МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Тобольский педагогический институт им. Д.И. Менделеева (филиал)
Тюменского государственного университета

УТВЕРЖДАЮ
Директор
_____ Шилов Є.П

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ
ПО ДИСЦИПЛИНЕ

АРХИТЕКТУРА КОМПЬЮТЕРА

44.03.05 – Педагогическое образование (с двумя профилями подготовки)
Профиль: математика; информатика
Форма обучения: очная

1. Паспорт оценочных материалов по дисциплине

1.1. Перечень компетенций

Код и наименование компетенции	Компонент (знаниевый/функциональный)
ПК-4 способность использовать возможности образовательной среды для достижения личностных, метапредметных и предметных результатов обучения и обеспечения качества учебно-воспитательного процесса средствами преподаваемых учебных предметов	Знает о специфике организации и функционировании вычислительных устройств на основе микропроцессоров
	Знает архитектурные особенности и принципы обработки информации и управления в микропроцессорной технике
	Умеет определять характеристики компьютерной техники и ее возможность для создания образовательной среды на уроках информатики в школе
	Умеет организовать обслуживание работоспособности компьютерной техники

1.2. Паспорт оценочных средств по дисциплине

№ п/п	Темы дисциплины в ходе текущего контроля, вид промежуточной аттестации	Код компетенции (или ее части)	Наименование оценочного средства (количество вариантов, заданий и т.п.)
1.	История вычислительной техники	ПК-4	Задания для самостоятельной работы
2.	Центральные и внешние устройства ЭВМ, их функционирование. Основы организации вычислительного процесса в цифровых устройствах	ПК-4	Задания для Лабораторных работ 1 - 16
			Задания для самостоятельной подготовки по данной теме и тестирование
3.	Вычислительные системы	ПК-4	Задания для Лабораторных работ 17 - 18
			Задания для самостоятельной подготовки по данной теме и тестирование
4.	*Современные тенденции развития архитектуры ЭВМ*	ПК-4	Задания для самостоятельной подготовки по данной теме и тестирование

2. Виды и характеристика оценочных средств

Текущий контроль осуществляется проверкой наличия конспектов лекций, выполнения заданий в ходе лабораторных работ, тестовых проверочных работ и самостоятельной работы.

2.1. Лабораторные работы

Лабораторные работы используются для формирования практико-ориентированных знаний, оценки умений по темам дисциплины. Выполнение заданий лабораторных работ включает в себя 3 этапа:

1) *Изучение/повторение необходимой теории* проходит в виде интерактивной беседы, рассказа, объяснения для понимания и уяснения студентами теоретической информации по данной теме, необходимой для эффективного выполнения практических заданий.

2*) Выполнение практических заданий* во время занятий и самостоятельной работы студентов.

3) *Защита заданий практической работы* проводится в виде сдачи результатов выполнения расчетно-графических работ и решения задач.

Содержание задний и критерии оценки результата доводятся до сведения обучающихся в начале семестра. Оценка объявляется непосредственно после проверки решения. В зависимости от уровня решения графического задания баллы могут распределяться от 0 до 4. Дополнительные баллы могут добавляться за сложность выполнения задания

Многие лабораторные работы носят проектный характер. Комплексная оценка компетенции ПК-4 оценивается в форме дидактических проектов по разработке управляющих программ, оценке работоспособности компьютерного оборудования и определения его характеристик, комплектации, подбор информации, разработку презентационных материалов и т.д.

Оценка «отлично»:

- работа оформлена в соответствии с требованиями,
- выполнены все задания лабораторной работы,
- студент четко и без ошибок ответил на все контрольные вопросы.

Оценка «хорошо»:

- работа оформлена в соответствии с требованиями,
- выполнены все задания лабораторной работы;
- студент ответил на все контрольные вопросы с замечаниями.

Оценка «удовлетворительно»:

- работа оформлена по требованиям, но с замечаниями,
- выполнены все задания лабораторной работы с замечаниями;
- студент ответил на все контрольные вопросы с замечаниями.

Оценка «неудовлетворительно» (не зачтено):

- студент не выполнил или выполнил неправильно задания лабораторной работы;
- студент не ответил на контрольные вопросы.

2.2. Тестовые задания

Критерии оценивания текстовых заданий

При составлении/подборе тестовых заданий заранее проектируется необходимый уровень сложности теста. Сложность теста определяется пятью уровнями:

2. Репродуктивный, основными операциями которого являются воспроизведение информации и ее преобразования алгоритмического характера.
3. Базовый, требующий от испытуемого понимания существенных сторон учебной информации, владения общими принципами поиска алгоритмов.
4. Повышенный, уровень сложности задания, требующий от испытуемого умения преобразовывать алгоритмы к условиям, отличающимся от стандартных, умение вести эвристический поиск.
5. Творческий, предполагающий наличие самостоятельного, критического оценивания учебной информации, умение решать нестандартные задания, владение элементами исследовательской деятельности.

Каждому из заданий в соответствии с его сложностью приписывается определенное число, например: информационного характера - 1; репродуктивного - 1,5; базового уровня - 2; повышенной сложности - 2,5; творческого – 3 (или другое количество баллов). Таким образом, получается измерительное устройство в виде шкалы, достаточно понятной и наглядной,

которую можно предлагать ученикам или использовать при выставлении баллов за работу над тестом.

Измерительная шкала

Задание	Информационное	Репродуктивное	Базовое	Повышенного уровня	Творческое
Балл	1	1,5	2	2,5	3

Сложность теста определяется как среднее арифметическое сложностей всех заданий, входящих в рассматриваемый тест: $CT = \frac{\sum_{i=1}^{n} C3_i}{n}$, где CT - сложность теста; $C3_i$ - сложность i-го задания теста; n - число заданий в тесте.

Для определения, каким будет тест по вычисленной сложности, следует воспользоваться специальной таблицей:

Определение вида теста по его сложности

Тест	Информативный (ТИ)	Репродуктивный (ТР)	Базовый (ТБ)	Повышенной сложности (ТП)	Творческий (ТТ)
СТ	1 - 1,3	1,4 – 1,6	1,7 – 2,1	2,2 – 2,4	> 2.5

Результаты выполнения различных тестов следует оценивать в зависимости от их сложности, при помощи специальной нормировочной таблицы:

Оценка результатов выполнения тестов различной сложности

СТ \ %	100	90	80	70	60	50	40	30	20	10	0
ТР	«5»		«4»		«3»		«2»		«1»		
ТБ	«5»			«4»		«3»		«2»		«1»	
ТП	«5»				«4»		«3»		«2»		

2.3. Зачет с оценкой

Дифференцированный зачет (зачет с оценкой) является формой оценки качества освоения обучающимся основной профессиональной образовательной программы по разделам дисциплины, демонстрирует сформированные умения и компетенции. По результатам дифференцированного зачета обучающемуся выставляется оценка «неудовлетворительно», «удовлетворительно», «хорошо», или «отлично».

Зачет может быть выставлен автоматически по результатам балльно-рейтинговой аттестации. Содержание оцениваемой работы студентов приведено в пункте 3 рабочей программы. Результаты освоения дисциплины в течение семестра оцениваются по балльно-рейтинговой системе (см. рабочую программу дисциплины).

Соответствие баллов рейтинговой системы оценки успеваемости студентов

Вид аттестации	Соответствие рейтинговых баллов и академических оценок		
	Удовлетворительно	***Хорошо***	***Отлично***
Зачет с оценкой	61-75 баллов	76-90 баллов	91-100 баллов

Зачет может проводиться по устным вопросам или в формате компьютерного тестирования. В случае устного ответа на вопросы время для подготовки 15 мин, для ответа на поставленный вопрос – не более 10 минут. Преподавателю предоставляется право задавать обучающимся дополнительные вопросы в рамках программы дисциплины. Общее время сдачи зачета на 1 студента – 15 минут. В случае компьютерного тестирования общее время тестирования – 45 минут.

Критерии выставления оценки на дифференцированном зачете

Оценка «отлично»:

- Результаты освоения программы дисциплины соответствуют повышенному уровню в соответствии с установленными критериями.
- При ответе на устные вопросы демонстрирует свободное владение материалом.

– Свободно отвечает на дополнительные вопросы.
– Или задания теста выполнены правильно не менее чем на 80%

Оценка «хорошо»:

– Результаты освоения программы дисциплины соответствуют базовому уровню в соответствии с установленными критериями.
– При ответе на устные вопросы демонстрирует знание основных теоретических и прикладных вопросов.
– Частично отвечает на дополнительные вопросы.
– Или задания теста выполнены правильно не менее чем на 65%

Оценка «удовлетворительно»:

– Результаты освоения программы дисциплины соответствуют пороговому уровню в соответствии с установленными критериями.
– С затруднением отвечает на устные вопросы зачета.
– Затрудняется отвечать на дополнительные вопросы.
– Или задания теста выполнены правильно не менее чем на 50%

Оценка «неудовлетворительно»: выставляется в том случае, если ответ студента не соответствует пороговому уровню критериев

При использовании компьютерного итогового тестирования, все вопросы предварительно заносятся преподавателем в среду тестовой оболочки Конструктор тестов 2.5 (Keepsoft) (для локального тестирования), или в приложение MS Forms среды Microsoft Office 360 для тестирования студентов в режиме on-line. Оценивание результатов компьютерного тестирования студентов определяется автоматически в соответствии со стандартными требованиями тестологии (п.2.2.)

3. Оценочные средства

3.1. Содержание практических работ

Лабораторная работа 1. Организация и архитектура материнской платы

Цель работы: знакомство с архитектурой и организацией системной платы компьютера. Изучение спецификаций и назначения основных компонентов материнской платы.

1. Изучите теоретический материал

Архитектура ЭВМ — это абстрактное представление ЭВМ, которое отражает логическую организацию. Понятие архитектуры включает в себя:

– организацию и способы адресации памяти;
– способы представления и форматы данных ЭВМ;
– набор машинных команд ЭВМ;
– форматы машинных команд;
– обработку нештатных ситуаций (прерываний).

Организация ЭВМ — это представление ЭВМ, которое отражает структурную и физическую ее организацию. Понятие организации включает в себя:

– структурную схему ЭВМ;
– средства и способы доступа к элементам структурной схемы ЭВМ;
– организацию и разрядность интерфейсов ЭВМ;
– набор и доступность регистров;
– назначение, основные характеристики, принципы работы устройств ЭВМ.

Структурную схему вычислительной машины можно представить в виде отдельных модулей, объединенных каналом обмена информации. Набор модулей, входящих в состав ЭВМ, может быть достаточно велик и изменяется в зависимости от назначения конкретной машины.

В мире существует множество ЭВМ (компьютеров) различных фирм, различающихся по сложности, назначению, производительности, размеру и т.д. Главным компонентом любого компьютера можно с уверенностью назвать материнскую плату.

Материнская плата (системная) **–** центральная комплексная печатная плата, предоставляющая электронную и логическую связь между всеми устройствами, входящими в состав персонального компьютера**.**

На материнскую плату нанесено огромное количество проводящих дорожек, объединяющих компоненты и разъёмы, и контактных площадок для микроконтроллеров и электронных компонентов. Причём плата состоит из нескольких слоев, изготовленных из диэлектрика текстолита, и каждый слой содержит такие дорожки. Выводы для установки компонентов находятся только на верхнем слое. Сверху плата покрыта диэлектрическим лаком, чтобы предотвратить короткое замыкание и защитить её от непредвиденных обстоятельств.

Производством материнских плат занимаются около одиннадцати различных производителей. Лидерами по производству системных плат на данный момент являются компании: Gigabyte, ASUS, MSI и Foxconn. Также можно встретить материнские платы от крупнейшей по производству процессоров компании – Intel. Материнские платы отличаются друг от друга размерами. Эти размеры стандартизированы и называются ***форм-факторами*** (табл 1).

Таблица 1

Форм – факторы системных плат

Форм-фактор	Физические размеры		Спецификация, год	Примечание
	дюймы	миллиметры		
XT	8,5 × 11	216 × 279	IBM, 1983	архитектура IBM PC XT устарел
AT	12× 11/13	305×279/330	IBM, 1984	архитектура IBM PC AT (Desktop/Tower) устарел
Baby-AT	8,5× 10/13	216× 254/330	IBM, 1990	архитектура IBM PC XT устарел
ATX	12 × 9,6	305 × 244	Intel, 1995	Для системных блоков типов Mini Tower, Miditower, Bigtower, FullTower
eATX	12 × 13	305 × 330	Intel, 1999	для системных блоков Bigtower, Full Tower
WTX	14 × 16,75	355,6 × 425,4	1999	для высокопроизводительных рабочих станций и серверов среднего уровня
Mini-ITX	6,7 × 6,7	170 × 170	VIA Technologies, 2003	для сверхмалых ПК, актуален и перспективен. Допускаются только 100 Вт блоки питания
Nano-ITX		120 × 120	VIATechnologies, 2004	для сверхмалых ПК, актуален и перспективен
BTX	12,8 × 10,5	325 × 267	Intel, 2004	допускается до 7 слотов и 10 отверстий для монтажа платы
MicroBTX	10,4 × 10,5	264 × 267	Intel, 2004	допускается до 4 слотов и 7 отверстий для монтажа платы
PicoBTX	8,0 × 10,5	203 × 267	Intel, 2004	допускается 1 слот и 4 отверстия для монтажа платы
CEB	12 × 10,5	305 × 267	2005	для высокопроизводительных рабочих станций и серверов среднего уровня

Pico-ITX	3,9 × 2,7	100 x 72	VIA, 2007	используются в ультракомпак-тных встраиваемых системах
SSI CEB	12 × 10,1	305 x 259		системные платы этого стандарта обычно служат для построения серверов. Разъемы для подключения блока питания имеют 24+8 контактов.
Extended ATX (EATX)	12 × 13	305 × 330		стандарт плат для рабочих станций и серверов в RackMount-исполнении. Обычно используется для материнских плат серверного класса с двумя процессорами и/или слишком большого для стандартной материнской платы стандарта ATX количества плат расширений
Ultra ATX	9,6 × 14,4	244 × 367	Foxconn, 2008	это негабаритная версия ATX, которая поддерживает 10 слотов расширения (в отличие от семи слотов в стандартной ATX плате). Вследствие этого требует корпус достаточной высоты (специально выпущены корпуса Ultra ATX).

Наиболее популярные форм-факторы материнских плат для персональных компьютеров представлен на рисунке 1.

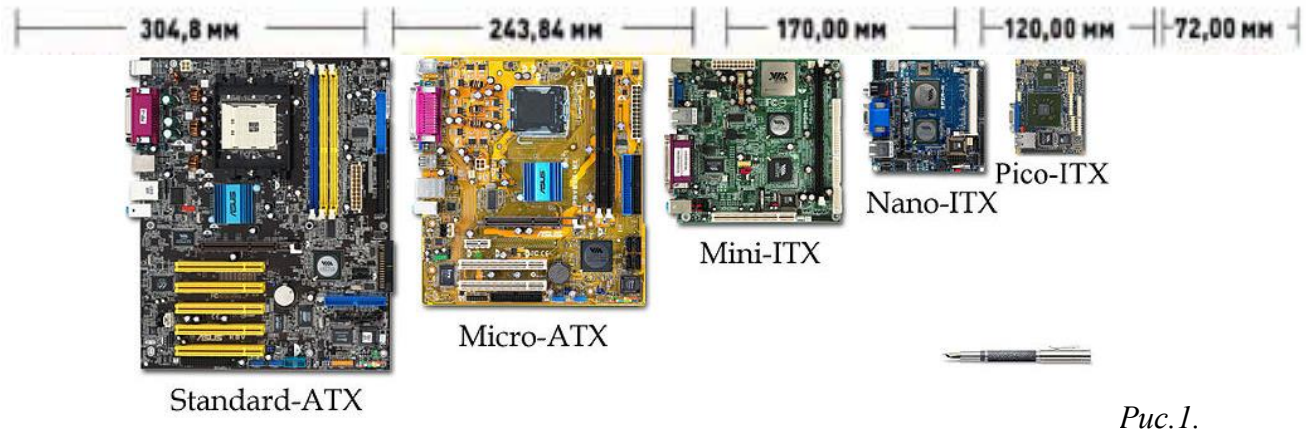


Рис.1.

Проводящие дорожки материнской платы объединяют между собой несколько ключевых подсистем - блоков материнской платы. Этими блоками являются:

- разъём (сокет) процессора и система его питания,
- подсистема памяти и разъёмы для установки модулей с собственной системой питания,
- разъёмы для установки карт расширения (функциональности),
- разъёмы для подключения накопителей.

Каждый набор таких дорожек может работать по собственному принципу (стандарту) и называется ***шиной***. Все указанные устройства объединяются в систему посредством общего канала обмена информацией, называемого интерфейсом «системная шина», представляющего собой иерархию шин: адресная, данных и управления (рис.2).

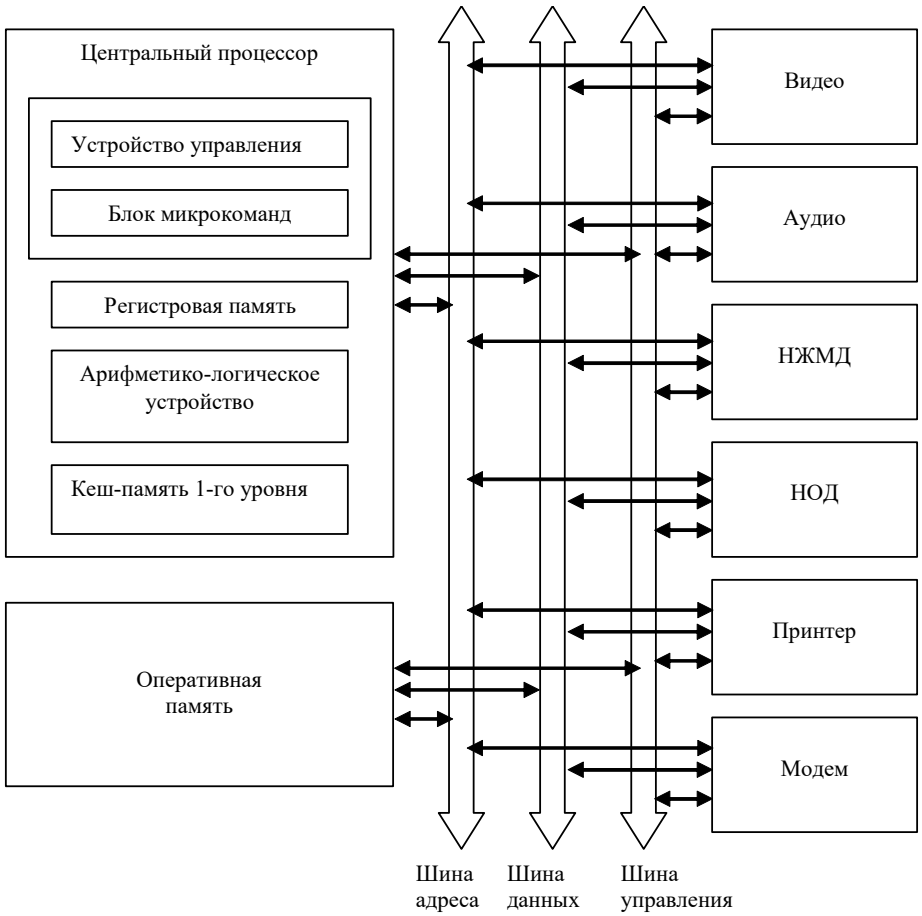


Рис.2.

Основой любой материнской платы является набор ключевых микросхем, также называемый ***набором логики или чипсетом***. Разработкой таких наборов занимаются несколько крупнейших мировых компаний: Intel, NVIDIA, AMD, VIA, SIS. То, какой чипсет положен в основу материнской платы определяет: какой процессор, какую оперативную память и в каком объёме можно установить, сколько устройств можно подключить и как быстро всё это будет работать. Чипсет состоит из интегральных микросхем, называемых ***мостами***. Чаще всего встречаются двухкомпонентные чипсеты, состоящие из ***северного*** и ***южного*** мостов.

Северный мост (Northbridge или MCH, MemoryControllerHub) обеспечивает взаимосвязь между процессором, оперативной памятью и специализированными шинами (PCI, PCI Express и т.п.). Именно возможности северного моста определяют, какую оперативную память (SDRAM, DDR, DDR2, DDR3) можно установить в материнскую плату, какой максимальный объём можно установить, в каких режимах она может работать. В прошлом северный мост в обязательном порядке обеспечивал работу специальной шины AGP, по которой подключалась видеокарта. На сегодняшний день её место заняла более универсальная шина PCI Express. Так как при скоростной передаче данных мост испытывает немалую нагрузку, он выделяет немало тепла и требует качественного охлаждения, поэтому на материнских платах устанавливают кулеры. Северный мост соединён с южным мостом посредством специальной шины или через несколько каналов шины PCI Express.

Задачейюжногомоста является предоставление интерфейсов ввода-вывода для устройств компьютера. (Поэтому по-английски он официально называется I/O Controller Hub (ICH) – контроллер-концентратор ввода-вывода). Он обеспечивает поддержку материнской платой низкоскоростных, но тем не менее важных шин. К устройствам, встроенным в южный мост относятся:

- контроллер DMA (Direct Memory Access),
- контроллер прерываний,

- контроллеры устройств хранения данных (IDE и SATA-жёстких дисков и оптических приводов),
- контроллер питания и др.

Кроме того, современные южные мосты чаще всего содержат: встроенные звуковые контроллеры, сетевые контроллеры, USB - контроллеры, RAID-контроллеры.

К функциям южного моста также относится: работа часов (Real Time Clock, RTC), специальной шины, позволяющей оперативное управление настройками платы, доступ к информации BIOS – базовой системы ввода-вывода. BIOS фактически является микропрограммой, позволяющей материнской плате обращаться к своим подсистемам и работать так, как нужно.

