1. **Описание подходов к структуре ЭВМ. Архитектуры Принстонская и Гарвардская;**

Архитектуры Принстонская и Гарвардская - это два основных подхода к структуре ЭВМ, которые отличаются тем, как они обращаются с памятью и данными.

Принстонская архитектура:

В этой архитектуре инструкции и данные хранятся в одной и той же памяти.

Центральный процессор имеет доступ к этой общей памяти и может выполнять операции над инструкциями и данными из одного и того же источника.

Принстонская архитектура обычно используется в микроконтроллерах и микропроцессорах, где необходимо минимизировать размер и стоимость устройства.

Гарвардская архитектура:

В этой архитектуре инструкции и данные хранятся в разных памятях, что позволяет иметь отдельные пути доступа к инструкциям и данным.

Центральный процессор получает инструкции из памяти инструкций и данные из памяти данных.

Гарвардская архитектура обычно применяется в более сложных системах, таких как многозадачные операционные системы, сетевые устройства, цифровые сигнальные процессоры и т.д.

Выбор между Принстонской и Гарвардской архитектурой зависит от конкретных требований к системе, таких как скорость доступа к данным, требования к безопасности или необходимость параллельной обработки инструкций и данных.

1. **Принципы построения устройств памяти;**

В ЭВМ используется несколько типов запоминающих устройств, отличающихся принципом действия, характеристиками и назначением. Основными операциями в памяти являются запись и чтение. Обе эти операции называются «обращение к памяти». Важнейшими характеристиками ЗУ являются их емкость и быстродействие. Емкость памяти определяется максимальным количеством данных, которые в ней могут храниться. Быстродействие памяти определяется продолжительностью операции обращения, т.е. временем, затрачиваемым на поиск нужной единицы информации и ее чтение. В зависимости от реализуемых в памяти операций обращения различают:

А) память с произвольным обращением (RAM – Random Access Memory);

Б) память только для чтения информации (ROM – Read Only Memory), такая память носит название односторонней или постоянной.

По способу организации доступа различают ЗУ:

1) с непосредственным (произвольным) доступом;

2) с прямым (циклическим) доступом; 26

3) с последовательным доступом

1. **Адресная, ассоциативная и стековая организация памяти;**

ЗУ с произвольным обращением обычно содержит множество одинаковых запоминающих элементов, образующих запоминающий массив (ЗМ). ЗМ разделен на отдельные ячейки, число разрядов в которых равно ширине выборки памяти. Способ организации памяти зависит от метода размещения и поиска информации в ЗУ. По этому признаку различают адресную, ассоциативную и стековую память.

Адресная память

В такой памяти размещение и поиск информации в ЗУ основаны на использовании адреса байта или слова.

Ассоциативная память

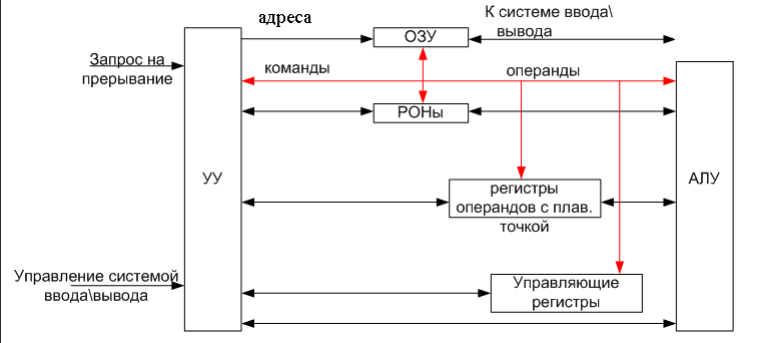
В памяти этого типа поиск нужной информации производится не по адресу, а по содержанию самой информации (т.е. по ассоциативному признаку).

Стековая память

Стековая память также как и ассоциативная, является безадресной, она представляет собой совокупность ячеек, образующих одномерный массив, в котором соседние ячейки связаны друг с другом разрядными цепями передачи слов.

1. **Логическая структура процессора;**

Рассмотрим логическую структуру процессора ЭВМ общего назначения на примере ЭВМ типа IBM 370.



Обобщенная структурная схема процессора содержит АЛУ, УУ и регистры. Процессор может обрабатывать числа, представленные в двоичном коде; числа с плавающей точкой фиксируемой длины; десятичные числа.

Выполняемые процессором команды образуют 5 классов:

1-команды управления системой;

2-общие команды;

3-команды для выполнения действий над числами с плавающей точкой;

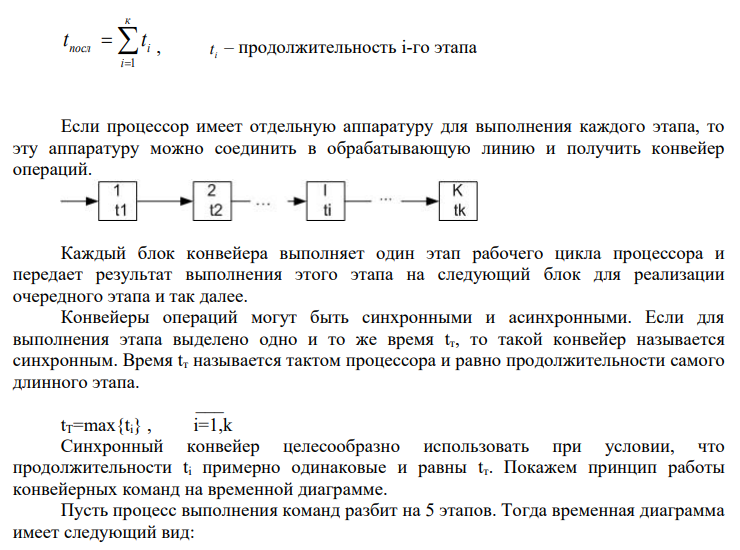
4-команды для выполнения действий над десятичными числами;

