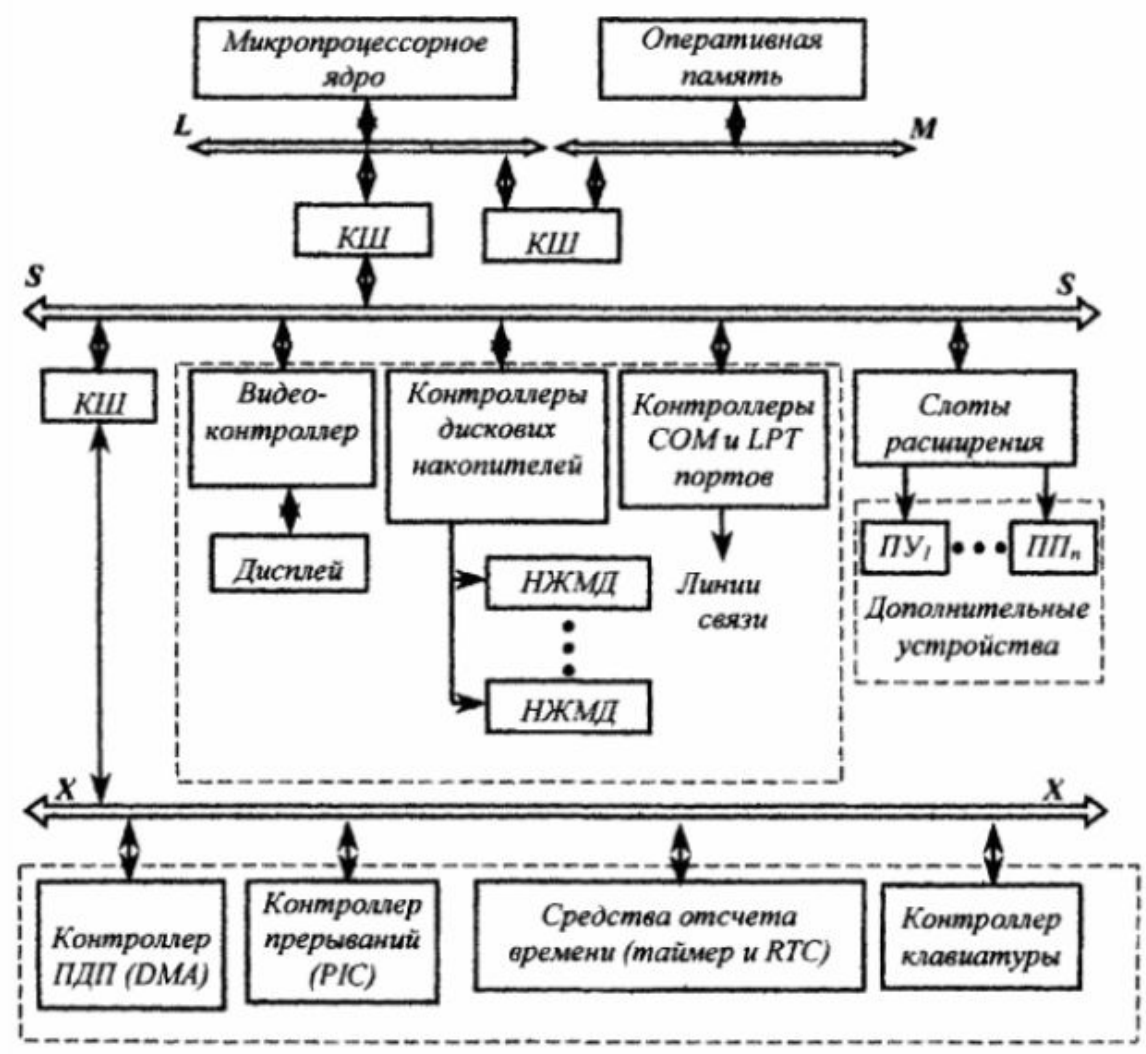


рис. 1.5.5 Типовая архитектура персонального компьютера IBM PC

Из рис. 1.5.5 видно, что взаимодействие шин обеспечивается контроллерами шин *(КШ),* которые включают шинные формирователи и буферные элементы.

Любая из этих шин является магистральной и состоит из оставляющих: адреса, данных и управления.

На рис. 1.5.5 приведен типовой набор модулей ПК. Здесь *микропроцессорное ядро* (МПЯ) включает модули и узлы, которые определяют работу центрального процессора и подключаются параллельно к его шинам. Непосредственно к системной шине S подключается внешняя периферия через слоты расширения и ее номенклатура может меняться. Периферия, которая подключенная к внешней шине *X,* и расположенная на материнской плате, обеспечивает минимально-необходимые условия функционирования ПК.