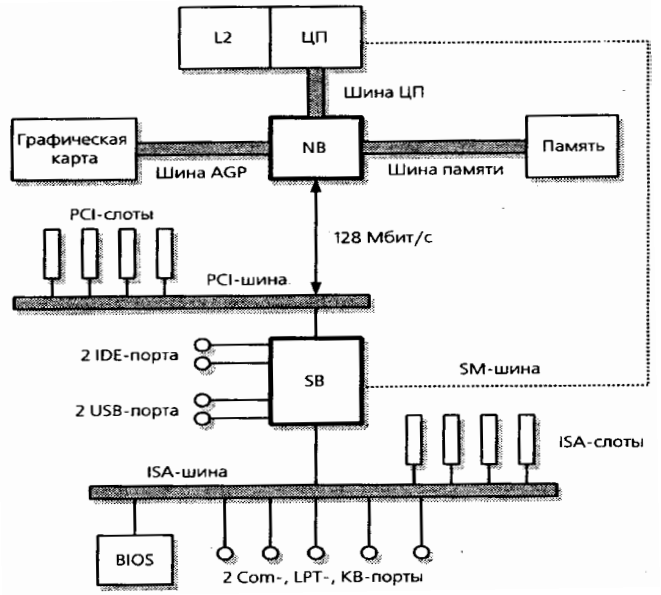


Рис.3.

Архитектура чипсета

В настоящее время выделяют чипсеты с *классической и хаб-архитектурой*

А) Классическая (рис.3)

Б) Хаб-архитектура (рис.4)

Термин «хаб» дословно означает концентратор, но здесь является коммутатором, т.е. каждый из чипов представляет собой коммутатор и может коммутировать подключенные к нему устройства для обмена между собой без участия центрального процессора.

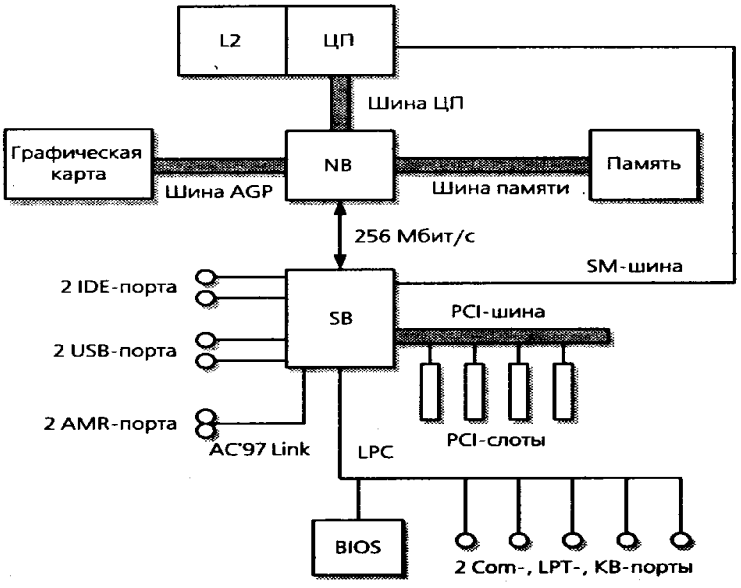


Рис.4.

Организация системной платы при двухкомпонентном чипсете

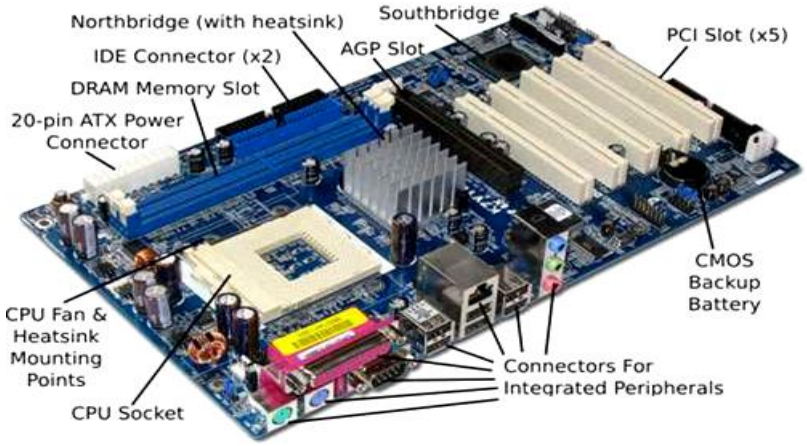


Рис.5.

Организация системной платы при однокомпонентном чипсете

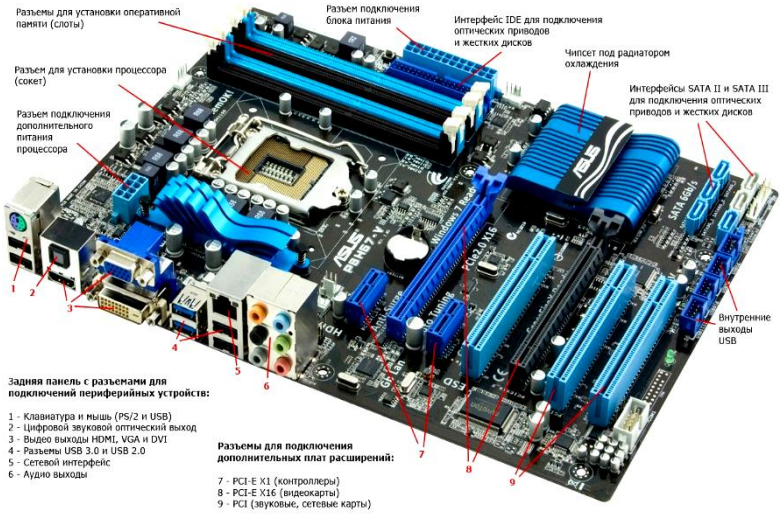


Рис.6.

Интерфейсы для подключения Floppy, мыши и клавиатуры чаще всего не включаются в состав южного моста, эти функции осуществляются специальным контроллером Super I/O. Кроме того, он следит за температурами, напряжениями и скоростями вращения вентиляторов.

Часто дополнительные, так называемые периферийные контроллеры, встроенные в южный мост, требуют дополнения в виде ещё одного чипа, чаще всего это контроллеры USB, FireWire, звука, сети.

Иногда встречаются чипсеты, состоящие только из одного чипа. Чаще всего это чипсеты для платформы AMD. Это объясняется тем, что основная часть северного моста – контроллер памяти – перенесен в сам процессор.

Основные разъемы материнской платы

- Важнейшим разъемом на плате является сокет процессора (см. рис. 5, 6). Он представляет собой специальное устройство, состоящее из большого количества контактов, расположенных в определённом порядке, определяющем правильное расположение процессора.
- На материнской плате для установки модулей памяти предусмотрены специальные разъёмы, называемые слотами DIMM, DDR. Чаще всего их 4, на материнских платах малых размеров иногда устанавливают только 2 слота.
- На плате находится разъём для подключения питания, на сегодняшний день стандарт предусматривает установку минимум двух разъемов – 24-контактного ATX и 4-контактного ATX12V для дополнительной линии 12 В.

Питание, подаваемое блоком питания, проходит преобразование, стабилизацию и фильтрацию с помощью силовых транзисторов («мосфетов»), дросселей и конденсаторов, составляющих VRM (Voltage Regulation Module, модуль регулирования напряжения). Питание процессора и чипсета осуществляется одним VRM, питание модулей памяти – чаще всего другим.

- Ниже процессорного сокета чаще всего расположен один или два специализированных разъёма для установки видеокарты.
- В той же области платы находятся слоты PCI для подключения карт расширения, они стандартизированы и позволяют подключить практически все возможные контроллеры.
- С краю платы, обычно расположены разъёмы для подключения накопителей – жёстких дисков и оптических приводов. Сегодня активно занимает лидирующую позицию интерфейс Serial ATA, постепенно вытесняя старый IDE даже с рынка оптических приводов.
- Также там может находиться разъём Floppy (дисковода 3,5”-носителей), но в последнее время всё идет к тому, что от него откажутся. Все эти накопители подключаются к материнской плате с помощью специальных кабелей, называемых шлейфами.
- В низу материнской платы располагаются контакты для подключения кнопок и индикаторов с передней панели корпуса: кнопки включения и перезагрузки, индикаторы питания и активности жёсткого диска, системный динамик.
- На плате можно увидеть батарейку (аккумулятор), которая обеспечивает питание микросхемы памяти, в которой содержится прошивка BIOS, и поддерживает работу системных часов. BIOS хранится в чипе памяти, который чаще всего устанавливается в специальную «кроватку», но может быть и впаян на плату.
- Работа всех компонент материнской платы подчиняется работе тактового генератора, состоящего из резонатора и специализированной микросхемы. В итоге с помощью всех этих встроенных разъёмов, дополнительных контроллеров материнская плата фактически объединяет абсолютно все устройства, входящие в состав компьютера в целостную систему.

На задней стороне материнской платы находится панель с разъёмами для подключения внешних устройств – клавиатуры и мыши, USB-устройств и многого другого.

2. Выполните практическое задание

1. Ознакомиться с теоретическим материалом.
2. Изучить основные компоненты материнской платы: форм-фактор, основные разъемы их спецификацию и размещение.

Контрольные вопросы

1. Материнская плата – это ……
2. Форм-фактор материнской платы – это ……
3. Перечислите основные подсистемы материнской платы
4. Набор проводящих дорожек, работающих по определенному протоколу, называется ……
5. Чипсет – это …….
6. Компонентами чипсета являются …….
7. Что обеспечивает северный мост?
8. Какие устройства входят в состав южного моста?
9. Если северный и южный мост соединяются между собой через специализированную шину, то такая архитектура называется …….

Лабораторная работа 2. Центральный процессор. работа с регистрами центрального процессора

Цель работы: знакомство с архитектурой и организацией центрального процессора. Знакомство с регистрами центрального процессора, их обозначением и назначением. Научиться устанавливать требуемое значение регистров с помощью программы DEBUG.

Основные теоретические сведения

Центральный процессор - основное устройство любой ЭВМ, осуществляющее обработку данных и выполняющее функции управления системой (инициирование ввода/вывода, управление доступом к основной памяти, обработку сигналов, поступающих от различных внешних устройств и от внутренних устройств ЭВМ и др.).

Краткая история развития микропроцессоров

- 1940-е – начало 1950-х годов - создание процессоров с использованием электромеханических реле, ферритовых сердечников (устройств памяти) и вакуумных ламп.
- 1950-е годы создание процессоров на основе использования транзисторов и транзисторных схем.
- Середина 1960-х годов - создание процессоров на основе использования интегральных микросхемы не высокой степени интеграции.
- 1971 г. - выпуск фирмой Intel первого в мире 4-разрядного микропроцессора 4004, предназначенного для использования в микрокалькуляторах (Джек Килби) (рис.1)

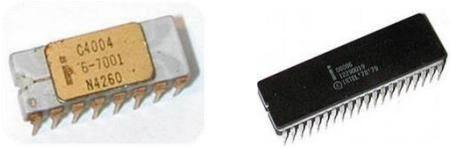


Рис.1.

- Начало 80-х г. XX века - начало выпуска микропроцессоров Intel 8086, заложивших основы архитектуры всех современных ПК (рис.1.).
- 1980-е – 1990-е годы - процессоры архитектуры x86 всё более активно используются во всех областях компьютерной индустрии.
- Начало XI-ого века - создание многоядерных процессоров.

При изготовлении современных сверхбольших интегральных схем используется метод литографии. При этом, на подложку будущего микропроцессора (тонкий круг из монокристаллического кремния, либо сапфира) через специальные маски, содержащие прорези, поочерёдно наносятся слои проводников, изоляторов и полупроводников. Соответствующие вещества испаряются в вакууме и осаждаются сквозь отверстия маски на кристалле процессора. В результате появляется сложная многослойная структура, содержащая от сотен тысяч до миллиардов транзисторов. В зависимости от подключения транзистор работает в микросхеме как транзистор, резистор, диод или конденсатор. Одновременно на подложке формируется порядка сотни процессорных кристаллов (рис.3).

Рис.2.

Рис.3.

Виды процессоров

- Центральные и специализированные;
- Ввода/вывода и передачи данных

- Коммуникационные.

Основные характеристики центрального процессора

1. Разрядность
2. Тактовая частота
3. Количество транзисторов
4. Количество ядер
5. Размер техпроцесса
6. Размер процессорных уровней КЭШ

Логическая структура центрального процессора

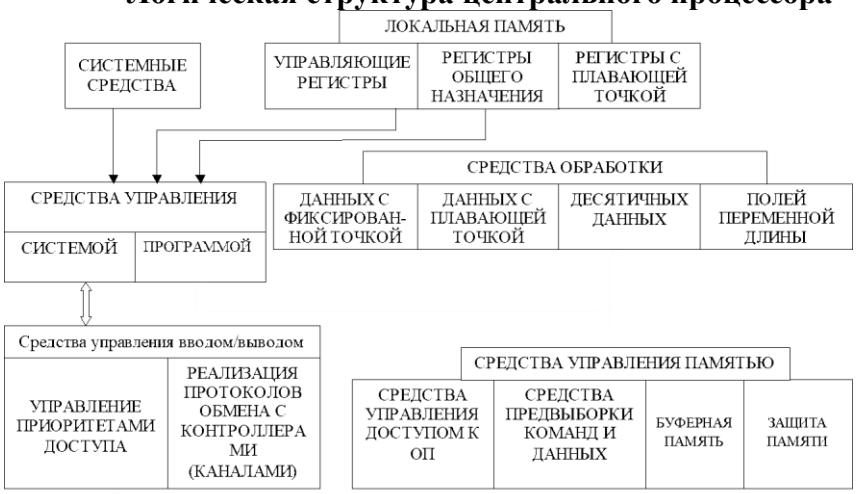


Рис.4.

Схема физического устройства ЦП

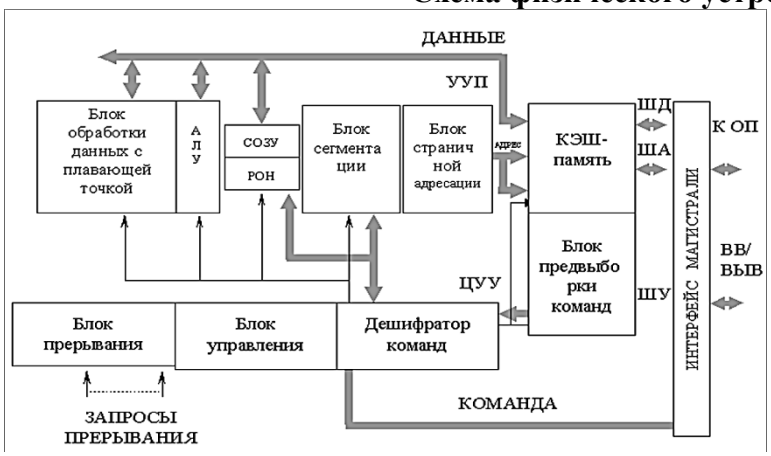


Рис.5.

ЦУУ - Центральное устройство управления;
АЛУ - Арифметико-логическое устройство;
УУП - Устройство управления памятью;
СЗУ - Сверхоперативное запоминающее устройство;
РОН – регистры общего назначения
ШД – шина данных;
ШУ – Шина управления
КОП – код операции

Устройство управления предназначено для организации обмена информации между центральным процессором и остальными частями ЭВМ. С помощью этого устройства процессор считывает команды и данные из оперативной памяти и записывает данные в память. Другой его функцией является дешифрирование кодов команд. Результат дешифрирования используется блоком микрокоманд, для генерации так называемой микропрограммы – последовательности внутренних сигналов процессора, обеспечивающих совместное

функционирование устройств центрального процессора с целью исполнения той или иной команды.

Арифметико-логическое устройство предназначено для организации переработки информации внутри центрального процессора. Оно позволяет выполнять арифметические и логические операции над данными, а также сдвиги. К арифметическим операциям относятся: сложение, вычитание, умножение и деление. В состав логических операций входят: отрицание, операция "И", операция "ИЛИ" и операция "ИСКЛЮЧАЮЩЕЕ ИЛИ".

Регистровая память используется центральным процессором для хранения данных и управляющей информации. Регистровая память образуется так называемыми регистрами, которые представляют собой ячейки памяти, входящие в состав процессора и доступные из машинной программы. Доступ к регистрам осуществляется значительно быстрее, чем к ячейкам оперативной памяти, поэтому использование регистров заметно уменьшает время выполнения программ. Все регистры центрального процессора Intelx86 имеют размер слова (два байта – 16 битов), за каждым из них закреплено определенное имя.

Кэш-память является "посредником" между процессором и оперативной памятью. В ней хранятся наиболее часто используемые участки оперативной памяти. Время доступа процессора к кэш-памяти в несколько раз меньше, чем к обычной памяти. За счет этого среднее время доступа к памяти значительно сокращается.

Организация многоядерного процессора

Многоядерный процессор — центральный процессор, содержащий два и более вычислительных ядра на одном процессорном кристалле или в одном корпусе. Термин мультиядерный обычно применяется к центральным процессорам, содержащим два и более ядра общего назначения. Под многоядерностью процессора понимают, что несколько ядер являются интегрированными на одну интегральную схему (изготовлены на одном кремниевом кристалле). Если в корпус объединены несколько полупроводниковых кристаллов, то конструкцию называют многопроцессорной (например, серверные материнские платы часто имеют 2 или 4 сокета для подключения нескольких чипов) (рис.6).

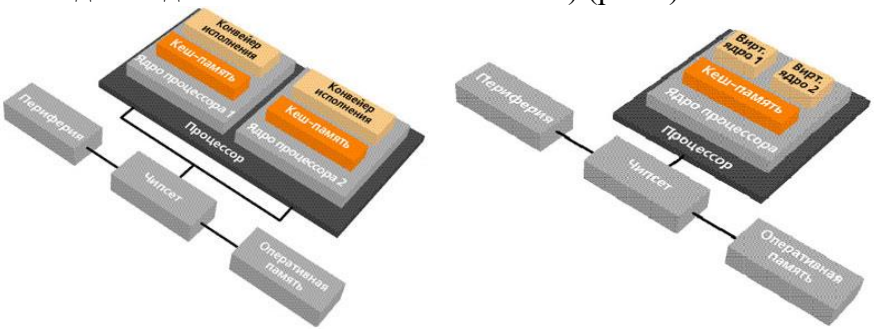


Рис.6.

Регистры центрального процессора

Для выполнения очередной команды данные (в виде двоичных чисел) загружаются в один из регистров процессора. В процессорах семейства Intel имеется четыре основных регистра: AX (аккумулятор), BX (база), CX (счетчик) и DX (данные). Регистры общего назначения применяются для хранения данных. Их можно использовать во всех арифметических и логических командах. В то же время каждый их них имеет определенную специализацию (некоторые команды «работают» только с определенными регистрами). Например, команды умножения и деления требуют, чтобы один из операндов находился в регистре AX или в регистрах AX и DX (в зависимости от размера операнда), а команды управления циклом используют регистр CX в качестве счетчика цикла.

Регистры AX, BX, CX и DX конструктивно устроены так, что возможен независимый доступ к их старшей и младшей половинам; можно сказать, что каждый из этих регистров состоит из двух байтовых регистров, обозначаемых AH, AL, BH и т.д. (H – high, старший; L – low, младший):

AX:	AH	AL
BX:	BH	BL
CX:	CH	CL
DX:	DH	DL
Биты:	15 8	7 0

Таким образом, с каждым из этих регистров можно работать как с единым целым, а можно - с его левой или правой частями.

Например, можно записать слово в AX, а затем считать только часть слова из регистра AH или заменить только часть в регистре AL и т.д. Такое устройство регистров позволяет использовать их для работы и с числами, и с символами.

Кроме того, в процессорах семейства Intel для хранения адресов выполняемых команд и адресов загружаемых данных существуют четыре сегментных регистра (CS, DS, SS, ES) и пять регистров смещения. Их обозначение и название приведены в таблице 1.

Таблица 1

Обозначение регистра	Название	Обозначение регистра	Название
SC	Сегмент кода (программы)	SP	Указатель стека
DS	Сегмент данных	BP	Указатель базы
SS	Сегмент стека	SI	Индекс источника
ES	Дополнительный сегмент	DI	Индекс назначения
IP	Указатель команд		

Регистры BX и BP очень часто используются как базовые регистры при обращении к памяти, а SI и DI – как индексные. Регистр SP обычно указывает на вершину стека – специально организованной области памяти, аппаратно поддерживаемой в ЭВМ.

Сегментные регистры CS, DS, SS и ES используются при формировании адресов команд и данных, расположенных в оперативной памяти. Они не могут быть операндами никаких команд, кроме команд пересылки и стековых команд.

Счетчик команд IP всегда содержит адрес (смещение от начала программы) той команды, которая должна быть выполнена следующей (начало программы хранится в регистре CS). Содержимое регистра IP можно изменить только командами перехода.

Существуют еще несколько вспомогательных регистров (флаговый регистр, регистры источника и приемника и др.), а также регистры математического сопроцессора (в нем производятся ускоренные операции над вещественными числами) и др.

Флаговый регистр

Флаговый регистр представляет собой набор отдельных регистров-битов, называемых флагами. Флаги логически можно объединить в две группы: шесть статусных флагов, в которые заносится информация о состоянии процессора (обычно указывающая на то, что произошло при выполнении арифметических операций и операций сравнения), и три управляющих флага, которые управляют работой некоторых команд процессора. Характеристики всех флагов приведены в таблице 2:

Таблица 2

Наименование флага	Назначение	Обозначение	Значение, когда установлен	Значение, когда сброшен
Статусные флаги				
Флаг переноса (да/нет)	Указывает на наличие переноса при выполнении арифметических операций	CF	CY	NC
Флаг переполнения (да/нет)	Указывает на переполнение при выполнении арифметических операций со знаковыми числами	OF	OV	NV
Флаг нуля (да/нет)	Указывает на нулевой результат при выполнении арифметических операций или на равенство при операциях сравнения	ZF	ZR	NZ

Флаг знака (отрицательный/ положительный)	Указывает на отрицательный результат при выполнении арифметических операций или устанавливает признак неравенства при операциях сравнения	SF	NG	PL
Флаг четности (четное / нечетное)	Указывает на наличие в операнде четного числа битов, равных 1, после выполнения команды	PF	PE	PO
Флаг арифметического переноса (да/нет)	Указывает на необходимость корректировки после выполнения арифметических операций с числами, представленными в виде двоично-десятичных кодов	AF	AC	NA
Управляющие флаги				
Флаг направления(уменьшение/увеличение)	Управляет способом изменения адресов при операциях при побайтовой обработке данных, например, при работе со строками	DF	DN	UP
Флаг прерывания(возможно/ невозможно)	Управляет процессом блокировки и разблокировки прерываний	IF	EI	DI
Флаг трассировки	Управляет пошаговым выполнением операций	TF		

Все флаги собраны в 16-разрядном регистре флагов (каждый флаг – это один из разрядов регистра, часть его разрядов не используется) (рис.7).