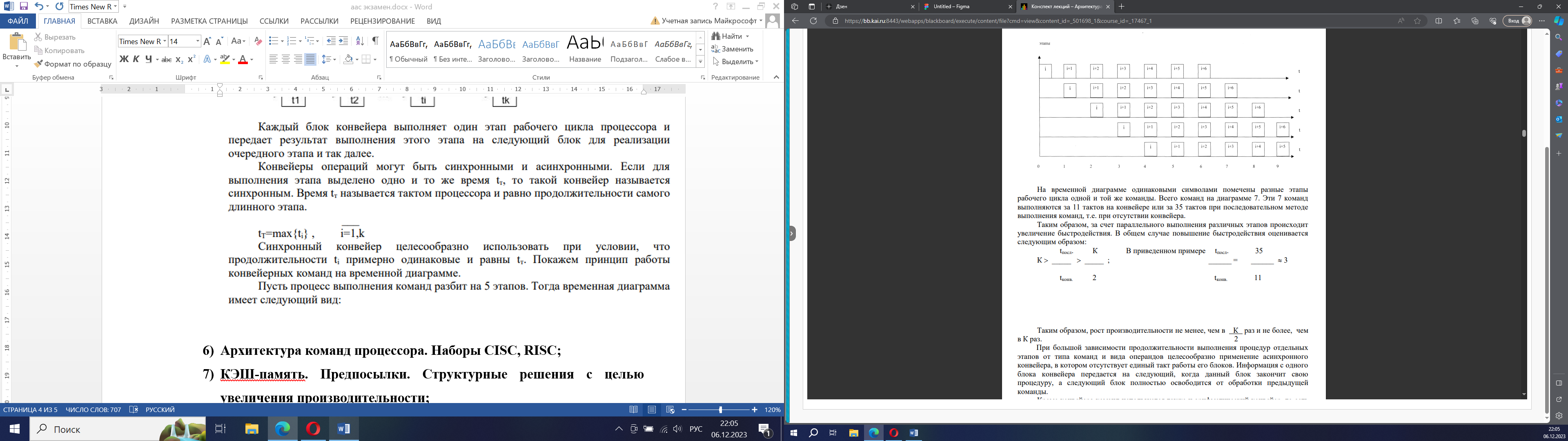
5-команды ввода/вывода.

1. **Методы повышения производительности работы процессора;**

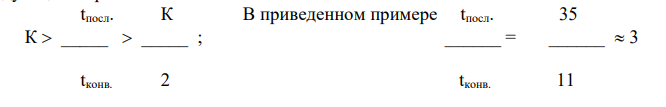
1.Конвейеризация (конвейер операций).

Повысить производительность процессора можно за счет параллельного выполнения отдельных этапов рабочего цикла команд. Пусть рабочий цикл процессора состоит из К этапов. Тогда при последовательном выполнении этапов продолжительность всех процедур рабочего цикла команды равна:





На временной диаграмме одинаковыми символами помечены разные этапы рабочего цикла одной и той же команды. Всего команд на диаграмме 7. Эти 7 команд выполняются за 11 тактов на конвейере или за 35 тактов при последовательном методе выполнения команд, т.е. при отсутствии конвейера. Таким образом, за счет параллельного выполнения различных этапов происходит увеличение быстродействия. В общем случае повышение быстродействия оценивается следующим образом:



Таким образом, рост производительности не менее, чем в К раз и не более, чем в К раз. 2 При большой зависимости продолжительности выполнения процедур отдельных этапов от типа команд и вида операндов целесообразно применение асинхронного конвейера, в котором отсутствует единый такт работы его блоков. Информация с одного блока конвейера передается на следующий, когда данный блок закончит свою процедуру, а следующий блок полностью освободится от обработки предыдущей команды. Кроме конвейера команд используется также и арифметический конвейер, то есть само АЛУ также можно строить в виде конвейера. Конвейерное АЛУ часто называют магистралью. Арифметический конвейер в основном используется в специализированных устройствах с ограниченным набором алгоритмов обработки данных.

1. **Архитектура команд процессора. Наборы CISC, RISC;**

Архитектура команд процессора относится к набору инструкций, которые процессор может выполнить. Существуют две основные архитектуры команд процессора: CISC (Complex Instruction Set Computing) и RISC (Reduced Instruction Set Computing).

CISC (Complex Instruction Set Computing):

CISC-процессоры предоставляют широкий набор сложных инструкций, способных выполнять разнообразные операции.

Инструкции CISC-процессоров могут выполнять несколько операций за один такт процессора, что делает их более гибкими и универсальными.

Примеры CISC-архитектур - Intel x86 и AMD64.

RISC (Reduced Instruction Set Computing):

RISC-процессоры используют более простые и базовые инструкции, предназначенные для выполнения конкретных операций.

Основная идея RISC заключается в том, что простые инструкции могут быть выполнены быстрее и эффективнее, чем более сложные инструкции.

RISC-процессоры обычно имеют фиксированную длину инструкций, что упрощает их декодирование и выполнение.

Примеры RISC-архитектур - ARM, MIPS, PowerPC.

Обе архитектуры имеют свои преимущества и недостатки. CISC-архитектуры обеспечивают более высокий уровень абстракции и универсальность, в то время как RISC-архитектуры обладают повышенной производительностью и эффективностью, основанными на более простых инструкциях. В настоящее время существует также гибридные архитектуры, пытающиеся объединить преимущества обеих концепций.

1. **КЭШ-память. Предпосылки. Структурные решения с целью увеличения производительности;**

КЭШ-память (или кэш) является важной частью архитектуры компьютерных систем, предназначенной для увеличения производительности путем ускорения доступа к данным. Предпосылки для использования кэш-памяти включают ограниченное время доступа к основной памяти (RAM) и принципы локальности данных, которые вносят кластеры адресов, обращающихся к памяти.

Структурные решения, направленные на увеличение производительности кэш-памяти, включают в себя:

Иерархия кэш-памяти:

КЭШ-память может быть организована в иерархическую структуру, включающую уровни с разной емкостью и скоростью доступа. Например, L1 (уровень 1), L2 (уровень 2), L3 (уровень 3) кэши. Это позволяет обеспечить быстрый доступ к наиболее часто используемым данным и уменьшить нагрузку на медленной основной памяти.