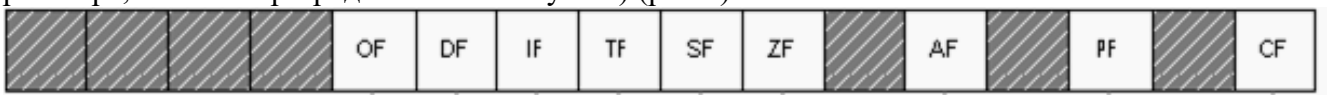


Рис.7.

Непосредственного доступа как к отдельным флагам, так и к регистру флагов в целом центральный процессор не предоставляет. Флаги условий автоматически меняются при выполнении команд и фиксируют те или иные свойства их результата (например, равен ли результат нулю). Флаги состояний меняются из программы и оказывают влияние на дальнейшее поведение процессора (например, блокируют прерывания).

Представление данных

С представлением данных в памяти ЭВМ тесно связано понятие типа данных. Понятие типа данных носит двойственный характер. С точки зрения размерности, микропроцессор аппаратно поддерживает следующие основные типы данных.

- ***Байт*** – восемь последовательно расположенных битов, пронумерованных от 7 до 0, при этом бит 0 является самым младшим значащим битом.
- ***Слово*** – последовательность из двух байт, имеющих последовательные адреса. Размер слова – 16 бит; биты в слове нумеруются от 15 до 0. Байт, содержащий нулевой бит, называется младшим байтом, а байт, содержащий 15-й бит, – старшим байтом. Процессоры Intel имеют важную особенность – младший байт всегда хранится по меньшему адресу. Адресом слова считается адрес его младшего байта. Адрес старшего байта может быть использован для доступа к старшей половине слова.
- ***Двойное слово*** – последовательность из четырех байт (32 бита), расположенных по последовательным адресам. Нумерация этих бит производится от 31 до 0. Слово, содержащее нулевой бит, называется младшим словом, а слово, содержащее 31-й бит, — старшим словом. Младшее слово хранится по меньшему адресу. Адресом двойного слова считается адрес его младшего слова. Адрес старшего слова может быть использован для доступа к старшей половине двойного слова.

Кроме трактовки типов данных с точки зрения их разрядности, процессор на уровне команд поддерживает логическую интерпретацию этих типов. С точки зрения логической интерпретации выделяют следующие типы данных.

- ***Целый тип без знака*** – двоичное значение без знака, размером 8, 16 или 32 бита. Числовой диапазон для этого типа следующий:

  –байт – от 0 до 255;

–слово – от 0 до 65 535;
–двойное слово – от 0 до $2^{32} – 1$.

- ***Целый тип со знаком*** – двоичное значение со знаком, размером 8, 16 или 32 бита. Знак в этом двоичном числе содержится в 7, 15 или 31-м бите соответственно. Ноль в этих битах в операндах соответствует положительному числу (+), а единица – отрицательному (-). Отрицательные числа представляются в дополнительном коде. Числовые диапазоны для этого типа данных следующие:
  –8-разрядное целое – от – 128 до 127;
  –16-разрядное целое – от – 32768 до 32767;
  – 32-разрядное целое – от $– 2^{31}$ до $2^{31} – 1$.
- ***Указатель*** на память(адрес) бывает двух типов.
  – *Ближний тип* – 16-разрядный логический адрес, представляющий собой относительное смещение в байтах от начала сегмента (короткий адрес).
  – *Дальний тип* – 32-разрядный логический адрес, состоящий из двух частей: 16-разрядной сегментной части и 16-разрядного смещения (полный адрес).
- ***Цепочка***- представляет собой некоторый непрерывный набор байтов, или слов максимальной длиной до 64 Кбайт.
- ***Символ*** – байт, в который записывается код символа – целое от 0 до 255. В ЭВМ используется система кодировки ASCII (American Standard Code for Information Interchange).
- ***Строка*** – последовательность символов, которая размещается в соседних байтах памяти, так, что код первого символа строки записывается в первом байте, код второго символа - во втором байте и т.п. Адресом строки считается адрес ее первого байта.
- ***Неупакованный двоично-десятичный тип*** – байтовое представление десятичной цифры от 0 до 9. Неупакованные десятичные числа хранятся как байтовые значения без знака по одной цифре в каждом байте. Значение цифры определяется младшим полубайтом.
- ***Упакованный двоично-десятичный тип*** представляет собой упакованное представление двух десятичных цифр от 0 до 9 в одном байте. Каждая цифра хранится в своем полубайте.

Программное средство для работы с регистрами центрального процессора

Для работы с регистрами процессора используется входящая в состав операционной системы Windows программа DEBUG (отладчик). Эта программа позволяет осуществить побайтное тестирование и побайтное редактирование одного мегабайта памяти. Основные функции этой программы:

- загрузка файлов, содержащих программы или данные, в память компьютера;
- вывод на экран и корректировка содержимого регистров процессора;
- вывод на экран содержимого участков памяти в шестнадцатеричном коде и в символьном виде в соответствии с системой кодировки ASCII;
- изменение содержимого участков памяти и портов;
- перенос блоков данных из одного места основной памяти в другое;
- генерация и вывод на экран команд ассемблера по коду, загруженному в память компьютера;
- сохранение содержимого участков памяти на гибких и жестких дисках;
- выполнение с помощью встроенного калькулятора шестнадцатеричного сложения и вычитания.

Запуск программы DEBUG

Чтобы запустить программу DEBUG:

1. щелкните на кнопке Пуск;
2. выберите пункт Выполнить;
3. в диалоговом окне «Запуск программы» набрать имя программы (рис.8);
4. щелкнуть на кнопке ОК.

Или

1. щелкните на кнопке Пуск;
2. в строке запроса «Найти программы и файлы» набрать имя программы (рис.9)
3. из списка выбрать «debug»

После этого появится пустое черное окно, в верхней строчке которого будет высвечен символ«—», который является признаком готовности отладчика принять команду. DEBUG – это программа, работающая по принципу «команда – действие», т.е., чтобы произвести некоторую операцию отладчик должен получить соответствующую команду.

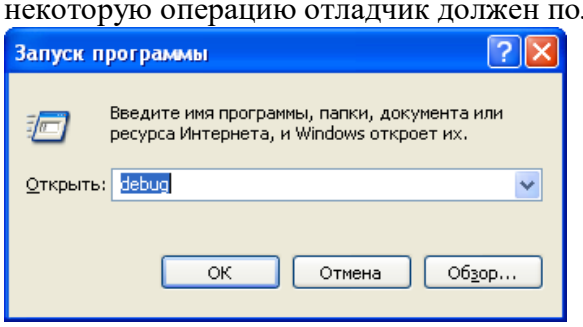


Рис.8.

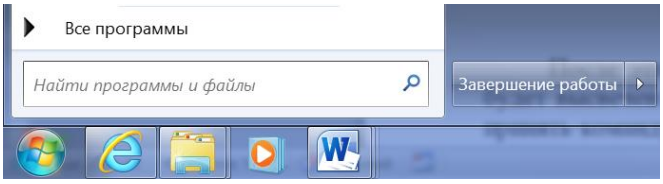


Рис.9.

Для управления процессом отладки в DEBUG применяется набор команд, список которых можно получить, введя команду помощи (символ «?»).Список поддерживаемых команд, приведен в таблице 3, окно отладчика, в котором выведен список команд, представлено на рис. 9.

Таблица 3

Команда	**Описание**	**Формат**
A (Assemble)	Транслирование команд ассемблера в машинный код; адрес по умолчанию - **CS:0100h**.	A [<адрес_начала_кода>]
C (Compare)	Сравнение содержимого двух областей памяти; по умолчанию используется **DS**.В команде указывается либо длина участков, либо диапазон адресов.	C <начальный_адрес_1> L<длина> <начальный_адрес_2> C <начальный_адрес_1> <конечный_адрес_1> <начальный_адрес_2>
D (Display/Dump)	Вывод содержимого области памяти в шестнадцатеричном и ASCII-форматах. По умолчанию используется **DS**; можно указывать длину или диапазон.	D [<начальный_адрес> [L<длина>]]D [начальный_адрес конечный_адрес]
E (Enter)	Ввод в память данные или инструкции машинного кода; по умолчанию используется **DS**.	E [<адрес> [<инструкции/данные>]]
F (Fill)	Заполнение области памяти данными из списка; по умолчанию используется **DS**. Использовать можно как длину, так и диапазон.	F <начальный_адрес_1> L<длина> '<данные>' F <начальный_адрес><конечный_адрес> '<данные>'
G (Go)	Выполнение отлаженной программы на машинном языке до указанной точки останова; по умолчанию используется **CS**. При этом убедитесь, что **IP** содержит	G [=<начальный_адрес>] <адрес_останова> [<адрес_останова> ...]

	корректный адрес.	
H (Hexadecimal)	Вычисление суммы и разности двух шестнадцатеричных величин.	H <величина_1><величина_2>
I (Input)	Считывание и вывод одного байта из порта.	I <адрес_порта>
L (Load)	Загрузка файла или данных из секторов диска в память; по умолчанию **-CS:100h**.Файл можно указать с помощью команды **N** или аргумента при запуске **debug.exe**.	L [<адрес_в_памяти_для_загрузки>] L [<адрес_в_памяти_для_загрузки> [<номер_диска><начальный_сек-тор><количество_секторов>]]
M (Move)	Копирование содержимого ячеек памяти; по умолчанию используется **DS**. Можно указывать как длину, так и диапазон.	M <начальный_адрес> L<длина><адрес_назначения> M <началь-ный_дрес><конечный_адрес> <адрес_назначения>
N (Name)	Указание имени файла для команд **L** и **W**.	N <имя_файла>
O (Output)	Отсылка байта в порт.	O <адрес_порта><байт>
P (Proceed)	Выполнение инструкций **CALL**, **LOOP**, **INT** или повторяемой строковой инструкции с префиксами **REP nn**, переходя к следующей инструкции.	P [=<адрес_начала>] [<количество_инструкций>]
Q (Quit)	Завершение работы **debug.exe**.	Q
R (Register)	Вывод содержимого регистров и следующей инструкции.	R <имя_регистра>
S (Search)	Поиск в памяти символов из списка; по умолчанию используется **DS**. Можно указывать как длину, так и диапазон.	S <начальный_адрес> L<длина>'<данные>' S <начальный_адрес><конечный_адрес> '<данные>'
T (Trace)	Пошаговое выполнение программы. Как и в команде **P**, по умолчанию используется пара **CS:IP**. Для выполнения прерываний лучше пользоваться командой **P**.	T [=<адрес_начала>] [<количество_выполняемых_команд>]
U (Unassemble)	Дизассемблирование машинного кода; по умолчанию используется пара **CS:IP**. К сожалению, **debug.exe** некорректно дизассемблирует специфические команды процессоров **80286+**, хотя они все равно выполняются корректно.	U [<начальный_адрес>] U [<начальный_адрес конечный_адрес>]

W (Write)	Запись файла из **debug.exe**; необходимо обязательно задать имя файла командой **N**, если он не был загружен. **А программы записываются только в виде файлов .COM!**	W [<адрес> [<номер_диска><начальный_сектор><количество_секторов>]]

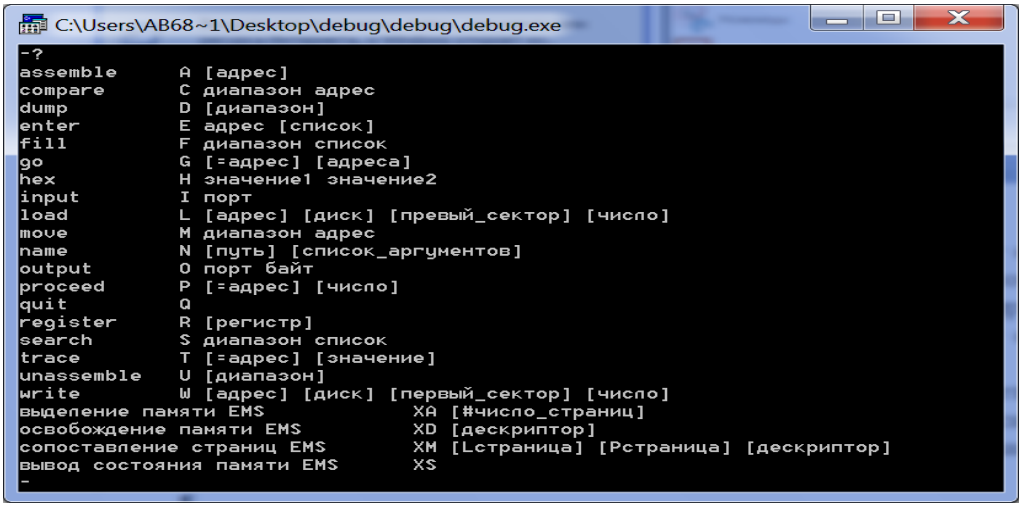


Рис.9.

Общими замечаниями по вводу команд отладчика является следующее:

- все команды начинаются с заглавной или строчной буквы;
- большая часть команд требует введения дополнительных параметров, часть из которых является необязательными;
- если два подряд расположенных параметра являются числами, то они разделяются пробелом или запятой (в противном случае параметры можно не отделять один от другого);
- все числа должны вводиться в шестнадцатеричном представлении;
- некоторые команды принимают в качестве параметра адрес, который может вводиться в двух формах: полный логический адрес – два шестнадцатеричных числа, записанные через двоеточие (первое число – сегментная компонента логического адреса, второе число – смещение) и короткий адрес – одно шестнадцатеричное число (смещение);
- при указании полного логического адреса допускается в качестве сегментной компоненты приводить имя сегментного регистра, из которого данная компонента будет выбираться.

С точки зрения изучения архитектуры и организации ЭВМ, набор команд, предлагаемый отладчиком, является избыточным. Минимально необходимый набор включает команды:

–(A)SSEMBLE – ассемблирование;
– (U)NASSEMBLE – дизассемблирование;
– (E)NTER – ввод данных в память;
– (D)UMP – вывод содержимого участка памяти на экран;
– (R)EGISTER – просмотр и изменение содержимого регистров;
– (T)RACE – пошаговое выполнение программы;
– (N)AME – задание имени файла программы;
– (L)OAD – загрузки файла в память;
– (W)RITE – запись области памяти в файл;
– (Q)UIT – выход из отладчика.

Перечисленные команды представляют определенный интерес. Часть рассмотрим подробнее сейчас, остальные - в следующих лабораторных работах.

Получение и корректировка содержимого регистров

Команда register**(r)** позволяет выводить на экран и корректировать значения регистров и флагов состояния процессора. Эта команда также выдает информацию о следующей выполняемой команде. После ввода команды **r** и нажатия клавиши [Enter] на экране появляется следующее содержание окна программы (рис.10).

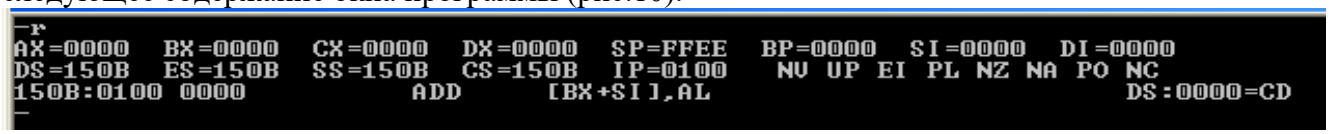


Рис. 10

В двух первых строчках представлены шестнадцатеричные значения регистров процессора:

- регистры сегментов (DS, ES, SS, CS) содержат одно и то же значение - 150B (у вас это может быть и другое число),
- значение регистра SP равно FFEE,
- значение регистра IP — 0100,
- в остальных регистрах записаны нули.
- в правой части второй строки находятся значения восьми флагов состояния процессора (см. табл.2).

При загрузке отладчика все флаги, кроме IF, сбрасываются. Данное состояние регистров устанавливается всегда после загрузки программы DEBUG (отличие может быть только в значении сегментных регистров, которое указывает на адрес ближайшего свободного сегмента памяти).

В третьей строке представлена справочная информация о выполняемой команде. Как известно, выполняемая программа представляет собой последовательность кодов машинных команд.

Например: Адрес машинного кода команды, которая будет выполняться следующей, фиксируется в регистрах CS и IP. В нашем примере — это адрес CS:IP = 150B:0100. Этот адрес и выведен в начале третьей строки. В ней представлена и другая информация об этой команде. Так, следующий параметр — это машинный код команды, содержащийся в поле памяти, которое начинается с адреса, указанного в начале строки.

С помощью команды **r** можно изменить значение любого регистра. Для этого в командной строке необходимо указать также имя изменяемого регистра. Отладчик в ответ выведет имя регистра, его значение, а в следующей строке — двоеточие, приглашающее ввести новое значение регистра

Например: -r cx
CX 0000
:

Для ввода нового значения в регистр, например, значения 89AB, необходимо после двоеточия указать это значение:

-r cx
CX 0000
: 89AB

Если вновь воспользоваться командой **-r** без параметров, то можно убедиться, что требуемое изменение выполнено.

Команда **r** с параметром **f** выводит на экран флаги состояния процессора:

Например: - rf
NV UP EI PL NZ NA PO NC -

Получив значения флагов, их можно изменить. Для этого вводится одно или несколько новых значений. Новые значения вводятся в любом порядке через пробел или без разделителя.

Например: - rf
NV UP EI PL NZ NA PO NC -OV NG CY

Завершение работы программы Debug

Чтобы выйти из отладчика и передать управление операционной системе, на его стандартный запрос вводится команда **q**:

-q

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом.
2. В отчете дать определение центральному процессор и его основным характеристикам.
3. Описать логическую и физическую схемы устройства центрального процессора.
4. Используя программу «Сведения о системе» или аналогичную, определить модель, производителя и основные характеристики процессора, установленного на ПК.
5. С помощью справочной системы Window sизучить систему команд отладчика.
6. Запустить отладчик и установить следующие значения регистров (в шестнадцатеричной системе счисления):
    a) DS = 1000 + N
    b) SS = 1000;
    c) BP = A - N;
    d) SI = 10 - 2 * N;
    e) DI= 1A+ 3 * N;
    f) CX = первый символ вашей фамилии;
    g) используя регистры AX и DX разместить число N4,7N

где N — вариант.

4. Установитm флаги знака, нуля и четности.

5. С помощью команды r убедиться, что установка значений регистрови флагов выполнена верно.

6. Выйти из программы отладчика.

Контрольные вопросы

1. Что такое центральный процессор?
2. Какова логическая структура процессора?
3. Перечислите основные блоки процессора
4. Назовите основные характеристики процессора
5. Для чего в состав процессора включены регистры?
6. Каково назначений каждого регистра?

Лабораторная работа 3. Исследование методов адресации

Цель работы: изучить способы адресации процессора и их представление в кодах команд. Научиться размещать данные в ячейках ОЗУ, производить обмен данными между процессором и памятью, используя наиболее важные методы доступа к памяти (методы адресации). Используя программу DEBUG научиться ассемблировать и трассировать инструкции центрального процессора.

Основные теоретические сведения

Лишь очень небольшое количество данных может быть обработано внутренними средствами процессора с помощью его немногочисленных регистров. Большая же часть задач требует запоминания данных (исходных величин, промежуточных и окончательных результатов) в памяти. Поэтому способы обмена информацией с памятью являются одной из основных вопросов архитектуры центрального процессора и ЭВМ в целом.

Краткие сведения об устройстве оперативной памяти, и реальном режиме работы центрального процессора с оперативной памятью

Технологии организации оперативной памяти

Оперативная память является одним из важнейших элементов компьютера. Именно из нее процессор берет программы и исходные данные для обработки, в нее он записывает полученные результаты. Название «оперативная» эта память получила потому, что она работает очень быстро, так что процессору практически не приходится ждать при чтении

данных из памяти или записи в память. Однако содержащиеся в ней данные сохраняются только пока компьютер включен или до нажатия кнопки сброса (Reset). При выключении компьютера содержимое оперативной памяти стирается.

Оперативная память - RAM (Random Access Memory, то есть память с произвольным доступом). Т.е., обращение к данным, хранящимся в оперативной памяти, не зависит от порядка их расположения в памяти. Физически оперативная память организована в виде микросхем, устанавливаемых на специальных печатных платах – линейках или ОЗУ может изготавливаться как отдельный блок или входить в конструкцию, например однокристальной ЭВМ или микроконтроллера. Крупнейшими производителями памяти являются Samsung, Toshiba и Hitachi и др..

Кратко история появления и развития данного устройства:

1. В 1834 году Чарльз Бэббидж начал разработку Аналитической машины. Одна из важных частей этой машины называлась «Склад» (store), и предназначалась для хранения промежуточных результатов вычислений. Результаты запоминались с использованием валов и шестерней.
2. ЭВМ первого поколения можно считать ещё экспериментальными, поэтому в них использовалось множество разновидностей запоминающих устройств:
    a. на ртутных линиях задержки,
    b. электронно-лучевых и электростатических трубках,
    c. магнитный барабан: он обеспечивал достаточное для компьютеров тех времён быстродействие и использовался в качестве основной памяти для хранения программ и вводимых данных.
3. Второе поколение требовало более технологичных в производстве схем оперативной памяти. Наиболее распространённым видом памяти в то время стала память на магнитных сердечниках.
4. Начиная с третьего поколения большинство узлов компьютеров стали выполнять на микросхемах, в том числе и оперативную память. Наибольшее распространение получили два вида ОЗУ: на основе конденсаторов (динамическая память) и триггеров (статическая память).