Умные алгоритмы замещения:

Когда кэш-память заполняется, а новые данные требуют записи, алгоритм замещения определяет, какие данные удалять из кэша. Умные алгоритмы, такие как LRU (Least Recently Used) или LFU (Least Frequently Used), могут помочь уменьшить количество кэш-промахов и повысить эффективность кэша.

Предварительная загрузка (Prefetching):

Механизмы предварительной загрузки могут использоваться для эффективного заполнения кэш-памяти данными, которые могут быть запрошены приложениями в будущем на основе паттернов доступа к данным.

Технологии множества путей (Ways):

Количество путей в кэше определяет, сколько блоков данных может быть хранено в конкретной части кэша, что влияет на его эффективность. Технологии сочетания или полностью ассоциативные кэши могут помочь увеличить пропускную способность и уменьшить конфликты.

Эти структурные решения направлены на повышение эффективности кэш-памяти за счет обеспечения быстрого доступа к данным, увеличения технологических возможностей и уменьшения вероятности кэш-промахов.

1. **Архитектура КЭШ-памяти;**

Архитектура КЭШ-памяти определяется тем, каким образом достаточно большая оперативная память отображается на сравнительно небольшой КЭШ.

Существует три разновидности отображения:

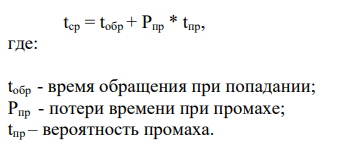
1) КЭШ память с прямым отображением.

2) Полностью ассоциативная КЭШ -память.

3) Частично ассоциативная КЭШ-память.

1. **Алгоритм замещения строк в КЭШ-памяти. Методы записи в КЭШ-память;**

Среднее время доступа к КЭШ памяти можно определить следующим образом:



Методы

Возможны два способа записи в КЭШ память:

1)метод сквозной записи;

2)метод обратной записи.

Первый метод предполагает наличие двух копий данных: одной в КЭШ памяти, а другой в ОЗУ. Запись выполняется одновременно и в КЭШ и в ОЗУ. В результате системная шина и процессор работают с большой нагрузкой, поскольку на каждую операцию изменения данных приходится две операции записи. Метод сквозной записи имеет преимущество в том, что ОЗУ всегда имеет свежую копию данных, что важно в мультипроцессорных системах и при организации ввода-вывода данных. Кроме того сквозная запись имеет простую схемную реализацию. Такой метод записи использовался в 80486 микропроцессоре фирмы Intel.

При использовании метода обратной записи цикл записи происходит только в КЭШ памяти, если в КЭШе находится строка, к которой идет обращение (КЭШ попадание). Если адресуемой строки в КЭШ нет, то информация записывается сразу в ОЗУ. При КЭШ попадании запись в ОЗУ происходит только при замещении строки КЭШа. Для сокращения частоты копирования строк запись в ОЗУ происходит только в том случае, если замещаемая строка КЭШа была модифицирована, то есть изменена. Для определения факта изменения строки с каждой строкой КЭШа связывают так называемый «бит модификации». Этот бит показывает, была ли изменена строка в КЭШ памяти или нет. Если строка в КЭШ не модифицирована, то обратное копирование отменяется, поскольку ОЗУ содержит туже самую информацию, что и КЭШ память. Преимущество данного метода заключается в том, что запись выполняется со скоростью КЭШ памяти. А несколько записей на одну и туже строку КЭШа требует только одной записи – в ОЗУ, что снижает загрузку системной шины. Запись с обратным копированием используется в Pentium.

1. **Система ввода\вывода. Предпосылки. Способы подключения системы ввода\вывода к ЦП;**

Система ввода/вывода призвана обеспечить обмен информацией между ядром ВМ и разнообразными внешними устройствами (ВУ). Технические и программные средства СВВ несут ответственность за физическое и логическое сопряжение ядра вычислительной машины и ВУ. В процессе эволюции вычислительных машин системам ввода/вывода по сравнению с прочими элементами архитектуры уделялось несколько меньшее внимание. Косвенным подтверждением этого можно считать, например, то, что многие программы контроля производительности (бенчмарки) вообще не учитывают влияние операций ввода/вывода (В/ВЫВ) на эффективность ВМ. Следствием подобного отношения стал существенный разрыв в производительности процессора и памяти, с одной стороны, и скоростью ввода/вывода — с другой.

Технически система ввода/вывода в рамках ВМ реализуется комплексом модулей ввода/вывода (МВВ). Модуль ввода/вывода выполняет сопряжение ВУ с ядром ВМ и различные коммуникационные операции между ними.

Две основныe функции МВВ:

- обеспечение интерфейса с ЦП и памятью («большой» интерфейс);

- обеспечение интерфейса с одним или несколькими периферийными устройствами («малый» интерфейс).

Анализируя архитектуру известных ВМ, можно выделить три основных способа подключения СВВ к ядру процессора.

1. **Адресное пространство системы ввода\вывода. Достоинства и недостатки каждого;**

Как и обращение к памяти, операции ввода/вывода также предполагают наличие некоторой системы адресации, позволяющей выбрать один из модулей СВВ, а также одно из подключенных к нему внешних устройств. Адрес модуля и ВУ является составной частью соответствующей команды, в то время как расположение данных на внешнем устройстве определяется пересылаемой на ВУ информацией. Адресное пространство ввода/вывода может быть совмещено с адресным пространством памяти или быть выделенным.

Достоинства совмещенного адресного пространства:

- Расширение набора команд для обращения к внешним устройствам, что позволяет сократить длину программы и повысить быстродействие;

- Значительное увеличение количества подключаемых внешних устройств;

- возможность внепроцессорного обмена данными между внешними устройствами, если в системе команд есть команды пересылки между ячейками памяти;

- возможность обмена информацией не только с аккумулятором, но и с любым регистром центрального процессора.

Недостатки совмещенного адресного пространства:

- сокращение области адресного пространства памяти;

- усложнение декодирующих схем адресов в СВВ;

- трудности распознавания операций передачи информации при вводе/выв среди других операций.

1. **Внешние устройства. Определение. Категории ВУ. Структура ВУ;**

Внешние устройства компьютера – это устройства, которые подключаются к компьютеру для расширения его функциональности и возможностей.

Внешнее устройство, подключенное к МВВ, обычно называют периферийным устройством (ПУ).

Все множество ПУ можно свести к трем категориям :

- для общения с пользователем;

- для общения с ВМ;

- для СВЯЗИ с удаленными устройствами.



Обобщенная структура ВУ показана на рис. 3. Интерфейс с МВВ реализуется в виде сигналов управления, состояния и данных. Данные представлены совокупностью битов, которые должны быть переданы в модуль ввода/вывода или получены из него. Сигналы управления определяют функцию, которая должна быть выполнена внешним устройством. Это может быть стандартная для всех устройств функция — посылка данных в МВВ или получение данных из него, либо специфичная для данного типа ВУ функция, такая, например, как позиционирование головки магнитного диска или перемотка магнитной ленты. Сигналы состояния характеризуют текущее состояние устройства, в частности включено ли ВУ и готово ли оно к передаче данных.

1. **Модули ввода\вывода информации. Основные функции и их характеристики;**

Функции модуля Модуль ввода/вывода в составе вычислительной машины отвечает за управление одним или несколькими ВУ и за обмен данными между этими устройствами с одной стороны, и основной памятью или регистрами ЦП — с другой.

Основные функции МВВ можно сформулировать следующим образом:

- локализация данных;

- управление и синхронизация;

- обмен информацией;

- буферизация данных;

- обнаружение ошибок.

Локализация данных

Под локализацией данных будем понимать возможность обращения к одному из ВУ, а также адресации данных на нем.

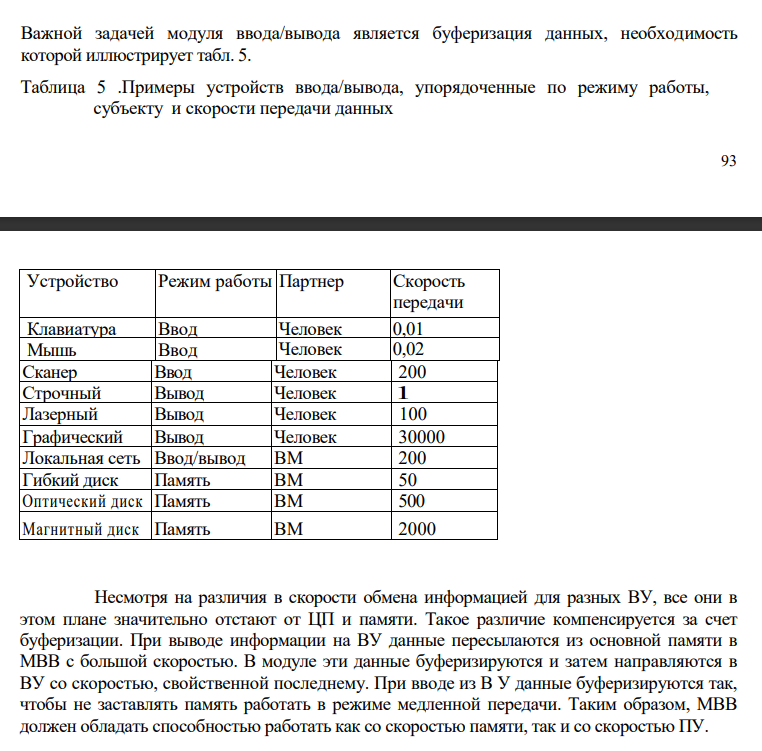
Управление и синхронизация

Функция управления и синхронизации заключается в том, что МВВ должен координировать перемещение данных между внутренними ресурсами ВМ и внешними устройствами. При разработке системы управления и синхронизации модуля ввода/вывода необходимо учитывать целый ряд факторов.

Обмен информацией

Основной функцией МВВ является обеспечение обмена информацией. Со стороны «большого» интерфейса — это обмен с ЦП, а со стороны «малого» интерфейса обмен с ВУ.

Буферизация



Обнаружение ошибок

Еще одной из важнейших функций МВВ является обнаружение ошибок, возникающих в процессе ввода/вывода. Центральный процессор следует оповещать о каждом случае обнаружения ошибки.

Причинами возникновения последних бывают самые разнообразные факторы, которые в первом приближении можно свести к следующим группам:

- воздействие внешней среды; - старение элементной базы;

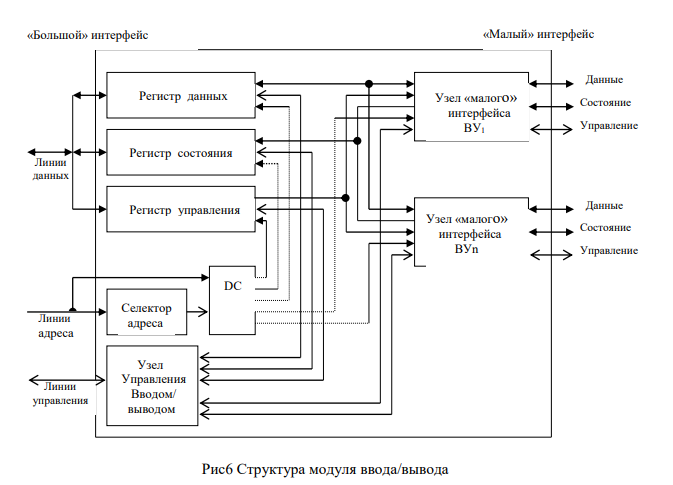
- системное программное обеспечение;

- Пользовательское программное обеспечение. Из наиболее «активных» факторов окружения ВМ следует выделить: - загрязнение и влагу;

- повышенную или пониженную температуру окружающей среды;

- электромагнитное облучение; - скачки напряжения питания.