Память статического типа (SRAM) - ОЗУ, собранное на триггерах. Достоинство этого вида памяти — скорость. Поскольку триггеры собраны на вентилях, а время задержки вентиля очень мало, то и переключение состояния триггера происходит очень быстро. Недостатки: высокая стоимость; группа транзисторов, образующих триггер, занимает много места. Используется для организации сверхбыстрого ОЗУ, критичного к скорости работы.

Память динамического типа - DRAM это экономичный вид памяти. Для хранения разряда (бита) используется схема, состоящая из одного конденсатора и одного транзистора (в некоторых вариациях конденсаторов два). Такой вид памяти решает проблему дороговизны и компактности. Минусы: более низкая производительность (для того чтобы установить в единицу один разряд памяти конденсатор нужно зарядить, а для того чтобы установить в ноль - разрядить. Это гораздо более длительные операции (в 10 и более раз), чем переключение триггера); конденсаторы склонны к «стеканию» заряда, при этом, чем меньше их ёмкость, тем быстрее они разряжаются. В связи с этим обстоятельством, заряд конденсаторов для восстановления необходимо «регенерировать» через определённый интервал времени. Регенерация выполняется центральным микропроцессором или контроллером памяти, за определённое количество тактов считывания при адресации по строкам. Для регенерации памяти периодически приостанавливаются все операции с памятью.

В настоящее время DRAM используется в оперативной памяти компьютера, а SRAM – для создания высокоскоростной Кэш памяти.

Реальный режим работы центрального процессора

Известно, что все данные и программы их обработки хранятся в компьютере в ***дискретном двоичном*** виде. Согласно классическим принципам, минимально возможным объемом информации является 1 бит, и именно бит служит основой компьютерной памяти, ее минимальным «конструктивным элементом». Бит — слишком маленькая единица информации и обеспечивать доступ к каждому отдельному биту памяти нецелесообразно. Минимальной единицей обмена информацией с памятью в современных компьютерах является 1 байт. Каждый байт имеет свой идентификационный номер, по которому к нему можно обращаться — ***физический адрес***. Адреса соседних байтов отличаются на единицу. Физический адрес используется для получения доступа к конкретной ячейке памяти. Для работы с памятью используются шина адреса и шина данных (рис.1).Именно эта информация выставляется центральным процессором на шину адреса. Практически при обращении к памяти задается адрес начального байта и их требуемое количество.

Физически память устроена таким образом, что возможно обращение к отдельным байтам, 16-разрядным словам (два смежных байта памяти), 32-разрядным двойным словам (четыре смежных байта памяти). Максимальный размер данных определяется разрядностью шины данных и режимом работы процессора. В современных персональных компьютерах шина адреса 32-разрядная, но в реальном (16-разрядном) режиме работы процессора максимальный размер обрабатываемых данных – 16 бит, т.е. слово.

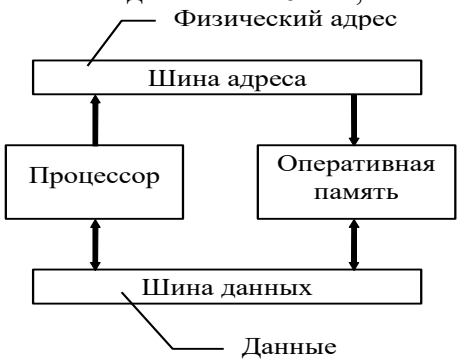


Рис. 1

Разрядность шины адреса определяет максимальный объем оперативной памяти. В современных персональных компьютерах шина адреса так же 32-разрядная, и, следовательно, процессор имеет возможность одновременно обращаться к 2^{32} = 4 Гбайт оперативной памяти.

Однако в реальном режиме работы из шины адреса используется только 20 младших разрядов. Таким образом объем адресуемой памяти в реальном режиме работы процессора составляет 2^{20} = 1 Мбайт.

Так как адреса принято записывать в шестнадцатеричной системе счисления, то диапазон физических адресов для 20-разрядной шины адреса выглядит следующим образом:

$$00000h \le [\text{физический адрес}] \le FFFFFh.$$

Здесь и далее буква «h», записанная после числа, подчеркивает, что данное число записано в шестнадцатеричной системе счисления.

Однако все регистры процессора в реальном режиме 16-разрядные. Возникает проблема представления 20-разрядного физического адреса памяти при помощи 16-разрядных регистров.

Для разрешения этой проблемы используется двухкомпонентный логический адрес. ***Логический адрес*** состоит из 16-разрядных компонент: компоненты сегмента памяти и компоненты смещения внутри сегмента.

Для получения 20-разрядного физического адреса к сегментной компоненте (сегменту) приписываются справа четыре нулевых двоичных разряда (для расширения до 20 разрядов), затем полученное число складывается с компонентой смещения (рис.2). Перед сложением к компоненте смещения (смещению) слева дописываются четыре нулевых бита (так же для расширения до 20 разрядов).

Логический адрес принято записывать в форме *<сегмент : смещение>*.

Например, пусть есть логический адрес 1234h:0123h. Сегментная компонента равна 1234h, а компонента смещения – 0123h. Вычислим физический адрес, соответствующий логическому адресу:

- расширяем до 20 бит сегментную компоненту, дописывая справа 4 нулевых бита, получаем число 12340h (здесь четыре двоичных разряда представляются одним шестнадцатеричным разрядом);
- расширяем до 20 бит компоненту смещения, дописывая справа 4 нулевых бита, получаем число 00123h;
- для получения физического адреса складываем полученные числа:12340h + 00123h = 12453h.

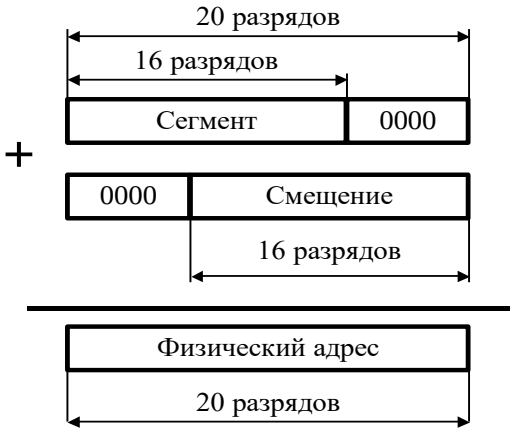


Рис. 2

Одному физическому адресу может соответствовать несколько логических. Например, полученному физическому адресу 12453h соответствует логический 1245h:0003h.

Фактически в схеме адресации памяти реального режима первый мегабайт памяти как бы разбивается на сегменты. Физический адрес начала сегмента (базовый адрес сегмента) равен расширенному до 20 бит сегментной компоненте адреса.

Сегмент - любой участок памяти размером до 64 Кб и с начальным физическим адресом, кратным 16. Поэтому он может начинаться только с границы параграфа (один параграф – это 16 идущих подряд байта). Компонента смещения при такой схеме адресации является смещением внутри сегмента памяти, а сам сегмент задается сегментной компонентой адреса (рис. 3).

Логический адрес в реальном режиме должен находиться в следующих пределах:

0000h:0000h ≤ [логический адрес] ≤ FFFFh:000Fh.

Для хранения начального адреса сегмента применяются сегментные регистры процессора. Процессор обеспечивает доступ к четырем сегментам одновременно. Эти сегменты называются сегментом кода, сегментом данных, сегментом стека и дополнительным сегментом данных.

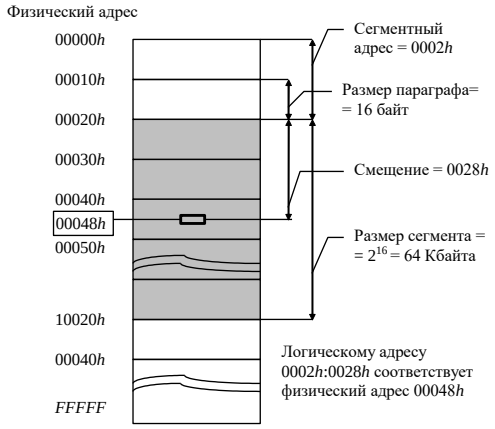


Рис. 3

Сегмент кода содержит команды программы. Для доступа к этому сегменту используется сегментный регистр **CS**. Он содержит адрес сегмента с машинными командами.

Сегмент данных содержит обрабатываемые программой данные. Для доступа к этому сегменту служит сегментный регистр **DS**, который хранит адрес сегмента данных текущей программы.

Сегмент стека – этот сегмент представляет собой область памяти, называемую стеком. Работу со стеком микропроцессор организует по следующему принципу: последний записанный в эту область элемент выбирается первым. Для доступа к этому сегменту служит сегментный регистр **SS**, содержащий адрес сегмента стека.

Дополнительный сегмент данных применяется в некоторых командах для организации обмена информации между этим сегментом и сегментом данных. Адрес дополнительного сегмента данных должен содержаться в сегментном регистре **ES**.

Компонента смещения логического адреса текущей исполняемой команды всегда располагается в указателе команд IP, а компонента смещения логического адреса вершины стека в регистре SP. Компонента смещения логического адреса ячейки памяти, рассматриваемой как данные, может располагаться в регистрах BX, BP, SI, DI.

Таким образом, реальный режим работы микропроцессора характеризуется следующим:

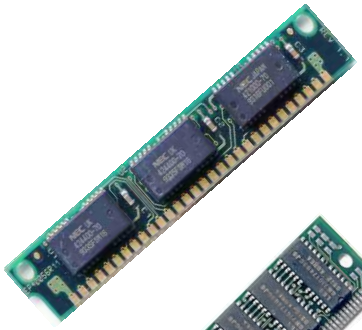
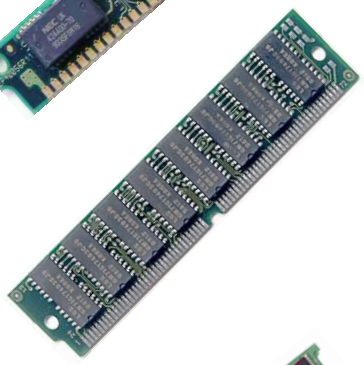
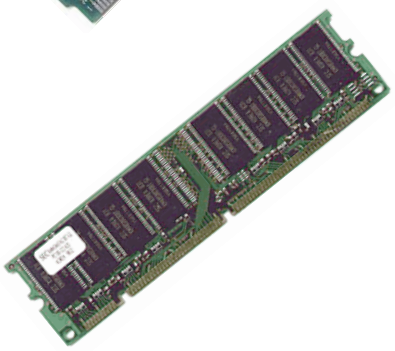
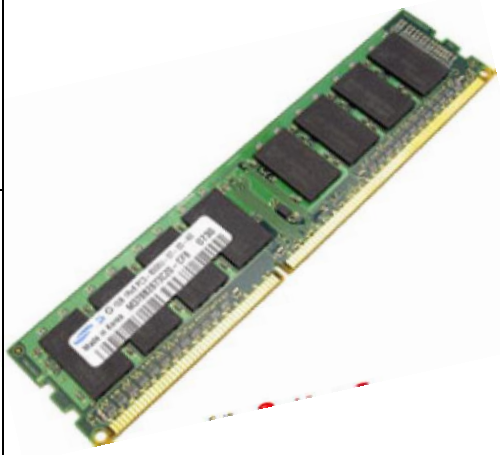
- Пространство оперативной памяти делится на сегменты по 64 Кб., сегменты в памяти могут перекрываться;
- Страничное преобразование адреса запрещено, то есть физический адрес равен линейному и формируется как сумма двух составляющих:
  - 16-разрядного эффективного адреса, который, в свою очередь, является суммой трех составляющих: базы, смещения и индекса;
  - 20-разрядного результата сдвига содержимого конкретного сегментного регистра на 4 разряда влево;
- Максимальное значение физического адреса равно 0fffffh, то есть 1 Мбайт, но, фактически, в реальном режиме микропроцессор адресуется на 64 Кбайт больше.

В реальном режиме схема распределения оперативной памяти - фиксированная:

1. В диапазоне адресов 00000h - 003ffh (1-ый Мб ОЗУ) находится ***таблица векторов прерываний***. Адрес программы прерывания называется вектором. таблица содержит 256 векторов размером 4 байта (указателей на программы обработки прерываний). В первом слове хранится значение IP, а во втором - CS;
2. Сразу за таблицей векторов прерываний В диапазоне адресов 00400h - 006ffh располагается область памяти, содержащая жестко структурированные данные, обеспечивающие работу BIOS и ОС;

3. С адреса 0b8000h располагается область видеопамяти, в которой формируется изображение, видимое на экране.

Типы микросхем памяти

Тип памяти	*Краткая характеристика*	*Вид*
FPM DRAM (Fast Page Memory - FPM). память с быстрым постраничным режимом	Появилась в конце 80-х годов XX века. Выпускалась на модулях с 30 или 72 контактами. Использовалась в компьютерах PC 386, 486 и Pentium (1987–1995 г.). Особенность - поддержка разбиения всего адрес-ного пространства на страницы и поддержка пакетного режима. (стра-ница, рассматривалась как строка матрицы, а адреса ячеек – столбцы.	
EDO DRAM (Extended Data Out - память с расширенным выводом).	Появилась в 1995 года в компьюте-рах на основе Pentium. Выпускалась в виде 72-контактных модулей SIMM. Усовершенствованный тип FPM, реализующий конвейерную выбор-у данных. Емкость линеек – 32 Мб	
SDRAM Synchronous DRAM (синхронная динамическая память)	Появилась в 1998 г. Выполнена на модулях DIMM с 168 контактами. Синхронизирована с системным тай-мером, управляющим ЦП, уменьша-ются временные задержки в процес-се циклов ожидания и ускоряется поиск данных. Синхронизация поз-воляет контроллеру памяти точно знать время готовности данных. Скорость доступа увеличивается благодаря тому, что данные дос-тупны во время каждого такта таймера	
SDRAM II (или DDR) (Double Data Rate – удвоен-ная скорость передачи данных)	Появилась в 2000 г. Выпускается в виде 184-контактных DIMM моду-лей. Основана на тех же принципах, что и SDRAM, но имеет удвоенную скорость передачи данных. Это достигается за счет передачи данных дважды за один цикл: первый раз — в начале цикла, а второй — в его конце. Кроме того, имеется буфер промежуточного хранения данных. Емкость линеек 128 Мб – 1 Гб	
DDR2 SDRAM (Double-Data-Rate – синхрон-ная динамичес-кая память с произвольным	Появились в 2003 г. Модули DDR2 изготовляются в новом корпусе типа BGA (FBGA), имеют 240 контакта (по 120 с каждой стороны). Исполь-зует передачу данных по обоим срезам тактового сигнала, Основное отличие DDR2 - вдвое	

доступом и удвоенной ско-ростью переда-чи данных 2 поколения)	большая час-тота работы шины, по которой дан-ные передаются в буфер микросхе-мы памяти. Емкость линеек 512 Мб - 4 Гб
DDR3	Появилась в 2007 г. Линейки имеют 240 контактов. Функционируют при напряжении питания 1,5 В. Имеют более высокую частоту работы, чем DDR2 за счет 8-битного буфера предварительной выборки. Размет страницы 1 Кб.
GDDR (graphics double data rate memory).	Это подвид энергозависимой дина-мической памяти, которая предназ-начена для использования в видео-картах. Технология аналогична DDR, но имеет более низкое потребление, и применяются специальные методы управления буфером ввода-вывода, для улучшения пропускной способности. Впервые появился в 2000 г. На сегодняшний день разра-ботано несколько видов памяти GDDR: GDDR2, GDDR3 GDDR4 GDDR5.

Методы адресации

Набор команд процессоров обеспечивает выполнение операций над операндами, которые находятся в регистрах, памяти и непосредственно в команде. В набор входят безадресные, которые не имеют операндов, одно- и двухадресные команды (по количеству операндов). Для одноадресных команд операнд может располагаться в регистре или в памяти. Для двухадресных команд возможны следующие схемы расположения операндов:

– регистр — регистр;
– регистр — память;
– память — регистр;
– регистр — непосредственные данные;
– память — непосредственные данные.

Существует несколько способов задания операндов команд процессора, которые обычно называют ***методами адресации***. Эти способы содержат конкретное указание процессору откуда брать или куда помещать данные. В частности для операндов, находящихся в памяти, режимы адресации определяют правила формирования логического адреса данных.

Рассмотрим адресацию простых данных. Под простыми данными принято понимать такие, которые хранят в себе только одно значение, например, целое число (целые 16-разрядные числа).Простейшие и очень часто используемые инструкции обработки данных, которые выполняются в регистрах микропроцессора, естественны и легко понимаются на интуитивном уровне. Полный перечень методов адресации на примере команды MOV (пересылки данных) представлен в таблице 1:

Таблица 1

Методы адресации реального режима

№	Название метода	Вычисление адреса	Операнд	Пример
1	Регистровая адресация	-	Содержимое регистра	MOVAX, BX
2	Непосредственная	-	Константа в команде	MOVAX, 30

3	Прямая	EA = Disp	Данные из памяти, из команды	MOVAX, [30]
4	Косвенная регистровая	EA = Base	Данные из памяти, из регистра	MOVAX, [BX]
5	Базовая	EA = Base + Disp	Данные из памяти, равному содержимому базового регистра + смещение	MOV AX, [BX+30] (или: [BX]+30, 30[BX])
6	Индексная	EA = Index + Disp	Данные из памяти, равному содержимому индексного регистра + смещение	MOV AX, 30 [SI] (или [SI+30])
7*	Масштабированная индексная	EA = Scale * Index + Disp	См. метод 6, но индекс предварительно умножается на масштаб	MOVAX, 30 [esi*2]
8	Базово- индексная	EA = Base+ Index	Данные из памяти, равному сумме двух регистров	MOV AX, [BX+SI] (или [BX] [SI])
9*	Масштабированная базово-индексная	EA = Base + Scale * Index	См. метод 8, но индекс предварительно умножается на масштаб	MOV AX, [EBX + ESI *2]
10	Базово-индексная со смещением	EA = Base + Index + Disp	Данные из памяти, равному сумме двух регистров и смещения	MOV AX, [BX + SI + 30] (или: [BX+30] [SI])
11*	Масштабированная базово-индексная со смещением	EA = Base + Scale * Index + Disp	См. метод 10, но индекс предварительно умножается на масштаб	MOV AX, [EBX + ESI*2 + 30]
12	Стековая (неявная адресация)	EA = SP – 2 EA = SP	Содержимое регистра Содержимое стека	PUSH AX POP AX

* - масштабирование индекса возможно только в 32 битовом режиме.

Обозначения:

- EA — эффективный адрес;
- Disp— смещение;
- Base — база;
- Index— индекс;
- Scale — масштаб (количество байт в каждом из данных).

Непосредственная адресация

В этом случае процессор выбирает данные непосредственно из самого кода команды. Этот метод адресации применяется для задания константных значений в качестве операнда – источника данных (только в двухадресных командах). При этом разрядность непосредственного операнда определяется разрядностью операнда – приемника данных.

Например, первый операнд (приемник данных) команды MOV AX,34h имеет регистровую адресацию, т.е. результат ее выполнения помещается в регистр (в данном случае AX), а второй операнд (источник данных) имеет непосредственную адресацию, т.е. является константой 0034h и располагается в коде этой команды. Таким образом данная команда помещает константу 0034h в регистр AX.

Регистровая адресация

В данном случае процессор принимает данные из регистра или помещает данные в регистр. При этом данные могут быть восьмиразрядными (в этом случае используются 8 восьмиразрядных регистров общего назначения: AL, AH, BL, BH, CL, CH, DL, DH), а также

шестнадцатиразрядными (используются 8 шестнадцатиразрядных регистра общего назначения: AX, BX, CX, DX, SP, BP, SI, и DI).

Прямая адресация

При этой адресации операнд располагается в оперативной памяти, а адрес операнда (точнее компонента смещения логического адреса) располагается в коде команды. Для обращения к такому операнду процессор формирует физический адрес, выбирая компоненту сегмента из сегментного регистра DS. Поскольку обращение к операнду требует от процессора выполнения определенных действий, компоненту смещения адреса операнда принято называть исполнительным адресом.

Обычно прямая адресация применяется, если операндом является метка (имя ячейки памяти). Ассемблер, встроенный в debug, не поддерживает метки, и при применении прямой адресации необходимо прямо указывать смещение конкретной ячейки памяти. Чтобы различать непосредственные данные и исполнительный адрес последний берется в квадратные скобки.

Например, команда MOV AX,[0034h] в отличие от предыдущего случая помещает в регистр AX не константу 0034h, а содержимое слова, находящегося по адресу DS:0034h.

Косвенная адресация

Как и в случае прямой адресации, операнд, имеющий косвенную адресацию располагается в памяти, однако исполнительный адрес операнда находится не в коде команды, а в одном из базовых или индексных регистров: BX, BP, SI и DI. При вычислении физического адреса операнда процессор использует содержимое сегментного регистра DS, если исполнительный адрес располагается в BX, SI, либо DI, и содержимое сегментного регистра SS, если исполнительный адрес находится в регистре BP.Чтобы различать регистровую и косвенную адресацию, при косвенной адресации имя регистра заключается в квадратные скобки.

Например, команда MOV [BX], AX, в которой первый операнд имеет косвенную адресацию, а второй регистровую, помещает содержимое регистра AX в ячейку памяти с адресом DS:BX.

Адресация по базе

Этот метод адресует данные в памяти. При адресации по базе процессор вычисляет исполнительный адрес с помощью сложения значения сдвига, который находится в коде команды с содержимым регистров BX или BP.

Данный метод адресации удобно использовать при доступе к структурированным записям данных, расположенным в разных областях памяти. В этом случае базовый адрес записи помещается в базовый регистр и доступ к ее отдельным элементам осуществляется по их сдвигу относительно базы. А для доступа к разным записям одной и той же структуры достаточно соответствующим образом изменить содержимое базового регистра.

Например, команда MOV byte ptr [BP + 4], 3Fh, в которой первый операнд имеет адресацию по базе и размещен в регистре BP, а сдвиг поля относительно этой базе, равный 4h, размещен в коде команды. Второй операнд имеет непосредственную адресацию, помещает значение константы 3Fh в ячейку памяти, имеющую адрес SS:BP + 4h.