1. **Структура модуля ввода\вывода информации. Зарисовать с кратким пояснением элементов;**



Данные, передаваемые в модуль и из него, буферизируются в регистре данных. Буферизация позволяет компенсировать различие в быстродействии ядра ВМ и внешних устройств. Разрядность регистра, как правило, совпадает с шириной шины данных со стороны «большого» интерфейса (2, 4 или 8 байт). В свою очередь, большинство ВУ ориентировано на побайтовый обмен информацией. Побайтовая пересылка информации по «широкой» системной шине — крайне неэффективное решение, поэтому со стороны «малого» интерфейса регистр данных часто дополняют узлом упаковки/распаковки (на схеме не показан). Этот узел при вводе обеспечивает последовательное побайтовое заполнение регистра данных (упаковку), при выводе — последовательную побайтовую выдачу содержимого регистра на ВУ (распаковку). В результате при обмене данными через «большой» интерфейс занята вся ширина шины данных. В МВВ, рассчитанных на работу с большим числом ВУ, могут входить несколько регистров данных, что позволяет независимо хранить данные каждого из внешних устройств. Помимо регистра данных в составе МВВ имеются также регистр управления и регистр состояния (либо совмещенный регистр управления/состояния). В регистре управления (РУ) фиксируются поступившие из ЦП команды управления модулем или подключенными к нему внешними устройствами. Отдельные разряды регистра могут представлять такие команды, как очистка регистров МВВ, сброс ВУ, начало чтения, начало записи и т. п. В сложных МВВ присутствует несколько регистров управления, например регистр управляющих сигналов для модуля в целом и отдельные РУ для каждого из ВУ. Регистр состояния (PC) служит для хранения битов состояния MBВ и подключенных к нему ВУ. Содержимое определенного разряда регистра может характеризовать, например, готовность устройства ввода к приему очередной порции данных, занятость устройства вывода или нахождение ВУ в автономном режиме (offine). В МВВ не исключается наличие и более одного регистра состояния. Процедура ввода/вывода предполагает возможность работы с каждым регистром МВВ или внешним устройством по отдельности. Такая возможность обеспечивается системой адресации. Каждому модулю в адресном пространстве ввода/ вывода (совмещенном или раздельном) выделяется уникальный набор адресов, количество адресов в котором зависит от числа адресуемых элементов. Поступивший из ЦП адрес с помощью селектора адреса проверяется на принадлежность к диапазону, выделенному Линии управления Узел Управления Вводом/ выводом Линии данных Регистр данных данных Регистр управления данных Регистр состояния данных Узел «малого» интерфейса ВУn Данные Состояние Управление Узел «малого» интерфейса ВУ1 Данные Состояние Управление DC Линии адреса Селектор адреса «Большой» интерфейс «Малый» интерфейс Рис6 Структура модуля ввода/вывода 97 данному МВВ. В случае подтверждения дешифратор DC выполняет раскодирование адреса, разрешая работу с соответствующим регистром модуля или В У. Узел управления вводом/выводом по сути играет роль местного устройства управления МВВ. На него возлагаются две задачи: обеспечение взаимодействия с ЦП и координация работы всех составляющих МВВ. Связь с ЦП реализуется посредством линий управления, по которым из ЦП в модуль поступают сигналы, служащие для синхронизации операций ввода и вывода. В обратном направлении передаются сигналы, информирующие о происходящих в модуле событиях, например сигналы прерывания. Часть линий управления может задействоваться модулем для арбитража. Вторая функция узла управления реализуется с помощью внутренних сигналов управления. Со стороны «малого» интерфейса МВВ обеспечивает подключение внешних устройств и взаимодействие с ними. Эта часть МВВ более унифицирована, поскольку внешние устройства всегда подгоняются под один из стандартных протоколов. Каждое из внешних устройств «обслуживается» своим узлом «малого» интерфейса, который реализует принятый для данного ВУ стандартный протокол взаимодействия. При управлении широким спектром ВУ модуль должен по возможности освобождать ЦП от знания деталей конкретных ВУ, так чтобы ЦП мог управлять любым устройством с помощью простых команд чтения и записи. МВВ при этом берет на себя задачи синхронизации, согласования форматов данных и т. п. Модуль ввода/вывода, который берет на себя детальное управление ВУ и общается с ЦП только с помощью команд высокого уровня, часто называют каналом ввода/вывода или процессором ввода/вывода. Наиболее примитивный МВВ, требующий детального управления со стороны ЦП, называют контроллером ввода/ вывода или контроллером устройства. Как правило, контроллеры ввода/выв типичны для микроЭВМ, а каналы ввода/вывода — для универсальных ВМ.

1. **Архитектура аппаратных средств. Понятие. Основные блоки аппаратной среды и их характеристика;**

Архитектура аппаратных средств относится к организации и структуре аппаратных компонентов в компьютерной системе. Она определяет, как аппаратные средства взаимодействуют между собой, чтобы обеспечить выполнение вычислений, обработку данных и выполнение других задач. Основные блоки аппаратной среды включают:

Процессор:

Центральный процессор (CPU) является "мозгом" компьютерной системы, который выполняет инструкции и управляет работой всех других компонентов.

Память:

Оперативная память (RAM) используется для временного хранения данных и инструкций, к которым процессор обращается во время выполнения программ.

Постоянное хранилище (например, жёсткий диск) используется для долговременного хранения данных и программ.

Ввод/вывод (I/O):

Вводно-выводные устройства, такие как клавиатура, мышь, монитор, принтер, сетевые адаптеры, обеспечивают взаимодействие компьютера с внешними устройствами.

Шина данных:

Шина данных обеспечивает передачу данных между различными компонентами компьютерной системы, такими как процессор, память и внешние устройства.

Шина управления:

Шина управления обеспечивает передачу команд и сигналов управления между различными компонентами, координируя их работу.

Чипсет:

Чипсет представляет собой комплект микросхем, которые управляют передачей данных между различными компонентами компьютера, такими как процессор, память и устройства ввода/вывода.

Характеристики аппаратных средств включают в себя такие аспекты, как производительность, скорость работы, объем памяти, интерфейсы ввода/вывода, энергопотребление, совместимость и другие технические параметры, которые определяют их возможности и способности к выполнению конкретных задач.

1. **«Теория» отрицательных чисел. Как представляются в системе, каким способом вычисляются. Пример;**

В компьютерных системах отрицательные числа обычно представляются с использованием двоичной системы счисления и метода дополнительного кода.

Дополнительный код:

В методе дополнительного кода для представления отрицательных чисел, сначала представляется модуль значения в двоичном коде, а затем полученный результат инвертируется (заменяются 0 на 1 и наоборот), а к полученному значению прибавляется 1.

Например, чтобы представить число -5 в 8-битном двоичном коде:

Шаг 1: Представляем модуль числа 5 в двоичной системе - 00000101.

Шаг 2: Инвертируем полученное значение - 11111010.

Шаг 3: Добавляем 1 к инвертированному значению - получаем дополнительный код числа -5: 11111011.

Вычисление с отрицательными числами:

При выполнении операций с отрицательными числами, компьютеры используют аппаратные средства для преобразования их в дополнительный код, после чего выполняют арифметические операции в стандартном двоичном представлении.

При сложении и вычитании отрицательных чисел, компьютеры выполняют арифметику на основе дополнительного кода, что позволяет избежать дополнительных сложных шагов для обработки отрицательных чисел.

Пример:

Давайте сложим числа -5 и -3, представленные в дополнительном коде:

-5: 11111011

-3: 11111101

-8: 11110100

После сложения двух чисел в дополнительном коде получаем результат -8. Далее, если требуется, компьютер может выполнить обратное преобразование этого значения обратно в десятичную систему для отображения пользователю.

1. **Язык программирования Ассемблер. Синтаксис. Форматы операндов. Примеры;**

Язык программирования Ассемблер (одинаковый) - это низкоуровневый язык программирования, который позволяет программистам писать инструкции, специфичные для архитектуры процессора. Синтаксис языка ассемблера может отличаться в зависимости от архитектуры процессора, но в основе у него общие принципы, какие-то части языка могут быть специфичны для конкретной архитектуры. Вот общие принципы синтаксиса и форматов операндов в языке ассемблера на примере архитектуры x86 (Intel и AMD процессоры):

Операнды:

Регистры процессора, например, EAX, EBX, ECX и др.

Память, обозначаемая адресами памяти или переменными.

Непосредственные значения (Immediate values), например, константы или литералы.

Смещения относительно регистров или адресов памяти.

Команды и их синтаксис:

MOV(op1, op2)-перемещение или пересылка данных.

CMP(op1, op2) –сравнение. От op1 отнимается op2, если ноль, то они равны, если не ноль, то они не равны.

Команды сравнения; JE, JA, JMP(безусловный переход), JZ, JNZ, JNC

MUL(op1) и IMUL(op1, op2) умножение

PUSH(источник или регистр) POP(op1)

IN(приёмник, адрес источника) ввод вывод, чтение с порта

SUB(op1, op2)-разность

INC и DEC (INC+1, DEC-1)

ADD(op1, op2)-сложение

LOOP(метка)- переход на метку

OR(op1, op2)- побитовое логическое или

CLR(op1) и CLRB(op1)

Примеры:

MOV EAX, 5 ; переместить значение 5 в регистр EAX

MOV ECX, [EBX] ; переместить значение, находящееся по адресу, содержащемуся в регистре EBX, в регистр ECX

ADD EAX, EBX ; прибавить значение регистра EBX к регистру EAX

ADD ECX, 10 ; прибавить значение 10 к регистру ECX

SUB EAX, EBX ; вычесть значение регистра EBX из регистра EAX

SUB ECX, 5 ; вычесть значение 5 из регистра ECX

1. **Характеристика процессора PDP-11. Форматы команда;**

Характеристики процессора PDP-11:

Архитектура:

Процессор PDP-11 имел 16-битную архитектуру с адресным пространством, состоящим из 16-разрядных адресов. Это означает, что он мог обрабатывать данные по 16 бит за раз.

Регистры:

PDP-11 имел однородный набор из восьми 16-битных регистров общего назначения (R0-R7), которые использовались для хранения данных, адресов и результата выполнения операций.

Команды:

Форматы команд в PDP-11 могут быть различными, но основное устройство команд включает в себя поле операции (Opcode), индексные режимы адресации и операнды.

1. **Режимы адресации PDP-11. Пример.;**

Прямая адресация (Direct Addressing):

В этом режиме адрес операнда содержится в самой команде. Процессор обращается к ячейке памяти по указанному адресу и выполняет операцию.

Пример команды с прямой адресацией:

MOV 100(R0), R1 ; переместить данные из памяти по адресу 100 плюс содержимое регистра R0 в регистр R1

Регистровая адресация (Register Addressing):

В этом режиме адрес содержится в регистре процессора. Процессор обращается к ячейке памяти, адрес которой содержится в указанном регистре.

Пример команды с регистровой адресацией:

MOV (R2)+, R3 ; переместить данные из памяти по адресу, содержащемуся в регистре R2, в регистр R3, затем увеличить значение регистра R2

Косвенная адресация (Indirect Addressing):

В этом режиме адрес содержит указатель на адрес в памяти. Процессор сначала обращается к ячейке памяти, содержащей адрес операнда, а затем выполняет операцию по полученному адресу.

Пример команды с косвенной адресацией:

MOV @100, R4 ; загрузить содержимое памяти по адресу 100 в регистр R4 (адрес 100 содержится в памяти по указанному адресу)

1. **Характеристика процессоров семейства Intel x86. Формат команды;**

Процессоры семейства Intel x86 являются одними из наиболее широко используемых процессоров в мире. Это семейство процессоров включает в себя множество различных моделей, начиная от ранних процессоров 8086/8088 и заканчивая современными многоядерными процессорами с поддержкой технологий виртуализации и расширения инструкций.