Для указания того, что в данном случае работа выполняется именно с байтами, используется префикс byte ptr (word ptr префикс используется для указания того, что действия производятся над словами). Первый операнд этой команды можно трактовать как поле некоторой записи, располагаемой в сегменте стека.

Индексная адресация

Этот метод адресует данные в памяти. В этом случае исполнительный адрес вычисляется как сумма значений сдвига, который находится в коде команды, и индексного регистра (DI или SI). Этот режим адресации удобен для доступа к элементам массива, когда сдвиг указывает на начало массива, а индексный регистр – на его элемент.

Индексная адресация отличается от базовой только используемыми регистрами и трактовкой содержимого регистра и сдвига.

Базово-индексная адресация

Как и в предыдущих случаях адресуется операнд в памяти. При базово-индексной адресации исполнительный адрес вычисляется как сумма значений базового регистра, индексного регистра и, возможно, сдвига, который находится в коде команды. Так как в этом режиме адресации складывается два отдельных смещения, то он удобен при адресации двумерных массивов, когда базовый регистр содержит начальный адрес массива, а значения сдвига и индексного регистра суть смещения по строке и столбцу. Возможна и другая трактовка базово-индексной адресации. Применение то же – для обращения к элементам двумерного массива, но начальный исполнительный адрес задается сдвигом, а базовый и индексный регистры задают смещения в массиве по строке и столбцу.

Например, команда MOV AL, [BX+SI+006Dh] имеет два операнда, первый из которых, – регистр AL, а второй – байт, расположенный в памяти. Для обращения ко второму операнду применяется базово-индексная адресация. Данный байт можно рассматривать как элемент двумерного массива (матрицы), который расположен в сегменте данных со смещением от начала 006Dh. Регистр BX задает смещение строки от начала матрицы, а регистр SI – смещение элемента относительно начала строки. Таким образом, данная команда помещает в AL содержимое байта, расположенного по адресу DS:BX + SI + 006Dh.

В данном примере для обращения к произвольным элементам матрицы, имеющей размер n × m, содержимое BX следует изменять от нуля (первая строка матрицы) до (n – 1) *· m (последняя строка матрицы) с шагом m (количество элементов в строке), а содержимое SI – от нуля (первый элемент строки) до m – 1 (последний элемент строки) с шагом 1 (размер элемента, равный 1 байту).

Команды программы DEBUG для изучения методов адресации

Команда ассемблирования

(перевод мнемокода ассемблера в машинный код)

Отладчик DEBUG можно использовать для введения команд ассемблера непосредственно в память машины. Команду ASSEMBLE можно использовать при составлении коротких инструкций на ассемблере, а также при внесении изменений в существующие программы. Эта команда позволяет вводить мнемокод ассемблера непосредственно в память, избавляя от необходимости трансляции. Вводимый текст не может включать метки перехода в чистом виде.

При введении команды, необходимо набрать «**a**» или «A» и, через пробел, необязательный параметр – адрес первой команды загружаемой программы.

Если указан короткий адрес, то адрес сегмента выбирается из регистра CS. Если адрес не задан вообще, то машинный код будет помещаться в память, начиная с того места, где закончилась обработка предыдущей командой ASSEMBLE. Если после старта отладчика команда вводится в первый раз и в командной строке отсутствует начальный адрес, то размещение машинного кода производится с адреса CS:0100.

После введения команды ассемблирования на экране появляется начальный адрес. Это сигнал на введение первой команды программы. Если команда введена без ошибок, на экран выдается адрес следующей команды и отладчик опять переходит в режим ожидания. В случае ошибки отладчик обозначает ее месторасположение. Если введены все команды программы, то нажимается Enter – команда ASSEMBLE заканчивает работу и возвращает управление отладчику (рис.4).

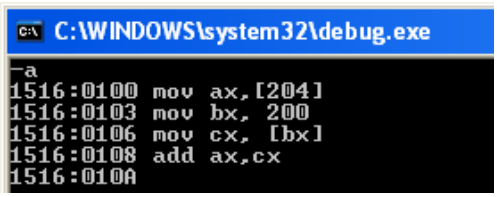


Рис.4.

Команда дизассемблирования
(перевод машинного кода в мнемокод ассемблера)

Команда UNASSEMBLE служит для перевода машинного кода на язык ассемблера. При введении команды необходимо набрать «**u**» или «**U**» и, через пробел, необязательные параметры – начальный адрес обрабатываемого кода, конечный адрес обрабатываемого кода или его размер.

В командной строке UNASSEMBLE можно не указывать начальный адрес обрабатываемого кода. Если указан короткий адрес, то адрес сегмента выбирается из регистра CS. Если адрес не задан вообще, то машинный код обрабатывается с того места, где закончилась обработка предыдущей командой UNASSEMBLE. Если после старта отладчика команда вводится в первый раз и в командной строке отсутствует начальный адрес, то обработка машинного кода производится с адреса CS:0100.

Обрабатываемый участок памяти можно определить начальным и конечным адресами. При этом, в не зависимости от формы начального адреса, конечный адрес должен быть коротким.

Другой вариант задания обрабатываемого участка памяти – задание его начального адреса и размера. Чтобы отличить размер от короткого конечного адреса перед ним вводится символ «L».

Если размер участка памяти, обрабатываемой командой UNASSEMBLE, не определен, то по умолчанию длина обрабатываемого участка равна 32 байтам.

Результатом выполнения команды дизассемблирования является листинг программы, сгруппированный в три колонки (рис.5). В листинге слева (первая колонка) указывается полный логический адрес команды. Затем (вторая колонка) – значение составляющих команду байтов в машинном коде. В третьей колонке находится соответствующая этому коду инструкция ассемблера.

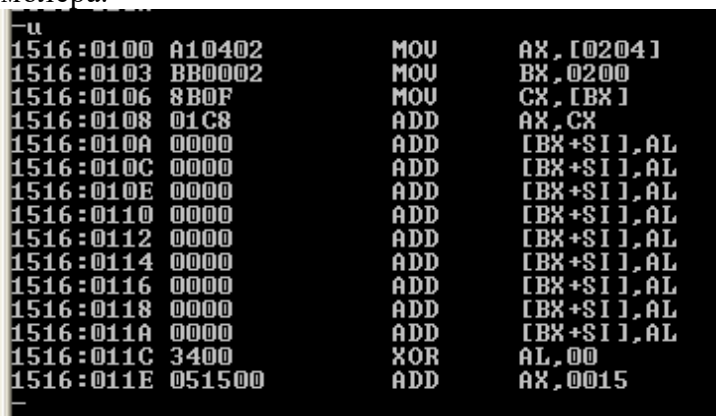


Рис.5.

Команды ввода и вывода данных в память/из памяти

Ввод данных осуществляется с помощью команды ENTER. Эта команда позволяет побайтно корректировать содержимое памяти. При введении команды необходимо набрать «**e**» или «**E**» и адреса первого байта корректируемого блока. Если указан короткий адрес, то адрес сегмента выбирается в регистре DS.

Вводимые данные также включаются в командную строку. Они представляют собой последовательность чисел в шестнадцатеричном представлении и/или символьных значений, разделенных пробелом или запятой. Символьные значения заключаются в апострофы (рис.6).

Рис.6.

Команда ENTER может использоваться для отображения и, в случае необходимости, корректировки значения конкретного байта. В этом случае после команды необходимо

указать адреса требуемой ячейки. При введении команды на экране появляется адрес байта и его значение:

-e 300

0958:0300 20.

При нажатии на клавишу пробела на экране появляется значение следующего байта:

-e 300

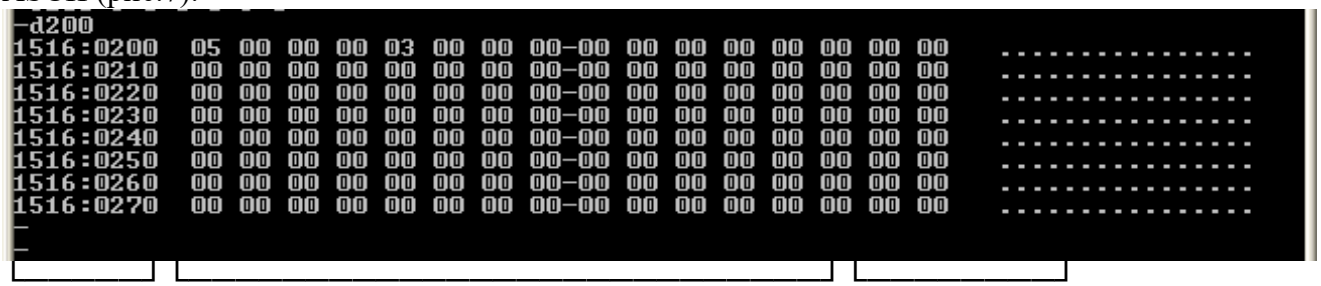
0958:0300 20. 2A.

Значение байта можно изменить. Для этого вводится новое шестнадцатеричное число. Однако символьные переменные в этом случае вводить нельзя.

Чтобы завершить выполнение команды следует нажатьEnter. Появление дефиса (-) – стандартного запроса отладчика, свидетельствует о его готовности принять следующую команду.

Команда вывода содержимого участка памяти на экран

Команда DUMP: «**d**» или «**D**», служит для отображения на экране содержимого участка памяти. Полученный фрагмент содержимого памяти – дамп, представляет собой последовательность значений байтов в шестнадцатеричном представлении, а также – в коде ASCII (рис.7).



Адрес Значения в шестнадцатеричном виде Значения в виде символов

Рис.7.

Числа в первом столбце, разделенные двоеточием – это полный логический адрес памяти, следующие 16 столбцов – значения, последовательно содержащиеся в памяти, начиная с этого адреса (в шестнадцатеричном виде), и, наконец, строка из 16 символов – символьное представление этих значений.

Значения, не имеющие символьного представления в коде ASCII, обозначаются символом "точка". В рассмотренном примере изображены только точки, поскольку в коде ASCII не существует печатных символов со значением 00. Дамп отображает содержимое 128 последовательно расположенных ячеек памяти. В приведенном выше примере начальный адрес дампа – 1516:0100, конечный – 15161:0270.

Если в команде «**d**» не указывать параметров, DEBUG будет последовательно выводить по 128 байтов памяти. То есть начальный адрес дампа будет на единицу превышать конечный адрес дампа, полученного при введении предыдущей команды «d». Если команда вводится первоначально, то дамп выводится, начиная с адреса, по которому был загружен обрабатываемый файл.

Команда DUMP может использоваться с параметрами, которые задают начальный адрес дампа и его размер. Использование параметров аналогично использованию параметров в команде UNASSEMBLE. За тем исключением, что, если начальный адрес дампа задан в коротком формате, то сегментная компонента адреса выбирается из регистра DS.

Команда пошагового выполнения программы

Команда TRACE: «**t**» или «**T**» – трассировка осуществляет пошаговое выполнение инструкций программы в машинном коде. При трассировке после выполнения каждой команды производится останов работы программы и на экран выводятся регистры и флаги состояния процессора. Содержание окна отладчика аналогично, выполнению команды

REGISTER. Разница заключается только в том, что при введении TRACE перед появлением картинки, выполняется одна команда отлаживаемой программы (рис.8).

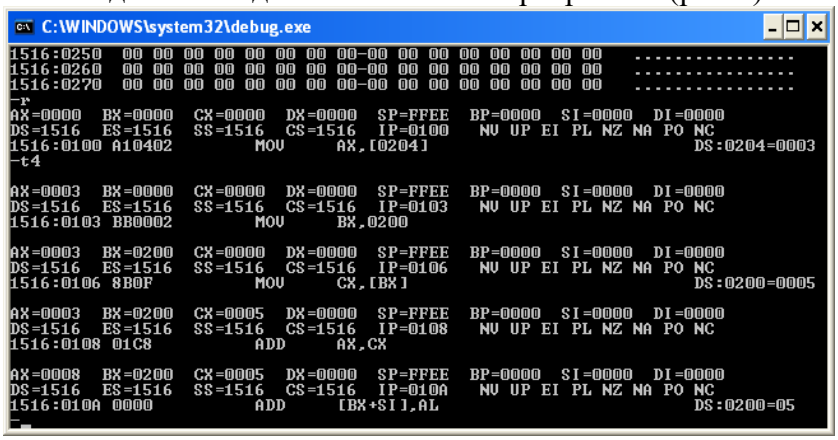


Рис.8.

В командной строке TRACE можно указать адрес выполняемой команды. В этом случае после «t» набирается знак равенства (=) и нужный адрес. Если указан короткий адрес, то адрес сегмента выбирается из регистра CS, например: -t=0100. В этом случае будет выполнена команда по адресу CS:0100. Адрес следующей команды находится в регистрах CS:IP. Он равен 1516:0103.

Одной командой TRACE можно одновременно трассировать несколько команд отлаживаемой программы. Для этого при введении «t» просто указывается их количество, Например: -t4.

После выполнения каждой команды на экране появляется картинка с содержимым регистров и флагов состояния. При заполнении экрана новые данные выводятся в нижней его части, сдвигая данные в верхней части за пределы экрана. Чтобы остановить движение данных вдоль экрана, следует нажать комбинацию клавиш ***Ctrl + Num Lock***. Чтобы возобновить движение, нажимается любая клавиша.

При нажатии ***Ctrl + C*** трассирование прекращается и на экране появляется стандартный запрос отладчика.

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом.
2. Описать построчно суть приведенных на рис.4 инструкций для ЦП (программы). Указать, какие методы адресации использованы.
2. Модифицировать решение на рис.4 так, чтобы доступ к обеим ячейкам 200 и 204 осуществлялся с помощью косвенной адресации через регистр BX. (продемонстрировать знание не менее 3 различных методов адресации).
3. Придумать и реализовать свой набор инструкций для ЦП, аналогичную приведенной выше, используя не менее 4 различных методов адресации, обеспечивающих доступ к данным, хранящимся в ОЗУ.

Контрольные вопросы

1. Чем отличаются процессоры CISC от RISC&
2. Какой регистр ЦП используется для хранения адреса команды?
3. Какой регистр ЦП используется для хранения указателя стека?
4. Укажите метод адресации, использованный в команде: MOV AX, [BX+100]
6. Укажите метод адресации, использованный в команде: MOV AX, 100
7. Какие из флагов относятся к статусным?

Лабораторная работа 4. Форматы представления данных в памяти эвм

Цель работы: Познакомиться с используемыми процессором методами определения данных. Научиться при использовании программы DEBUG просматривать требуемые сегменты памяти и представлять данные в требуемом формате.

Основные теоретические сведения

Аппаратное представление данных

С представлением данных в памяти ЭВМ тесно связано понятие типа данных. Понятие типа данных носит двойственный характер. С точки зрения размерности микропроцессор ***аппаратно поддерживает*** следующие основные типы данных.

1. ***Байт*** – восемь последовательно расположенных битов, пронумерованных от 7 до 0, при этом бит 0 является самым младшим значащим битом.
2. ***Слово*** – последовательность из двух байт, имеющих последовательные адреса. Размер слова – 16 бит; биты в слове нумеруются от 15 до 0. Байт, содержащий нулевой бит, называется младшим байтом, а байт, содержащий 15-й бит, – старшим байтом. Процессоры Intel имеют важную особенность – младший байт всегда хранится по меньшему адресу. Адресом слова считается адрес его младшего байта. Адрес старшего байта может быть использован для доступа к старшей половине слова.
3. ***Двойное слово*** – последовательность из четырех байт (32 бита), расположенных по последовательным адресам. Нумерация этих бит производится от 31 до 0. Слово, содержащее нулевой бит, называется младшим словом, а слово, содержащее 31-й бит, — старшим словом. Младшее слово хранится по меньшему адресу. Адресом двойного слова считается адрес его младшего слова. Адрес старшего слова может быть использован для доступа к старшей половине двойного слова.

Форматы данных

Кроме трактовки типов данных с точки зрения их разрядности, процессор на уровне команд поддерживает логическую интерпретацию этих типов. С точки зрения ***логической интерпретации*** выделяют следующие типы данных.

1. ***Целый тип без знака*** – двоичное значение без знака, размером 8, 16 или 32 бита. Числовой диапазон для этого типа следующий:
   – байт– от 0 до 255;
   – слово– от 0 до 65 535;
   – двойное слово – от 0 до $2^{32} – 1$.
2. ***Целый тип со знаком*** – двоичное значение со знаком, размером 8, 16 или 32 бита. Знак в этом двоичном числе содержится в 7, 15 или 31-м бите соответственно. Ноль в этих битах в операндах соответствует положительному числу, а единица – отрицательному. Отрицательные числа представляются в дополнительном коде. Он образуется следующим образом:
   – находится двоичное представление абсолютной величины числа;
   – найденный код инвертируется, т.е. в нем нули заменяются на единицы и наоборот;
   – к полученному коду арифметически прибавляется единица.

   Например, процесс получения дополнительного кода десятичного числа -75 таков: 01001011 → 10110100 → 10110101

   Числовые диапазоны для этого типа данных следующие:
   – 8-разрядное целое – от – 128 до 127;
   – 16-разрядное целое – от – 32 768 до 32 767;
   – 32-разрядное целое – от $– 2^{31}$ до $2^{31} – 1$.
3. ***Символ*** – байт, в который записывается код символа – целое от 0 до 255. В этом формате двоичный код интерпретируется обрабатывающей его командой как код символа. При работе

с персональными ЭВМ обычно используется система кодирования *ASCII (American Standard Code for Information Interchange — стандартный американский код для обмена информацией*). В этой системе стандартизированы (закреплены за определенными символами) коды, у которых значение старшего бита равно 0; все прочие коды остаются за символами национальных алфавитов и дополнительными специальными символами (см. Приложение 1).

В настоящее время используются *и двухбайтовые* представления символов (*Unicode*) (см. Приложение 2).

4. ***Строка*** – последовательность символов, которая размещается в соседних байтах памяти, так, что код первого символа строки записывается в первом байте, код второго символа - во втором байте и т.п. Адресом строки считается адрес ее первого байта.
5. ***Цепочка*** представляет собой некоторый непрерывный набор байтов, или слов максимальной длиной до 64 Кбайт.
6. ***Указатель на память*** (адрес) бывает двух типов: ближний тип – 16-разрядный логический адрес, представляющий собой относительное смещение в байтах от начала сегмента (короткий адрес); дальний тип – 32-разрядный логический адрес, состоящий из двух частей: 16-разрядной сегментной части и 16-разрядного смещения (полный адрес).
7. ***Неупакованный двоично-десятичный тип*** – байтовое представление десятичной цифры от 0 до 9. Неупакованные десятичные числа хранятся как байтовые значения без знака по одной цифре в каждом байте. Значение цифры определяется младшим полубайтом.
8. ***Упакованный двоично-десятичный тип*** представляет собой упакованное представление двух десятичных цифр от 0 до 9 в одном байте. Каждая цифра хранится в своем полубайте.

Команды ассемблера для определения аппаратного представления данных

Для определения элементов данных имеются следующие команды:

- DB (байт),
- DW (слово),
- DD (двойное слово),

Аргументом данных команд может быть константа, например: *DB 25*. Или числовой массив данных, разделенных запятыми: *DB 11, 12, 13, 14, 15, ….* Ассемблер определяет эти константы в виде последовательности смежных байт.

Определение однобайтовых данных

Ассемблер переводит символьные строки в объектный код в обычном формате ASCII. Символьная строка определяется только командой ***DB***, в которой указывается более двух символов в нормальной последовательности слева направо. Следовательно, директива DB представляет единственно возможный формат для определения символьных данных.

Например:

DB 25	– размещение в текущей ячейке символа с кодом 25-«%»
DB 25, 35, 45	– размещение в последовательно расположенных ячейках символов с кодами 25, 35, 45 – «%», «5», «E»
DB 'JAN','FEB','MAR'	– размещение в последовательно расположенных ячейках символьных констант «JAN», «FEB», «MAR»
DB '32654'	– размещение в последовательно расположенных ячейках символьной константы «32654»

Аргумент в команде DB может содержать строку символов любой длины. Объектный код показывает символы кода ASCII для каждого байта (рис.1)

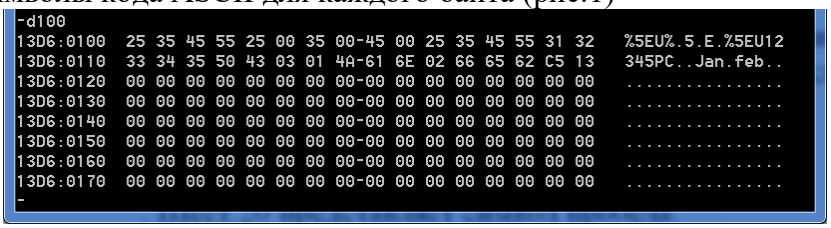


Рис.1.

```
-a100
13D6:0100 db 2
13D6:0101 db 3
13D6:0102 db 1F
13D6:0103
-d100
13D6:0100  02 03 1F 00 05 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0110  00 00 00 00 00 00 00 00-00 00 00 00 34 00 C5 13   ............4...
13D6:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Рис.2.

Числовой аргумент в команде DB может содержать одну или более однобайтовых констант (от 00 до 7F). Наибольшее положительное шестнадцатеричное число в одном байте - 7F, все «большие» числа от 80 до FF представляют отрицательные значения. В десятичном исчислении эти пределы выражаются числами +127 и -128.

При использовании данной команды следующее данное или команда будут размещены в следующей ячейке памяти (рис.2)

Определение данных длиной в слово (2 байта)

Команда ***DW*** определяет элементы, которые имеют длину в одно слово (2 байта). Аргумент команды DW может содержать одну или более двухбайтовых констант. Два байта представляются четырьмя шестнадцатеричными цифрами. Наибольшее положительное шестнадцатеричными число в двух байтах - 7FFF; все «большие» числа от 8000 до FFFF представляют отрицательные значения. В десятичном исчислении эти пределы выражаются числами +32767 и -32768.

Заметим, что объектный код для каждой константы этой команды имеет длину в одно слово (два байта) и занимает две последовательно расположенные ячейки. Ассемблер записывает данное в обратной последовательности (рис.3).