Характеристики процессоров семейства Intel x86 (например, процессоры Intel Core i7, i9 и т.д.) включают:

32/64-битная архитектура:

Процессоры x86 поддерживают как 32-битные, так и 64-битные режимы работы. В 32-битном режиме составляющие процессора имеют ширину 32 бита, а в 64-битном - 64 бита, что позволяет обрабатывать большие объемы данных и памяти.

Различные уровни кэш-памяти:

Процессоры x86 обычно оборудованы различными уровнями кэш-памяти (L1, L2, L3), которые служат для временного хранения данных и инструкций, ускоряя доступ к ним.

Наличие множества специализированных инструкций:

Архитектура x86 поддерживает разнообразные инструкции для работы с векторными вычислениями, шифрованием, виртуальной машиной и многими другими задачами.

Присутствие аппаратной поддержки виртуализации:

Многие процессоры x86 предоставляют аппаратную поддержку для технологий виртуализации, что позволяет запускать несколько виртуальных машин непосредственно на одном физическом процессоре.

Формат команды в процессорах x86 значительно разнообразен из-за многочисленных опкодов и различных режимов адресации. Однако, в общем виде, команда x86 имеет следующую структуру:

[Префиксы] Опкод [Операнды]

Префиксы могут быть использованы для указания дополнительных свойств инструкции, таких как размерность операндов (32 или 64-битная), повторение инструкции и другие аспекты.

Опкод (Operation Code) определяет тип операции (например, LOAD, STORE, ADD и т.д.).

Операнды указывают на регистры, непосредственные значения или адреса в памяти, с которыми выполняется операция.

Процессоры семейства Intel x86 имеют сложную систему команд, которая обеспечивает обширные возможности для работы с данными, обработки информации и управления программным обеспечением.

1. **Режимы адресации процессора семейства Intel x86. Пример;**

Процессоры семейства Intel x86 поддерживают разнообразные режимы адресации, которые определяют способы доступа к операндам инструкций. Вот несколько основных режимов адресации для процессоров семейства Intel x86:

Прямая адресация (Immediate Addressing):

В этом режиме значение операнда указывается непосредственно в инструкции.

Пример:

MOV EAX, 10 ; переместить значение 10 в регистр EAX

Регистровая адресация (Register Addressing):

В этом режиме адрес операнда указывается через регистр процессора.

Пример:

ADD EAX, EBX ; добавить значение регистра EBX к значению регистра EAX

Прямая адресация (Direct Addressing):

В этом режиме значение операнда используется как адрес памяти.

Пример:

MOV EAX, [0x1000] ; переместить значение, находящееся по адресу 0x1000, в регистр EAX

Косвенная адресация (Indirect Addressing):

В этом режиме значение операнда интерпретируется как адрес, по которому находится нужное значение.

Пример:

MOV EAX, [EBX] ; загрузить значение, находящееся по адресу, содержащемуся в регистре EBX, в регистр EAX

Индексированная адресация (Indexed Addressing):

В этом режиме значение регистра используется как базовый адрес, а смещение добавляется к этому адресу.

Пример:

MOV EAX, [EBX+8] ; загрузить значение, находящееся по адресу (значение EBX + 8), в регистр EAX

Это лишь несколько примеров режимов адресации, поддерживаемых процессорами семейства Intel x86. Комбинации различных режимов адресации обеспечивают гибкость в работе с памятью и данными, позволяя программистам эффективно реализовывать алгоритмы и структуры данных.

1. **Триггеры. Виды триггеров. Пример с таблицей истинности;**

В контексте ассемблера триггеры могут быть связаны с микро-операциями и управлением состоянием процессора или других устройств. Одним из видов триггеров, который часто встречается в ассемблерном программировании, является триггер состояния. Такой триггер может изменять свое состояние в ответ на сигналы или определенные события.

В контексте ассемблерного программирования "триггер" обычно относится к устройствам или сигналам, которые инициируют определенные действия или реакции в программе или системе. Вот несколько основных видов триггеров, которые могут использоваться в ассемблерном программировании:

Триггеры по перепаду (Edge-triggered):

Эти триггеры реагируют на изменение состояния входного сигнала (например, сигнал переходит от 0 к 1 или от 1 к 0). Их также можно назвать истинными триггерами.

Триггеры по уровню (Level-triggered):

Эти триггеры реагируют на сигналы, оставаясь активными, пока входной сигнал находится в определенном состоянии (например, на уровне 1).

Триггеры по времени (Time-triggered):

Эти триггеры реагируют на сигналы в определенные моменты времени или с определенной задержкой.

Триггеры счетчиков (Counter-triggered):

Эти триггеры активируются после достижения определенного количества событий или импульсов.

Внешние триггеры (External triggers):

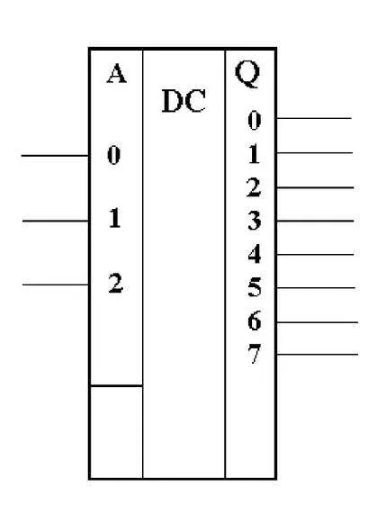
Реагируют на сигналы от внешних устройств или интерфейсов.

Таким образом, в ассемблерном программировании триггеры могут быть использованы для управления потоком выполнения программы, обработки внешних сигналов или взаимодействия с периферийными устройствами. Реализация триггеров уникальна для каждой конкретной аппаратной платформы и может варьироваться в зависимости от желаемых функций и задач программы.

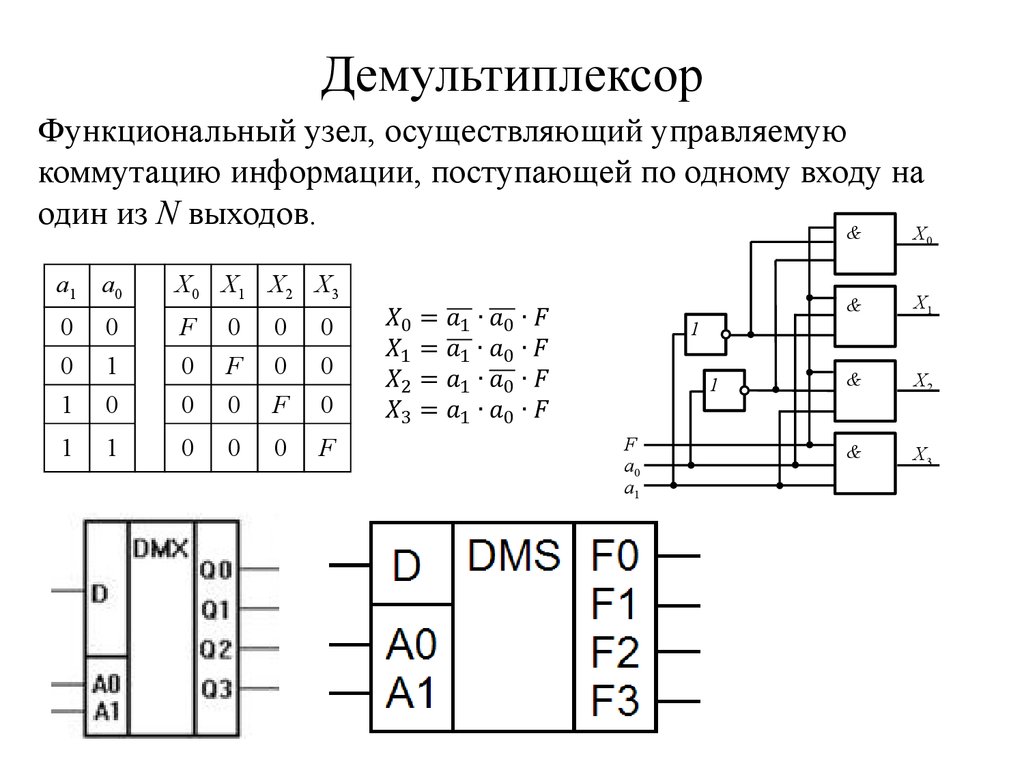
1. **(Де)шифратор, (де)мультиплексор. Описание и УГО каждого устройства с таблицей истинности;**

Дешифратор- Дешифратором называют комбинационную схему на M входов и выходов (нумеруемых в натуральном порядке – 0,1,

Демультиплексор — это логическое устройство, предназначенное для переключения сигнала с одного информационного входа на один из информационных выходов. Таким образом, демультиплексор в функциональном отношении противоположен мультиплексору. На схемах демультиплексоры обозначают через DMX или DMS.



Дешифратор



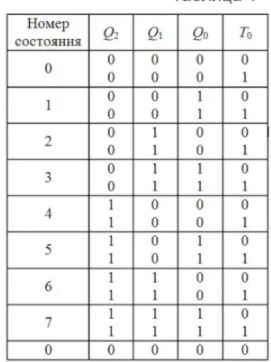
1. **Счётчик. Регистр. Описание и УГО каждого устройства с таблицей истинности;**

В контексте ассемблерного программирования счетчики и регистры являются главными устройствами для хранения и обработки данных. Давайте опишем каждое устройство и рассмотрим их таблицы истинности.

Счетчик:

Описание: Счетчик - это устройство, которое используется для подсчета сигналов или событий. В ассемблерном программировании счетчик может быть представлен регистром, значение которого увеличивается или уменьшается в зависимости от внешних сигналов или инструкций программы.

Управление: Возможны операции инкрементации, декрементации, сброса счетчика и загрузки нового значения.



Регистр:

Описание: Регистр - это высокоскоростное устройство памяти, используемое для хранения данных в компьютере или другом цифровом устройстве. В ассемблерном программировании регистры часто используются для хранения операндов, результатов промежуточных вычислений и адресов памяти.

Управление: Регистры могут быть загружены, выгружены, перезаписаны или использованы в арифметических и логических операциях.

1. **Необходимые компоненты для успешного запуска компьютера. Перечислите все компоненты компьютера, которые Вам известны.**

Для успешного запуска компьютера необходимо наличие нескольких основных компонентов. Вот перечень основных компонентов компьютера, необходимых для его функционирования:

Центральный процессор (CPU):

Основной вычислительный компонент компьютера, обрабатывающий инструкции программ и выполняющий расчеты.

Оперативная память (RAM):

Используется компьютером для временного хранения данных, на которые процессор должен быстро обратиться. RAM необходима для запуска операционной системы и приложений.

Материнская плата (материнская плата):

Основная плата, на которую устанавливаются все другие компоненты компьютера (процессор, оперативная память, видеокарта и др.).

Блок питания (Power Supply Unit, PSU):

Предоставляет энергию для работы всех компонентов компьютера.

Жесткий диск (Hard Disk Drive, HDD) или твердотельный накопитель (Solid State Drive, SSD):

Устройство для хранения данных и программ.

Видеокарта (Video Card) или встроенное графическое ядро:

Отвечает за вывод изображения на экран монитора.

Монитор (Display):

Устройство для визуального отображения информации, генерируемой компьютером.

Клавиатура и мышь:

Устройства ввода для управления компьютером и взаимодействия с программами.

Операционная система:

Программное обеспечение, управляющее ресурсами компьютера и обеспечивающее взаимодействие с пользователем.

Это основные компоненты, необходимые для запуска и функционирования компьютера. Разумеется, существует множество других компонентов, таких как система охлаждения, аудио- и сетевые карты, оптический привод, но они могут рассматриваться как дополнительные по отношению к основным компонентам.