```
-a100
13D6:0100 dw 3
13D6:0102 dw 2
13D6:0104 dw 1f
13D6:0106 dw 1234
13D6:0108
-d100
13D6:0100  03 00 02 00 1F 00 34 12-00 00 00 00 00 00 00 00   ......4.........
13D6:0110  00 00 00 00 00 00 00 00-00 00 00 00 34 00 C5 13   ............4...
13D6:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13D6:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Рис.3.

Для определения символьных строк директива DW имеет ограниченное применение.

Двубайтные данные можно вводить и напрямую в ячейки памяти:

```
-e300
13E2:0300  00.30   00.31   00.32   00.33   00.34   00.35   00.36   00.37
13E2:0308  00.38   00.39
-d300
13E2:0300  30 31 32 33 34 35 36 37-38 39 00 00 00 00 00 00   0123456789......
13E2:0310  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13E2:0320  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13E2:0330  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13E2:0340  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13E2:0350  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13E2:0360  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13E2:0370  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Вещественные данные

Разберемся с тем, как представлены в памяти компьютера *вещественные числа*. Для упрощения будем считать, что вещественные числа – это НЕ целые числа, то есть числа с дробной частью, например: 1,5; 1,0; 1,52321456.

В общем виде эти числа можно записать следующим образом:

$$(\text{знак})(\text{мантисса})*10^{(\text{знак})(\text{порядок})}$$

Например: $-1{,}35*10^{5}$.Здесь мантисса = – 1, 35, порядок = - 5. Порядок тоже может иметь знак.

Нормализованная запись отличного от нуля действительного числа – это запись вида: $a = \pm m * P^q$,

где q – целое число (положительное, отрицательное или ноль)

m – правильная P-ичная дробь, у которой первая цифра после запятой не равна нулю, то есть: $1/P <= m < 1$

Например:

$3,14 = 0,314 * 10^1$

$2000 = 0,2 * 10^4$

$0,05 = 0,5 * 10^{-1}$

двоичные числа:

$1 = 0,1 * 2^1$

$100 = 0,1 * 2^3$

$11,11010010 = 0,1111010010 * 2^2$

$0,01 = 0,1 * 2^{-1}$

Умножение чисел на основание в какой-то степени, и если эта степень – целое положительное число, реализуется как сдвиг *влево* на количество разрядов, которое соответствует степени числа

Как нетрудно догадаться, деление – это сдвиг *вправо*.

Например:

$0,1 * 2^{-1} = 0,1 / 2 = 0,01$ (сдвинули число вправо на один разряд)

Число НОЛЬ не может быть записано в такой нормализованной форме, поэтому считается, что нормализованная запись нуля в десятичной системе будет такой: $0 = 0,0 * 10^0$

Т.о., если целая часть мантиссы числа состоит из одной цифры, не равной нулю, то число с плавающей точкой называется нормализованным

Преимущества использования нормализованных чисел

Для фиксированной разрядной сетки числа (то есть для фиксированного количества цифр в числе) нормализованные числа имеют наибольшую точность. Кроме того, нормализованное представление исключает неоднозначность - каждое число с плавающей точкой может быть представлено различными (ненормализованными) способами, например:

$123.5678*10^5 = 12.35678*10^6 = 1.235678*10^7 = 0.1235678*10^8$

Вещественные типы аппаратно могут иметь два представления:

- вещественные числа с фиксированной точкой
- вещественные числа с плавающей точкой.

Как правило, по умолчанию компиляторы преобразуют вещественные значения в экспоненциальный формат (формат с плавающей точкой), если синтаксис языка явно не указывает применение формата с фиксированной точкой.

Количество знаков после запятой может быть бесконечно большим, но память компьютера не бесконечна. Поэтому любое вещественное число будет помещено в память компьютера с какой-то погрешностью.

Нормализованная экспоненциальная запись числа– это запись вида:

$a = \pm m * P^q$

где q – целое число (положительное, отрицательное или ноль)

m – правильная P-ичная дробь, у которой целая часть состоит из одной цифры, при этом m – это мантисса числа, а q – порядок (или экспонента) числа.

Для тех, кто программировал на языках высокого уровня, знакомо следующее представление чисел с плавающей точкой: (знак)(мантисса)E(знак)(порядок)

Например, -5.35E-2 означает число $-5.35*10^{-2}$. Такое представление называется *научной нотацией*.

Описанные выше примеры в нормализованной экспоненциальной записи будут выглядеть так, как показано ниже.

Пример:

запись десятичных чисел:

$3,14 = 0,314 * 10^1 = 3,14 * 10^0$

$2000 = 0,2 * 10^4 = 2,0 * 10^3$

$0,05 = 0,5 * 10^{-1} = 5 * 10^{-2}$

запись двоичных чисел:

$1 = 0,1 * 2^1 = 1,0 * 2^0$

$100 = 0,1 * 2^3 = 1,0 * 2^2$

$11,11010010 = 0,1111010010 * 2^2 = 1,111010010 * 2^1$

$0,01 = 0,1 * 2^{-1} = 0,1 * 2^0$

В нормализованной форме первая цифра после запятой НЕ может быть нулём, а в нормализованной экспоненциальной форме это допускается.

Рассмотрим, как выполняется преобразование дробной части числа в двоичную форму. Дробная часть числа преобразуется в двоичное число путем разложения на сумму дробей вида 1/2 + 1/4 + 1/8 + …, то есть на сумму дробей, в знаменателе которых степени двойки.

Например:

Десятичная дробь	Разложение	Двоичное вещественное число
1/2	1/2	0,1
1/4	1/4	0,01
3/4	1/2 + 1/4	0,11
1/8	1/8	0,001
7/8	1/2 + 1/4 + 1/8	0,111

Поясним:

$1/2 = 1 * 2^{-1} = 0,1$

$3/4 = 1/2 + 1/4 = 1 * 2^{-1} + 1 * 2^{-2} = 0,1 + 0,01 = 0,11$

Преобразуем дробную часть числа 3,14:

1. 0,14 < 1/2, поэтому старший разряд равен 0
2. 0,14 < 1/4, поэтому следующий разряд также равен 0
3. 0,14 > (1/8 = 0,125), поэтому следующий разряд равен 1
4. 0,14 – 0,125 = 0,015
5. 0,015 < (1/16 = 0,0625), поэтому следующий разряд равен 0
6. 0,015 < (1/32 = 0,03125), поэтому следующий разряд равен 0
7. 0,015 < (1/64 = 0,015625), поэтому следующий разряд равен 0
8. 0,015 > (1/128 = 0,0078125), поэтому следующий разряд равен 1
9. 0,015 – 0,0078125 = 0,0071875
10. 0,0071875 > (1/256 = 0,00390625), поэтому следующий разряд равен 1

Таким образом: 3,14 = 11,0010011

Итак, для записи числа с фиксированной точкой используется один байт для записи целой части, а другой – для записи дробной части. Запись в памяти компьютера имеет вид

Знак	Целая часть	Дробная часть
0	0000011	00100011

Таким образом, число 3,14159265359 после помещения в слово данных будет равно приблизительно 3,14. Для снижения погрешностей используются специальные алгоритмы. Представление числа с фиксированной запятой имеет недостатки:

- высокая погрешность
- нерациональное использование памяти

Нерациональное использование памяти: в нашем случае для представления целой части достаточно всего двух битов, но используются семь, потому что точка фиксированная, то есть находится всегда в одном месте (между двумя байтами слова). Из-за этого же страдает точность.

Если использовать для целого числа только требуемое количество битов, то можно для записи дробной части уже использовать не 8 битов, а большее количество битов, тем самым повысить точность дробной части. Решение этой проблемы нашли – сделали точку плавающей и несколько изменили принцип записи числа в память. Чтобы повысить точность и максимально компактно расположить вещественное число в памяти компьютера, были придуманы числа с плавающей точкой.

Большинство вещественных чисел нельзя представить в виде конечного числа двоичных разрядов. Например, дробь 1/5 = 0,2 раскладывается достаточно сложно, при этом сумма дробей со знаменателями в степени двойки лишь приблизительно будет равна 1/5:

1/8 + 1/16 + 1/64 = 0,203125 = 0,2

А в двоичной форме это будет

0,203125 = 0,2 = 0,001+ 0,0001 + 0,000001 = 0,001101

Т.о. потребует 6 разрядов для записи, да ещё и с потерей точности, а, например, число

7/8 = 0,875 = 0,111 потребует всего 3 разряда.

Для представления вещественных чисел в памяти компьютера часть разрядов отводится для записи порядка числа, а остальные – для записи мантиссы.

Если это число со знаком, то старший бит отводится для знака. Но в этом формате знак может иметь не только число, но и порядок числа (то есть степень дроби может быть как положительной, так и отрицательной). Чтобы не хранить знак порядка, был придуман *смещённый порядок*.

Если для задания порядка выделено k разрядов, то к истинному значению порядка прибавляют смещение, таким образом, смещённый порядок определяется по формуле:

$$СП = ИП + 2^{k-1} – 1$$

где СП – смещённый порядок,

ИП – истинный порядок,

k – количество разрядов, выделенных для порядка

Например, истинный порядок, лежащий в диапазоне –127…+128 представляется смещённым порядком, значения которого меняются в диапазоне 0…255.

То есть при ИП = -127:

$СП = -127 + 2^{8-1} - 1 = -127 + 128 - 1 = 0$

При ИП = 128:

$СП = 128 = 128 + 2^{8-1} – 1 = 128 + 128 - 1 = 255$

Для представления числа в диапазоне 0…255 требуется 1 байт (8 разрядов), то есть k = 8.

В старшие биты стали записывать порядок числа. Размер порядка числа известен заранее (зависит от типа данных) и занимает намного меньше места, чем могла бы занять целая часть числа.

В младшие биты стали записывать мантиссу – нормализованную экспоненциальную форму числа без запятой. Таким образом, в пределах мантиссы точка может как бы «плавать», то есть её расположение зависит от порядка числа.

Процессор может работать с вещественными числами в трех форматах:

- одинарной точности;
- двойной точности;
- расширенной точности

Эти числа занимают в памяти, соответственно, 4, 8 или 10 байт (рис. 4).

В любом представлении старший бит определяет знак вещественного числа:

- ·0 - положительное число;
- ·1 - отрицательное число

Все равные по абсолютному значению положительные и отрицательные числа отличаются только этим битом. В остальном числа с разным знаком полностью симметричны. Для представления отрицательных чисел здесь не используется дополнительный код.

Рис. 4.

Обработкой вещественных чисел занимается аппаратная часть процессора - сопроцессор. Во многих процессорах даже есть специальные команды для операций над числами с плавающей точкой. В большинстве компьютеров используются именно числа с плавающей точкой (а не с фиксированной), т.к. это позволяет экономить память и получать большую точность

Алгоритм представления вещественного числа в памяти компьютера

1. Перевести число из P-ичной системы в двоичную
2. Представить двоичное число в нормализованной экспоненциальной форме
3. Рассчитать смещённый порядок числа
4. Разместить знак, порядок и мантиссу в соответствующие разряды

Рассмотрим пример: пусть имеется двоичное представление целой и дробной частей числа Пи:

3 = 11

0,14 = 0,00100011

То есть число 3,14 в двоичном виде равно:3,14 = 11,00100011

Теперь преобразуем это число в нормализованную экспоненциальную форму:

11,00100011 = 1,100100011b * 2^1

Рассчитаем смещённый порядок (предположим, что для хранения порядка у нас используется 5 бит). Тогда исходные данные:

ИП = 1 (у нас 2 в степени 1)

k = 5

СП = ИП + 2^{k-1} – 1 = 1 + 2^{5-1} – 1 = 1 + 16 – 1 = 16

Записываем знак числа, порядок и мантиссу в соответствующие разряды:

Знак	Порядок	Мантисса
0	10000	0010001100

Как видно, в мантиссе младшие два разряда – это нули. Эти разряды не используются, но при желании их можно использовать и тем самым повысить точность.

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом.
2. Разместить в ячейках ОЗУ текущего сегмента, используя соответствующие команды ассемблера и Приложение 1, коды следующих символов: а) C, б) p, в) #,г) ?.
3. Разместить в ячейках ОЗУ текущего сегмента, используя соответствующие команды ассемблера следующие данные: а) IBM PC, б) Computer Science.
4. Разместить в ячейках ОЗУ текущего сегмента, используя соответствующие команды ассемблера, следующие числа: а) 5000, б) 103, в) 21, г) 270.
5. Определить двоичные дополнения для следующих двоичных чисел:
   а)00010011, б) 00111100, в) 00111001. Результат представьте в шестнадцатеричном коде и разместите в регистрах центрального процессора.

6. Определить положительные значения для следующих отрицательных двоичных чисел: а) 11001000, б) 10111101, в) 10000000. Результат представить в десятичном коде и разместить в регистрах центрального процессора.
7. Представьте числа: а) 34567,123 б) 0,1234567 в) -13,14 г) -1/128 в вещественном формате с фиксированной точной. Разместить значение мантиссы в регистре AX, значение порядка в регистре BX.

Контрольные вопросы

1. Какие форматы данных могут быть представлены в ЭВМ на техническом уровне?
2. С какими форматами данных логически может работать программист?
3. Сколько бит информации используется для кодирования символов?
4. Какова длина в байтах для элементов данных, определенных директивами: а) DB, б) DW в) DD?
5. Как в ЭВМ представляются отрицательные числа: а) целые, б) вещественные.
6. Что такое упакованное вещественное число?
7. Сколько ячеек памяти занимает неупакованное вещественное число?

Лабораторная работа 5. Организация простейшего вычислительного процесса с использованием инструкций процессора

Цель работы: Познакомиться с основными арифметическими командами процессора, закрепить знание мнемонической записи методов адресации. Научиться составлять вычислительные программы на языке Ассемблер при использовании программы DEBUG.

Основные теоретические сведения

Форматы команд

В семействе процессоров Intel количество считываемых или записываемых байт определяется кодом машинной инструкции. Коды команд обращения к байту или слову отличаются в зависимости от разрядности. Например, байтовая команда MOV AL, 1 имеет код B0 01, двухбайтовая MOV AL, 1 — кодируется B8 01 00 (длина команды увеличилась из-за размера константы). Только из кода команды процессор узнает о необходимых действиях и способе адресации.

Рассмотрим обобщенные форматы команд, описывающих рассмотренные в предыдущей лабораторной работе методы адресации.

Формат команды, имеющий два операнда, один из которых располагается в регистре, а другой в регистре или памяти.

В общем случае такие команды могут занимать от двух до четырех байт памяти. Весь этот массив байт разбит на поля, каждое из которых несет определенную информацию.

Байт	0			1			2	3
Бит	7…2	1	0	76	543	210	7……0	7……0
Поле	*code*	*d*	*w*	*mod*	*reg*	*R/m*	*disp_low*	*disp_high*

Поля формата имеют следующее применение:

- поле *code* содержит код операции, описывающий операцию, выполняемую командой (под это поле отводится 6 старших бит нулевого байта кода команды);
- поле *d* (первый бит нулевого байта кода) определяет направление передачи результата: при *d* = 1 первый операнд определяется полем *reg*, т.е. всегда является регистром, а второй операнд определятся полем *r/m*, т.е. может быть ячейкой памяти или регистром; при *d* = 0 – наоборот, первый операнд определяется полем *r/m*, а второй – полем *reg*;
- поле *w* (нулевой бит нулевого байта кода) определяет размер операндов: при *w* = 0 – байты; при *w* = 1 – слова;
- поле *mod* (два старших бита первого байта кода) определяет режим адресации и, в частности, смысл поля *r/m* и существование полей *disp_low* и *disp_high*: при *mod* = 00 – поля

disp_low и *disp_high* отсутствуют, а поле *r/m* определяет операнд в памяти; при *mod* = 01 – поле *disp_high* отсутствует, а поле *r/m* определяет операнд в памяти; при *mod* = 10 – оба поля *disp_low* и *disp_high* присутствуют в коде команды, а поле *r/m* определяет операнд в памяти; при *mod* = 11 – поле *r/m* определяет операнд в регистре, а поля *disp_low* и *disp_high* естественно отсутствуют;

- поле *reg* (пятый, четвертый и третий биты первого байта кода) кодирует операнд располагаемый в регистре:

Таблица 1

Поле *reg*	*w* = 0	*w* = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- поле *r/m* (младшие три бита первого байта кода) используется совместно с полем *mod*:при значении поля *mod* = 11 поле *r/m* определяет операнд в регистре и кодирует регистр так же как и поле *reg* (см. выше);при значении поля *mod* отличном от 11 поле *r/m* определяет операнд в памяти и колирует режим адресации следующим образом:

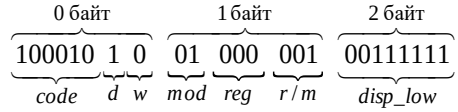
Таблица 2

Поле *r/m*	Режим адресации	Исполнительный адрес
000	Базово-индексная	BX + SI + disp
001	То же	BX + DI + disp
010	>>	BP + SI + disp
011	>>	BP + DI + disp
100	Косвенная или индексная	SI + disp
101	То же	DI + disp
110	Косвенная или по базе	BP + disp (при mod ≠ 00)
111	То же	BX + disp

Если поле *mod* = 00 и поле *r/m* = 110, то исполнительный адрес операнда определяется полями *disp_low* и *disp_high*, которые в данном случае обязательно присутствуют в коде команды. Этот случай определяет прямую адресацию;

- поля *disp_low* и *disp_high* являются соответственно младшим и старшим байтом компоненты исполнительного адреса (сдвига), присутствующей в коде команды. Наличие этих полей в коде команды отличают базовую и индексную адресацию от косвенной. Если в коде команды присутствует только поле *disp_low*, то при вычислении исполнительного адреса, оно расширяется до слова, старший байт которого равен нулю.

Например, команда имеет код *8A413Fh*. Выделим поля этого кода и удостоверимся, действительно это та самая команда. Переведем код команды в двоичную систему счисления:



Выделим поля кода команды:

code = 100010 – код операции пересылки (*MOV*);

- *d* = 1 – первый операнд располагается в регистре (см. поле *reg*), а второй в регистре или памяти (см. поле *r/m*);
- *w* = 0 – операнды имеют размер в 1 байт;

mod = 01 – поле *r/m* определяет операнд в памяти и поле *disp_high* в коде команды отсутствует;

reg = 000 – при *w* = 0 это регистр *AL*;

r/m = 001 – при *mod* = 01 это ячейка памяти с исполнительным адресом *BX* + *DI* + disp;

disp_low = 00111111 – младший байт компоненты сдвига в исполнительном адресе операнда;

disp (сдвиг) = 00000000 00111111 – получается расширением *disp_low* (в шестнадцатеричной системе это число *003Fh*).

Таким образом, приходим к выводу, что это код команды *mov AL, [BX + DI + 003Fh]*.

Формат команды, имеющий два операнда, один из которых располагается в регистре или памяти, а другой непосредственно в коде команды.

Этот формат применяется для команд, один из операндов которых имеет непосредственную адресацию. Длина кода этого формата от 3 до 6 байт.

Байт	0			1			2	3	4	5
Бит	7…2	1	0	76	543	210	7……0	7……0	7……0	7……0
Поле	*code*	*s*	*w*	*mod*	*reg*	*r/m*	*disp_low*	*disp_high*	*data_low*	*data_high*

Поля формата имеют следующее применение:

- поле *code* вместе с *reg*, а иногда и с *s*, определяет выполняемую операцию;
- поле *s* вместе с полем *w* определяет размер второго операнда и, тем самым, наличие поля *data_high*:при *s* = 0 размер второго операнда определяется полем *w*;при *s* = 1 размер второго операнда равен 1 байту, причем при *w* = 1 этот байт расширяется до слова путем размножения старшего разряда младшего байта в старший байт;
- поле*w* определяет размер первого операнда: при *w* = 0 – байт; при *w* = 1 – слово;
- поля mod и r/m используются как в рассмотренном ранее формате;
- поле reg является частью кода операции;
- поля *disp_low* и *disp_high* используются как в рассмотренном ранее формате (в частности могут отсутствовать);
- поля *data_low* и *data_high* – соответственно младший и старший байт непосредственного (второго) операнда (поле *data_high* может отсутствовать).

Краткие сведения о системе команд процессора

В общем случае система команд процессора включает в себя следующие четыре основные группы команд:

- команды пересылки данных;
- арифметические команды;
- логические команды;
- команды переходов.

Команды пересылки

Команды пересылки данных можно разделить на следующие группы:

- команды пересылки общего назначения;
- команды ввода/вывода через порт;
- команды пересылки адреса;
- команды пересылки содержимого регистра флагов.

В таблице 3 приведена мнемоника этих команд и описание выполняемых ими действий.

Таблица 3

Мнемоника команд передачи управления

Обозначение	Выполняемая функция

Команды пересылки общего назначения	
MOV	Пересылка значения второго операнда в первый операнд
PUSH	Помещение значения операнда в стек
POP	Извлечение значения из стека и помещение его в операнд
XCHG	Обмен значениями между операндами
XLAT	Табличное преобразование одного кода в другой (перекодировка)
Команды ввода/вывода через порт	
IN	Ввод данных (чтение порта)
OUT	Вывод данных (запись в порт)
Команды пересылки адреса	
LEA	Загрузка в первый операнд исполнительного адреса второго операнда
LDS	Загрузка первого операнда и регистра DS значением второго операнда
LES	Загрузка первого операнда и регистра ES значением второго операнда
Команды пересылки содержимого регистра флагов	
LAHF	Пересылка в регистр AH флагов из регистра признаков
SAHF	Запись содержимого регистра AH в регистр флагов
PUSHF	Сохранение содержимого регистра флагов в стеке
POPF	Извлечение слова из стека и помещение его в регистр флагов

Рассмотрим некоторые из них (остальные команды будут описании в следующих лабораторных работах):

1. Команда MOV

Мнемоника	Функция команды	Синтаксис	Примечание
MOV	Пересылает значение второго операнда в первый операнд	*MOV* X, Y	X:=Y, Первый операнд может иметь любой метод адресации, кроме непосредствен-ной. Второй операнд может иметь любой метод адресации. В качестве *операндов* могут быть использованы любые сегментные регистры, кроме *CS*
Использование регистра флагов		Регистр флагов не используется.	

Команда *MOV* имеет следующие форматы машинных кодов:

1) пересылка аккумулятора (*AX* или *AL*) в память или памяти в аккумулятор при прямой адресации памяти
2) пересылка из регистра в регистр или память или наоборот
3) пересылка непосредственных данных в память
4) пересылка непосредственных данных в регистр
5) пересылка сегментного регистра в регистр или память или наоборот

Примеры использования команды *MOV*:

MOV [123], AX	пересылка содержимого регистра AX в ячейку памяти 123*h* из текущем сегменте
MOV AX, [123]	пересылка содержимого ячейки памяти с адресом 123*h* в текущем сегменте в регистр AX
MOV word ptr [123], 12	пересылка числовой константы (число 12) в ячейку памяти с адресом 123*h*
MOV CX,123	пересылка числовой константы (число 123) в регистр CX
MOV DS, AX	пересылка содержимого регистра AX в сегментный регистр *DS*

2. Команда XCHG

Мнемоника	Функция команды	Синтаксис	Примечание
XCHG	Обмен операндов	XCHG x, y	X:=Y, Y:=X Возможен обмен между регистром и

			ячейкой памяти или обмен между регистрами. Операнды могут иметь любую адресацию, кроме непосредственной
Использование регистра флагов		Регистр флагов не используется.	

Команда XCHG имеет следующие форматы машинных кодов:

1) обмен содержимого регистра с аккумулятором (*AX*)
2) обмен содержимого регистра и регистра или ячейки памяти

Примеры использования команды *XCHG*:

XCHG AX, BX - обмен содержимого регистров AX и BX

XCHG DX, [BX + 2] - обмен содержимого ячейки памяти и регистра

XCHG [BX + 2], DX - обмен содержимого регистра и ячейки памяти

Основные арифметические команды

Арифметические команды можно подразделить на следующие подгруппы:

- команды сложения;
- команды вычитания;
- команды умножения;
- команды деления;
- команды изменения знака и размера операнда.

В таблице 4 приведена мнемоника этих команд и описание выполняемых ими действий.

Таблица 4

Мнемоника арифметических команд

Обозначение	Выполняемая функция
Команды сложения	
ADD	Сложение значений операндов
ADC	Сложение значений операндов и значения флага переноса
INC	Увеличение значения операнда на 1
AAA	Коррекция ASCII кода при сложении
DAA	Коррекция двоично-десятичного кода при сложении
Команды вычитания	
SUB	Вычитание значения второго операнда из значения первого
SBB	Вычитание значения первого операнда и значения флага переноса (заем) из значения первого операнда
DEC	Уменьшение значения операнда на 1
CMP	Сравнение значений операндов
AAS	Коррекция ASCII кода при вычитании
DAS	Коррекция двоично-десятичного кода при вычитании
Команды умножения	
MUL	Умножение без учета знака
IMUL	Умножение с учетом знака
AAM	Коррекция ASCII кода при умножении
Команды деления	
DIV	Деление без учета знака
IDIV	Деление с учетом знака
AAD	Коррекция ASCII кода при делении
Команды изменения знака и размера	
NEG	Изменение знака у операнда
CBW	Преобразование байта в слово
CWD	Преобразование слова в двойное слово

Рассмотрим некоторые из них:

1. Команда ADD

Мнемоника	Функция команды	Синтаксис	Примечание

ADD	Сложение двух кодов	ADD x, y	X:=X+Y X может быть регистром или ячейкой памяти. Y может быть регистром, ячейкой памяти или явно заданным числом. Нельзя, чтобы оба операнда были ячейками
Использование регистра флагов	Команда ADD изменяет значения флагов переполнения (OF), знака (SF), нуля (ZF), дополнительного переноса (AF), переноса (CF) и четности (PF). Флаг переполнения устанавливается, если при выполнении операции произошло переполнение (результат сложения не вмещается в размер операндов). Флаг знака устанавливается, если в результате выполнения операции получилось отрицательное число. Флаг нуля устанавливается, если результатом операции является ноль. Флаг дополнительного переноса устанавливается, если при сложении байт осуществлен перенос из 3-го бита. Флаг переноса устанавливается, если при сложении осуществлен перенос из старшего бита. Флаг четности устанавливается, если количество единичных битов в младшем байте результата операции четное.		

Примеры использования команды ADD:

ADD AX, [BX] - сложение содержимого регистра AX и ячейки памяти, адрес которой указан в регистре BX

ADD [BX], DX - сложение содержимого ячейки памяти, адрес которой указан в регистре BX и значения регистра *DX*

ADD AX, 123 - сложение содержимого регистра *AX* и константы

ADD word ptr [123],123 - сложение числовой константы с содержимым ячейки памяти

2. Команда SUB

Мнемоника	Функция команды	Синтаксис	Примечание
SUB	Вычитание значения регистра или ячейки памяти	SUB x, y	X:=X-Y X может быть регистром или ячейкой памяти. Y может быть регистром, ячейкой памяти или явно заданным числом. Нельзя, чтобы оба операнда были ячейками памяти.
Использование регистра флагов	Команда SUB изменяет значения флагов OF, SF, ZF, AF, CF и PF. Флаг дополнительного переноса устанавливается, если при вычитании байт осуществлен заем из 3-го бита. Флаг переноса устанавливается, если при вычитании осуществлен заем из старшего бита.		

Примеры использования команды SUB:

SUB DX, [BX + 123] - вычитание содержимого ячейки памяти из содержимого регистра DX

SUB [BX + 123], DX - вычитание содержимого регистра DX из содержимого ячейки памяти

SUB AX,123 - вычитание числовой константы из содержимого регистра AX

SUB word ptr [23],13 - вычитание числовой константы (числа 13) из содержимого ячейки памяти с адресом 23 в текущем сегменте

3. Команда INC

Мнемоника	Функция команды	Синтаксис	Примечание
INC	Декремент регистра или памяти	INC x	X:=X+1 Увеличивает значение операнда на 1. При этом операнд может быть регистром или ячейкой

			памяти.
Использование регистра флагов	Содержимое регистра флагов (кроме флага CF) при выполнении команды INC изменяется аналогично тому, как это выполняется командой ADD. Команда INC не изменяет состояние флага переноса.		

Примеры использования команды INC:

INC AX - увеличение на 1 значение регистра AX

INC byte ptr [BX+ 12] - увеличение на 1значение ячейки памяти.

4. Команда DEC

Мнемоника	Функция команды	Синтаксис	Примечание
DEC	Декремент регистра или памяти	DEC x	X:=X-1 Уменьшает значение операнда на 1
Использование регистра флагов	Содержимое регистра флагов (кроме флага CF) при выполнении команды DEC изменяется аналогично тому, как это делает команда SUB. Команда DEC не изменяет состояние флага переноса.		

Примеры использования команды DEC:

DEC AX - уменьшение значения регистра AX на 1

DEC byte ptr [BX + 2] - уменьшение значения ячейки память на 1

5. Команда MUL

Мнемоника	Функция команды	Синтаксис	Примечание
MUL	Вычисляет произведение двух кодов (разрядность результата вдвое больше разрядности сомножителей)	MUL x	AX:=AX * x Умножение чисел без знака. В команде указывается в качестве операнда второй множитель. Первый должен располагаться в регистре AX. Результат размещается в AX. Операнд-множитель может иметь любой метод адресации, кроме непосредственной При умножении байта множимое находится в регистре *AL*, произведение в регистре *AX*, При умножении слова – множимое в регистре *AX*, произведение в паре регистров *DX*:*AX*.
Использование регистра флагов	Команда MUL воздействует на флаги CF и OF (флаги AF, PF, SF, ZF не определены). Флаги CF и OF устанавливаются, если старшая часть произведения (AH или DX в зависимости от вида умножения) не равна 0. В противном случае эти флаги сбрасываются.		

Примеры использования команды MUL:

MUL BX - умножение данного из регистра BX на значение AX, размер данных – машинное слов

MUL BH - умножение данного из регистра BX на значение AX, размер данных – байт

MUL byte ptr [BX + 123] - умножение значения AX, на множитель, находящийся в памяти, длинна данного - байт

MUL word ptr [SI + 123] - умножение значения AX, на множитель, находящийся в памяти, длинна данного - слово

6. Команда IMUL

Мнемоника	Функция команды	Синтаксис	Примечание
IMUL	Команда умножает знаковое множимое на	IMUL x	AX:=AX*x Операнд-множитель может иметь любой метод адресации, кроме непосредственной.

	знаковый множитель		При умножении байта множимое находится в регистре *AL*, произведение в регистре *AX*, При умножении слова – множимое в регистре *AX*, произведение в паре регистров *DX*:*AX*.
Использование регистра флагов	Команда *IMUL* воздействует на флаги *CF* и *OF* (флаги *AF*, *PF*, *SF*, *ZF* не определены). Флаги *CF* и *OF* устанавливаются, если младшая часть произведения (*AL* или *AX* в зависимости от вида умножения) равна максимальному положительному числу, которое может в ней разместиться (7F*h* и 7FFF*h* соответственно), или старшая часть произведения (*AH* или *DX* в зависимости от вида умножения) не равна максимальному беззнаковому числу, которое может в ней разместиться (0FF*h* и 0FFFF*h* соответственно). В противном случае эти флаги сбрасываются		

Примеры использования команды IMUL:

IMUL BX - знаковое умножение слов, 1-ый множитель находится в регистре AX, 2-ой множитель - в регистре BX

IMUL byte ptr [BX + 123] - знаковое умножение байт, 1-ый множитель находится в регистре AX, 2-ой множитель находится в памяти по адресу

IMUL word ptr [CX] - знаковое умножение слов, 1-ый множитель находится в регистре AX, 2-ой множитель находится в памяти по адресу, указанному в CX

7. Команда DIV

Мнемоника	Функция команды	Синтаксис	Примечание
DIV	Вычисляет частное от деления одного кода на другой.	DIV x	Деление беззнаковых операндов при этом делимое должно находиться в регистре AX, а делитель указывается в команде. Значение операнда может быть размещено в регистре или в памяти. При делении на байт делимое находится в регистре AX, частное помещается в AL, а остаток от деления – в AH. При делении на слово делимое находится в паре регистров DX:AX, частное помещается в AX, а остаток от деления в DX.
Использование регистра флагов		После выполнения команды DIV флаги CF, OF, AF, PF, SF, ZF не определены	

Примечание:

1. Если частное слишком велико для того, чтобы поместиться в регистре (AL или AX), то при выполнении команды DIV происходит программное прерывание INT 0h. Во избежание этой ситуации рекомендуется перед выполнением деления осуществлять проверку делителя на неравенство нулю.
2. Другое правило, обеспечивающее благоприятные условия для выполнения деления, заключается в следующем: если делитель - байт, то его значение должно быть меньше, чем старший байт (AH) делимого; если делитель – слово, то его значение должно быть меньше, чем старшее слово (DX) делимого.

Примеры использования команды DIV:

DIV BX - деление на слово, делимое в *DX*:*AX*, делитель в регистре BX

DIV BH - деление AX на байт, делимое в *AX*, делитель в регистре BX

DIV byte ptr [BX] - деление на байт, делимое в *AX*, делитель в памяти

DIV word ptr [BX + 12] - деление на слово, делимое в *DX*:*AX*, делитель в памяти

8. Команда IDIV

Мнемоника	Функция команды	Синтаксис	Примечание

IDIV	Вычисляет частное от деления одного кода на другой.	IDIV x	Команда во всем аналогична команде DIV, с тем лишь исключением, что ее операнды рассматриваются как числа со знаком. При этом знак остатка всегда совпадает со знаком делимого Значение операнда может быть размещено в регистре или в памяти.
Использование регистра флагов	Команда воздействует на флаги CF и OF (флаги AF, PF, SF, ZF не определены).		

Примеры использования команды IDIV:

IDIV BX - знаковое деление данного из DX:AX на слово, находящееся в BX

IDIV BH - знаковое деление данного из AX на байт, делитель в регистре BX

IDIV byte ptr [BX] - знаковое деление данного из AX на байт, делитель в ячейке памяти, адрес которой указан в BX

IDIV word ptr [120] - знаковое деление данного из DX:AX на слово, делитель в памяти по указанному адресу

9. Команда NEG

Мнемоника	Функция команды	Синтаксис	Примечание
NEG	Изменение знака операнда	NEG x	X:= - X Операнд может быть регистром или ячейкой памяти. Эта команда изменяет знак своего операнда путем построения его дополнительного кода
Использование регистра флагов	Команда NEG изменят флаги AF, CF, OF, SF, PF и ZF так же, как команда SUB при вычитании операнда dest из 0.		

Примеры использования команды NEG:

NEG AX - изменение знака содержимого16-разрядного регистра AX

NEG AH - изменение знака содержимого 8-разрядного регистра

NEG byte ptr [BX + 12] - изменение знака содержимого ячейки памяти

10. Команда XADD

Мнемоника	Функция команды	Синтаксис	Примечание
XADD	Сложение с обменом	XADD x, y	X:= X+Y, Y:=X X может быть регистром или ячейкой памяти. Y может быть только регистром
Использование регистра флагов	Флаги устанавливаются так, как будь то выполнена инструкция ADD		

Дополнительные команды

11. Команда DW

Мнемоника	Функция команды	Синтаксис	Примечание
DW	Размещение числовых констант	DW x	X – числовая константа Размещение двухбайтового числа в текущей ячейке памяти
Использование регистра флагов	Регистр флагов не используется.		

12. Команда JMP

Мнемоника	Функция команды	Синтаксис	Примечание
JMP	Безусловный переход (прыжок) по адресу	JMP x	X – регистр, ячейка памяти или числовая константа Переход на считывание ячейки по указанному адресу в текущем блоке. Адрес возврата не сохраняется.
Использование регистра флагов	Возможно только в защищенном режиме процессора		

Команды программы DEBUG для работы с файлами

Команда загрузки файла в память

Загрузка файла в память осуществляется, если в командной строке DEBUG указать имя файла.

Другой способ – использование *команды LOAD.* При введении команды необходимо набрать «***l***» или «***L***».

При использовании команды LOAD необходимо специфицировать файл с помощью команды NAME (смотри ниже).

В командной строке LOAD можно указать начальный адрес, по которому загружается файл. Если указан короткий адрес, то адрес сегмента выбирается из регистра *CS*. При отсутствии начального адреса, загрузка производится по адресу *CS*:0100.

После загрузки отладчик запоминает количество занятой файлом памяти (в байтах) в регистрах *BX* (старшее слово) и *CX* (младшее слово).

Например, загрузка в память файла «mytest.pro» по адресу CS:0100 выглядит следующим образом:

```
-n mytest.pro
-L
-r
AX=0000 BX=0000 CX=00CFDX=0000 SP=FFEEBP=0000 SI=0000 DI=0000
DS=0958 ES=0958 SS=0958 CS=0958 IP=0100    NV UP DI PL NZ NA PO NC
0958:0100  2A2A        SUB          CH,[BP+SI]          SS:0000=CD
```

В регистрах BX и CX находится значение 207 (000000CF). Это значит, что файл занял 207 байт. Тот же результат можно получить при введении спецификации файла в командной строке команды старта отладчика («debug mytest.pro»).

Команда записи области памяти в файл

Команда WRITE (*w* или *W*) переписывает на диск данные, выбирая их из памяти. При этом спецификация создаваемого файла должна задаваться с помощью команды NAME (см. ниже).

Перед введением команды WRITE в регистры BX и CX записывается размер занимаемой файлом памяти в байтах (шестнадцатеричное число, занимающее 4 байта). Поэтому перед записью необходимо проверить содержимое этих регистров (с помощью REGISTER).

В командной строке WRITE можно указать начальный адрес памяти, по которому производится чтение данных с последующей записью их на диск. Если указан короткий адрес, то адрес сегмента выбирается из регистра *CS*.

Если начальный адрес не указан, то запись производится, начиная с адреса CS:0100.

Команда задания имени файла программы

Команда NAME (*n* или *N*) присваивает имя обрабатываемому файлу. Затем этот файл загружается в память командой LOAD или записывается на диск командой WRITE.

Чтобы идентифицировать файл, наберите «*n*» и, через пробел – имя файла. Воспользуемся NAME, чтобы присвоить нашей программе имя «mytest.pro»:

```
-n mytest.pro
```

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом.

2. Организовать вычислительный процесс (написать программу), вычисляющий значение выражения в соответствии с номером варианта, используя ячейки памяти и регистры процессора.

Исходные данные в шестнадцатеричной системе счисления(X, Y, Z), должны быть расположены в ячейках памяти.

№ варианта	X	Y	Z	Выражение
1	3	D	8	K = X*(10 – Y) + 30/Z
2	5	15	A	K = 75*X – MOD(Z,2)*DIV(Y,3)
3	C	3	7	K = 5*Y – MOD(X,7) + 9^2 +Z
4	9	4	D	K = Z + (X*Z – Y)(X + Z/Y)
5	4	E	6	K = X^2 – DIV(Y,2^2) – MOD(Z,2^2)
6	2	9	B	K = 3*X^2 + 9*Y – 16/Z
7	B	5	3	K = (DIV(Y*Z) – X)*(2*Y – Z)
8	6	12	F	K = (Z + Y)*(Y/3 – X) +A
9	14	F	9	K = 11*Z – DIV(X,5) + 11*Y
10	A	6	2	K = (X + Y) * MOD(12, Z) – (X – Z)

Например:

Написать программу, вычисляющую значение выражения

K = 2X – Y + Z, где X = 5, Y = 6, Z = 4C.

Протокол вычислительного процесса:

```
-a
150E:0100 jmp 108
150E:0102 dw 5
150E:0104 dw 6
150E:0106 dw 4c
150E:0108 mov ax, 2
150E:010B mov bx, [102]
150E:010F mul bx
150E:0111 mov bx, [104]
150E:0115 sub ax, bx
150E:0117 mov bx, [106]
150E:011B add ax, bx
-t4
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=150E  ES=150E  SS=150E  CS=150E  IP=0108   NV UP EI PL NZ NA PO NC
150E:0108 B80200        MOV     AX,0002
AX=0002  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=150E  ES=150E  SS=150E  CS=150E  IP=010B   NV UP EI PL NZ NA PO NC
150E:010B 8B1E0201      MOV     BX,[0102]                   DS:0102=0005
AX=0002  BX=0005  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=150E  ES=150E  SS=150E  CS=150E  IP=010F   NV UP EI PL NZ NA PO NC
150E:010F F7E3          MUL     BX
AX=000A  BX=0005  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=150E  ES=150E  SS=150E  CS=150E  IP=0111   NV UP EI PL NZ NA PO NC
150E:0111 8B1E0401      MOV     BX,[0104]                   DS:0104=0006
-t4
AX=000A  BX=0006  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=150E  ES=150E  SS=150E  CS=150E  IP=0115   NV UP EI PL NZ NA PO NC
150E:0115 29D8          SUB     AX,BX
AX=0004  BX=0006  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=150E  ES=150E  SS=150E  CS=150E  IP=0117   NV UP EI PL NZ NA PO NC
150E:0117 8B1E0601      MOV     BX,[0106]                   DS:0106=004C
AX=0004  BX=004C  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=150E  ES=150E  SS=150E  CS=150E  IP=011B   NV UP EI PL NZ NA PO NC
150E:011B 01D8          ADD     AX,BX
```

```
AX=0050  BX=004C  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=150E  ES=150E  SS=150E  CS=150E  IP=011D   NV UP EI PL NZ AC PE NC
150E:011D 00FD          ADD     CH,BH
```

Ответ: значение выражения = 50

Контрольные вопросы

1. Для чего используется регистр DS?
2. Какие регистры ЦП можно использовать для размещения данных?
3. Что можно использовать в качестве аргументов в командах сложения и вычитания?
4. Каков синтаксис команд умножения и деления?
5. Какими способами можно изменить значение аргумента на 1?

Лабораторная работа 6. Адресация структурированных данных

Цель работы: Познакомиться с методами адресации структурированных данных на примере одномерных и двумерных массивов. Закрепить знание мнемонической записи команд. Научиться составлять вычислительные инструкции по обработке структурированных данных на языке Ассемблер.

Основные теоретические сведения

Адресация одномерных массивов данных

Рассмотрим теперь более сложную ситуацию, когда однотипные данные (в нашем случае 16-битовые целые) последовательно хранятся в памяти в виде массива. В этом случае имеет место заполнение памяти, изображенное на рис.1.

A_0+10h	08	00	M_8
A_0+Eh	07	00	M_7
A_0+Ch	06	00	M_6
A_0+Ah	05	00	M_5
A_0+8	04	00	M_4
A_0+6	03	00	M_3
A_0+4	02	00	M_2
A_0+2	01	00	M_1
A_0	00	00	M_0

Рис.1.

Как видно из рис. 1, адрес начального байта любого числа легко вычислить по формуле: $A_i = A_0 + D_i = A_0 + 2i$.

В данном случае выберем A0 = 200, а индекс i = 0, 1, …, 8.

Следует обратить внимание на тот факт, что нумерацию в массиве удобно начинать с нуля: именно в этом случае расчетная формула выглядит наиболее просто. Простота объясняется тем, что фактически для адресации необходимого элемента приходится вычислять объем памяти, занятый всеми **предшествующими ему** элементами. Т.о., например, пятому элементу удобнее всего ставить в соответствие номер 4, тогда первому, соответственно 0 (т.е. ему «ничего не предшествует»).

Пусть в памяти, начиная с адреса 200h, располагается массив из 9 двухбайтовых целых чисел, который изображен на рис.1. Значения чисел можно задать произвольно, но для удобства проверки правильности работы программ в элементы массива будут занесены их индексы от 0 до 8. Поставим задачу: найти в массиве и извлечь в регистр AX пятый элемент (на рис.1 он обведен пунктирами имеет номер 4).

Для доступа к элементам массива в процессорах фирмы Intel предусмотрен большой ассортимент методов адресации. Доступ к массивам основывается на двух методах – ***базовом*** и ***индексном***, остальные, являются их модификациями.

Для хранения индексов в рассматриваемом семействе процессоров предусмотрены специальные индексные регистры SI и DI. Хранящееся в любом из них значение индекса может

автоматически складываться с начальным адресом массива, который должен быть указан явным образом в виде константы.

Например, запись 200[SI] означает, что при обращении к памяти будет выбран адрес, равный сумме начального адреса массива 200 и индексного смещения, находящегося в данный момент в регистре SI.

Величина смещения не есть значение индекса: в данном примере она вдвое превышает последнее, поскольку каждое число занимает 2 байта. Вместо умножения на 2 как правило, используют команды сдвига:

Команда SHL/SAL

Мнемоника	Функция команды	Синтаксис	Примечание
SHL/SAL	Сдвигает влево содержимое своего первого операнда на число разрядов, указанных во втором операнде	SHL X, Y или SAL X, Y	X – первый операнд, который может иметь любой режим адресации, кроме непосредственной Y - второй операнд, который может быть регистром *CL* или числовым выражением со значением 1.
Использование регистра флагов		Используются флаги*OF* и *CF*, данная команда воздействует так же на флаги *PF*, *SF* и *ZF*, флаг *AF* – не определен.	

Замечание:

1. Хотя во мнемонике ассемблера данной команде соответствует два обозначения (*SHL* и *SAL*), тем не менее это одна и та же команда.
2. Если вторым операндом является 1, то сдвиг производится на 1 разряд. Если второй операнд – регистр *CL*, то сдвиг осуществляется на количество разрядов равное содержимому данного регистра.

При сдвиге на 1 разряд старший (знаковый) бит первого операнда помещается во флаг переноса (*CF*), а остальные разряды сдвигаются влево. В младший бит помещается 0. Если затем *CF* совпадает со старшим битом результата, то во флаг переполнения (*OF*) помещается 0, иначе он устанавливается в 1. Сдвиг на несколько разрядов можно рассматривать как последовательность сдвигов на один разряд, но флаг *OF* в этом случае не определен. Следует отметить, что сдвиг влево на n разрядов эквивалентен умножению операнда на число 2n.

Примеры использования команды *SHL/SAL*:

SHL DX,1 - сдвиг содержимого регистра на 1 разряд влево
MOV CL,3 - сдвиг содержимого регистра на 3 разряда влево
SHL DX, CL
SHL byte ptr [12],1 - сдвиг содержимого ячейки памяти на1 разряд влево

Команда SHR

Мнемоника	Функция команды	Синтаксис	Примечание
SHR	Сдвигает вправо содержимое своего первого операнда на число разрядов, указанных во втором операнде	*SHR*X,Y	X – первый операнд, который может иметь любой режим адресации, кроме непосредственной Y - второй операнд, который может быть регистром *CL* или числовым выражением со значением 1.
Использование регистра флагов		Используются флаги*OF* и *CF*, данная команда воздействует так же на флаги *PF*, *SF* и *ZF*, флаг *AF* – не определен.	

Замечание:

1. Если вторым операндом является 1, то сдвиг производится на 1 разряд.
2. Если второй операнд – регистр *CL*, то сдвиг осуществляется на количество разрядов равное содержимому данного регистра.

При сдвиге на 1 разряд младший бит первого операнда помещается во флаг переноса (*CF*), а остальные разряды сдвигаются вправо. В старший бит помещается 0. Если затем знаковый бит результата совпадает со следующим по порядку битом, то во флаг переполнения (*OF*) помещается 0, иначе он устанавливается в 1.

Сдвиг на несколько разрядов можно рассматривать как последовательность сдвигов на один разряд, но флаг *OF* в этом случае всегда помещается ноль.

Помимо изменения флагов *OF* и *CF*, данная команда воздействует на флаги *PF*, *SF* и *ZF*, флаг *AF* – не определен.

Следует отметить, что данный сдвиг вправо на *n* разрядов эквивалентен беззнаковому делению операнда на число 2^n.

Примеры использования команды *SHR*:

SHR DX,1 - сдвиг содержимого регистра на 1 разряд вправо
MOV CL,3 - сдвиг содержимого регистра на 3 разряда вправо
SHR DX, CL
SHR byte ptr [123],1 - сдвиг содержимого ячейки памяти ; на 1 разряд вправо

Пример 1

```
-a
1423:0100 mov si,4
1423:0103 shl si,1
1423:0105 mov ax,200[si]
1423:0109
-u
1423:0100 BE0400       MOV     SI,0004
1423:0103 D1E6         SHL     SI,1
1423:0105 8B840002     MOV     AX,[SI+0200]
1423:0109 0000         ADD     [BX+SI],AL
-e200 0 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0
-d200
1423:0200  00 0001 0002 0003 00-04 0005 0006 0007 00   ................
1423:0210  08 0000 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
...
-t3
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0004  DI=0000
DS=1423  ES=1423  SS=1423  CS=1423  IP=0103   NV UP EI PL NZ NA PO NC
1423:0103 D1E6         SHL     SI,1
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0008  DI=0000
DS=1423  ES=1423  SS=1423  CS=1423  IP=0105   NV UP EI PL NZ AC PO NC
1423:0105 8B840002     MOV     AX,[SI+0200]                DS:0208=0004
AX=0004  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0008  DI=0000
DS=1423  ES=1423  SS=1423  CS=1423  IP=0109   NV UP EI PL NZ AC PO NC
1423:0109 0000         ADD     [BX+SI],AL                  DS:0008=1D
```

При разборе протокола следует обратить внимание на следующие детали (соответствующие места в тексте подчеркнуты).

- Индексный операнд 200[SI] Debug расшифровывает «по-своему»: [SI+200].
- Сдвиг влево преобразует значение SI из 4 в 8, т.е действительно удваивает.
- В случае, когда выводимая на экран команда работает с памятью, Debug вычисляет адрес данных с учетом текущего значения регистров (в нашем случае SI) и выводит соответствующее содержимое справа от ассемблерного представления инструкции.

Используемый в данном примере метод адресации называется ***индексной адресацией***. В ходе некоторых методов адресации процессор самостоятельно способен умножать индекс на размер данных. В частности, ***масштабированный индексный*** метод адресации хорошо подходит к данной задаче. Главная проблема применения метода заключается в том, что он работает только в 32-битном режиме и, следовательно, с расширенными регистрами, а их Debug не поддерживает. Трудность состоит не в том, что при работе с Debug *нельзя* использовать 32-разрядные инструкции, а в том, что их не удается ввести напрямую. В связи с этим можно предложить воспользоваться «обходным» путем: предварительно узнав коды необходимых инструкций ввести их в память как набор байтов.

Учитывая автоматизацию умножения на 2, указанная выше небольшая экспериментальная программа сокращается до двух команд, которым соответствуют следующие шестнадцатеричные коды:

команда	код
mov esi,4	66 BE 04 00 00 00
mov ax,200[esi*2]	67 8B 84 36 00 02 00 00

Для получения кодов инструкций можно воспользовался программой Flat Assembler (автор Tomasz Grysztar). Чмсла 66 и 67 это *префиксы*, переводящие данную команду из 16-разрядного в 32-разрядный режим выполнения, причем первый влияет на размер данных (число 4 заносится в 32-разрядный регистр ESI), а второй – на длину адреса, что, собственно, и позволяет применять изучаемый масштабированный индексный метод адресации.

Дальнейшие действия представлены в протоколе 3. Он является продолжением предыдущего примера по поиску пятого элемента массива. Необходимо очистить регистр AX и «сбросить» на начало программы счетчик команд IP (см. в протоколе команды ***r ax*** и ***r ip***).

Протокол 3 (продолжение протокола 2)

```
-a100
1423:0100 db 66 be 4 0 0 0
1423:0106 db 67 8b 84 36 0 2 0 0
1423:010E
-u100
1423:0100 66          DB      66
1423:0101 BE0400      MOV     SI,0004
1423:0104 0000        ADD     [BX+SI],AL
1423:0106 67          DB      67
1423:0107 8B843600    MOV     AX,[SI+0036]
1423:010B 0200        ADD     AL,[BX+SI]
1423:010D 0000        ADD     [BX+SI],AL
...
-r ax
AX 0004
:0
-r ip
IP 0109
:100
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0008 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=0100   NV UP EI PL NZ AC PO NC
1423:0100 66          DB      66
-t2
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0004 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=0106   NV UP EI PL NZ AC PO NC
1423:0106 67          DB      67
AX=0004 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0004 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=010E   NV UP EI PL NZ AC PO NC
1423:010E 0000        ADD     [BX+SI],AL                  DS:0004=00
```

Хотя Debug и не способен вводить и правильно выводить 32-разрядные команды, те прекрасно выполняются под отладчиком даже в режиме трассировки (анализ содержимого счетчика команд IP показывает, что сборка байтов в команды происходит абсолютно корректно).

Разберем ***базовый*** метод адресации.

Его суть состоит в том, что адрес данных получается в результате суммирования содержимого базового регистра (BX или BP) с некоторой константой, например, [BX+30]. При этом имеются некоторые тонкости: как известно, адрес элемента массива является суммой двух слагаемых – A_0 и D_i. Если A_0 помещается в виде константы в команду, а – D_i в регистр, то адресация называется *индексной*, а при обратном размещении слагаемых – *базовой*. У

процессоров с универсальными регистрами, допускающими любые методы адресации, отличие чисто смысловое; в тексте программы на ассемблере оно еще заметно по способу записи, но на уровне машинных кодов разница между индексной и базовой адресацией весьма условна.

У процессоров Intel регистры неравноценны. В частности, регистры BP и BX называются базовыми, а SI и DI – индексными, отсюда адресация по BP и BX - базовая, а по SI и DI – индексная. Согласно принятым в языке ассемблер правилам, индексная адресация записывается в виде 200[SI], а базовая – [BP+200]. Тем не менее, отладчик кроме стандартного 200[SI] нормально воспринимает набор [SI+200], то же самое и относится к регистру BX.

Адресация двумерных массивов данных

Рассмотрим размещения в памяти двумерных массивов и способов доступа к их отдельным элементам на примере массива 3x3 (рис. 2). Отдельные строки на рисунке для наглядности сдвинуты.

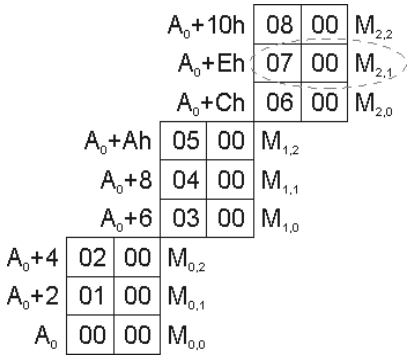


Рис. 2.

Все языки программирования хранят двумерные массивы как последовательность строк. Для доступа к фиксированному элементу массива с текущими значениями индексов i и j необходимо, добавить еще слагаемое, учитывающее полный размер всех предшествующих строк (строки опять же удобнее нумеровать с 0) В итоге расчетная формула для данного массива с i = 0, 1, 2 и j = 0, 1, 2 приобретет вид: Ai,j = A0 + Di,j = A0 + 6i + 2j. (число 6 получено как произведение числа элементов в строке на размер каждого из них).

Для индексирования теперь требуется два регистра BX и SI (один базовый, другой индексный). Применив базово-индексный метод адресации со смещением, получим программу, реализация которой дается в протоколе 4.

Протокол 4

```
-a
1423:0100 mov bl,2
1423:0102 mov si,1
1423:0105 mov al,6
1423:0107 mul bl
1423:0109 mov bx,ax
1423:010B shl si,1
1423:010D mov ax,200[bx][si]
1423:0111
-u
1423:0100 B302          MOV     BL,02
1423:0102 BE0100        MOV     SI,0001
1423:0105 B006          MOV     AL,06
1423:0107 F6E3          MUL     BL
1423:0109 89C3          MOV     BX,AX
1423:010B D1E6          SHL     SI,1
1423:010D 8B800002      MOV     AX,[BX+SI+0200]
1423:0111 0000          ADD     [BX+SI],AL
...
-e200 0 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0
-d200
1423:0200  00 0001 0002 0003 00-04 0005 0006 0007 00   .....
```

```
1423:0210  08 0000 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
...
-t5
AX=0000 BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=0102  NV UP EI PL NZ NA PO NC
1423:0102 BE0100        MOV     SI,0001
AX=0000 BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0001 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=0105  NV UP EI PL NZ NA PO NC
1423:0105 B006          MOV     AL,06
AX=0006 BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0001 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=0107  NV UP EI PL NZ NA PO NC
1423:0107 F6E3          MUL     BL
AX=000C BX=0002 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0001 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=0109  NV UP EI PL NZ NA PO NC
1423:0109 89C3          MOV     BX,AX
AX=000C BX=000C CX=0000 DX=0000 SP=FFEE BP=0000 SI=0001 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=010B  NV UP EI PL NZ NA PO NC
1423:010B D1E6          SHL     SI,1
-t2
AX=000C BX=000C CX=0000 DX=0000 SP=FFEE BP=0000 SI=0002 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=010D  NV UP EI PL NZ AC PO NC
1423:010D 8B800002      MOV     AX,[BX+SI+0200]                 DS:020E=0007
AX=0007 BX=000C CX=0000 DX=0000 SP=FFEE BP=0000 SI=0002 DI=0000
DS=1423 ES=1423 SS=1423 CS=1423 IP=0111  NV UP EI PL NZ AC PO NC
1423:0111 0000          ADD     [BX+SI],AL                      DS:000E=8A
```

Примечание. Важно подчеркнуть, что описанный в примере способ вычисления адреса используется только в том случае, когда значения i и j заранее неизвестны и в ходе исполнения берутся из переменных. Если же в тексте программы написан элемент с конкретными индексами, например, $M_{1,1}$, то современный компилятор немедленно определяет адрес этого элемента в памяти и генерирует всего одну команду с прямой адресацией, в нашем примере *MOV AX,[208]*.

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом.
2. Используя данные таблицы1 из лабораторной работы 3 (стр. 37) приведите разнообразные допустимые варианты записи индексного и базового методов адресации. (например, [SI]30[BX]) проверьте на практике, что Debug правильно интерпретирует указанные адреса.
3. В программе из протокола 3 команду mov esi,4 замените на mov esi,4000000 с кодом 66 BE 00 00 00 04. Что при этом происходит. Объясните почему.
4. Изменив значения констант в первых двух командах (протокол 3), убедитесь в правильности получения элемента массива с заданными вами индексами.
5. Введите и выполните программу, которая реализует 32-разрядный метод адресации с масштабированием (16-разрядные команды, выделенные жирным шрифтом), введите и исполните ее в Debug.

команда	код
mov ebx,2	**66 BB 02 00 00 00**
mov esi,1	**66 BE 01 00 00 00**
mov al,6	B0 06
mul bl	F6 E3
mov bx,ax	89 C3
mov ax,[ebx+esi*2+200h]	**67 8B 84 73 00 02 00 00**

6. Выполните задание в соответствии с номером варианта:

Вариант	*Задание*
1	Найти сумму одномерного массива из 10 элементов

2	Найти сумму двумерного массива из 3х3 элементов
3	Дан одномерный массив из 10 элементов. Найти сумму элементов с четными индексами
4	Дан одномерный массив из 10 элементов. Найти сумму элементов с не четными индексами
5	Дан двумерный массив из 3х3 элементов. Найти сумму элементов не четных строк
6	Дан двумерный массив из 4х3 элементов. Найти сумму элементов четных строк
7	Дан двумерный массив из 3х3 элементов. Найти сумму элементов не четных столбцов
8	Дан двумерный массив из 3х4 элементов. Найти сумму элементов четных столбцов
9	Дан одномерный массив из 20 элементов. Найти сумму элементов с индексами, кратными 3
10	Дан одномерный массив из 20 элементов. Найти сумму элементов с индексами, кратными 4

Контрольные вопросы

1. Для чего используется регистр SI?
2. Какие регистры ЦП можно использовать для размещения адреса данных?
3. В чем разница между базовой и индексной адресацией?
4. Как располагаются структурированные данные (массивы) в ОЗУ?
5. Какие команды используются для операции сдвига данных?

Лабораторная работа 7. Организация вычислительного процесса с циклами

Цель работы: Познакомиться с основными командами процессора, осуществляющими организацию циклов. Закрепить знание мнемонической записи команд. Научиться составлять вычислительные программы на языке Ассемблер с использованием циклов.

Основные теоретические сведения

Организация не линейного вычислительного процесса осуществляется с помощью команд передачи управления. Команды передачи управления можно подразделить на следующие группы:

– команды безусловной передачи управления;
– команды условного перехода;
– команды управления циклами.

Таблица 1

Мнемоника команд передачи управления

Обозначение	Выполняемая функция
Команды безусловной передачи управления	
CALL	Вызов подпрограммы
RET	Возврат из подпрограммы
INT	Вызов программного прерывания
INTO	Вызов прерывания переполнения
IRET	Возврат из прерывания
JMP	Безусловный переход
Команды условного перехода	
Jxx	Группа команд условного перехода
JCXZ	Переход по счетчику

Команды управления циклами	
LOOP	Цикл с заданным количеством повторений
LOOPE/LOOPZ	Цикл до тех пор, пока не нуль
LOOPNE/LOOPNZ	Цикл до тех пор, пока нуль

Рассмотрим команды, предназначенные для организации циклов. Команда JMP реализует бесконечный цикл. Но в подавляющим числе случаев программа должна выполнять определенное число циклов. Команды управления циклами (LOOP, LOOPE, LOOPZ, LOOPNE, LOOPNZ) использует начальное значение регистра **CX** (счетчик цикла). В каждом цикле команда LOOP автоматически уменьшает содержимое регистра CX на 1 (декрементируют).

1. Команда LOOP

Мнемоника	Функция команды	Синтаксис	Примечание
LOOP	Организует цикл	LOOP *адрес*	Команда LOOP автоматически уменьшает содержимое регистра CX на 1. Пока значение в CX не станет равно нулю, управление передается по адресу, указанному в операнде. Если в CX будет 0, управление переходит на следующую ячейку после LOOP Данная команда способна только на короткие переходы в пределах от - 127….128 байт от текущего адреса инструкции
Использование регистра флагов		Регистр флагов не используется.	

Примеры использования команды LOOP:

LOOP*0109* -переход на ячейку 0109 в текущем сегменте

Алгоритм выполнения инструкции:

1. Выполнить декремент содержимого регистра ECX/CX;
2. Анализ регистра ECX/CX:
   - если ECX/CX=0, передать управление следующей за loop команде;
   - если ECX/CX=1, передать управление команде, метка которой указана в качестве операнда LOOP.

Состояние флагов после выполнения команды не меняется.

Команду LOOP применяют для организации цикла со счетчиком. Количество повторений цикла задается значением в регистре ECX/CX перед входом в последовательность команд, составляющих тело цикла.

2. Команда LOOPE

Мнемоника	Функция команды	Синтаксис	Примечание
LOOPE	Организует цикл, если CX не равно нулю и значение флага ZF равно 1	LOOPE *адрес*	Похожа на предыдущую команду, но выход из цикла произойдет, либо когда CX = 0, либо флаг ZF=0. Полезна когда нужно прервать цикл. В этом случае нужно только установить значение флага в 0. Даже если значение CX не будет рано 0, цикл все равно прервется
Использование регистра флагов		Регистр флагов не используется.	

3. Команда LOOPNE

Мнемоника	Функция команды	Синтаксис	Примечание
LOOPNE	Организует цикл, если CX не равно нулю и значение флага ZF равно 0	LOOPNE *адрес*	Похожа на предыдущую команду, но выход из цикла произойдет, либо когда CX = 0, либо флаг ZF=1. Полезна когда нужно прервать цикл. В этом случае нужно только установить

			значение флага в 1. Даже если значение CX не будет рано 0, цикл все равно прервется
Использование регистра флагов		Регистр флагов не используется.	

Замечание:

- для предотвращения выполнения цикла при изначально нулевом ECX/CX следует использовать команду JECXZ/JCXZ (см. лабораторную работу 6). Если этого не сделать, то цикл повторится 4 294 967 295/65 536 раз;
- смещение метки, являющейся операндом LOOP, не должно выходить из диапазона -128...+127 байт. Это смещение, является относительным от значения счетчика адреса следующей за LOOP команды.
- команды позволяют легко реализовать цикл с условием или заданным количеством повторений, которое задается в CX.

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом.
2. Написать программу вычисления 12 чисел Фибоначчи: 1, 1, 2, 3, 5, 8, 13,... (каждое число в последовательности представляет собой сумму двух предыдущих чисел). Следует иметь ввиду, что система счисления ЦП – шестнадцатеричная. Выполнить с помощью отладчика DEBUG трассировку программы.
3. Согласно номеру варианта выполнить индивидуальное задание:

Вариант	*Задание*
1	Используя операцию вычитания, написать программу нахождения частного и остатка от деления 16 числа на 3
2	На отрезке [1; 10] (целые числа) найти такое целочисленное значение первого члена арифметической прогрессии, при котором один из ее членов равен X. Разность d ($d \neq 1$) задать самостоятельно. Сколько членов последовательности предшествуют члену со значением X?
3	Даны два натуральных числа. Определить, сколько натуральных чисел расположено между ними, и поместите эти числа в последовательно расположенные ячейки памяти
4	Вычислить 2^n, где n –натуральное число
5	Вычислить сумму 10 чисел, последовательно размещенных в ячейках памяти
6	Найдите сумму всех целых чисел, принадлежащих отрезку [a, b].
7	К числу A прибавить n раз на число B
8	Вычислить $S = 1^2 + 2^2+3^2+\ldots10^2$
9	Вычислить 3^n, где n –натуральное число
10	Число A умножить n раз на число B

Контрольные вопросы

1. От чего зависит, сколько раз повториться цикл?
2. Если сравнивать циклический вычислительный процесс на Ассемблере и алгоритмические конструкции циклов, то укажите какой алгоритмической структуре соответствует вычислительный процесс?
3. Какие действия выполняют команды цикла в ассемблере?
4. Какую команду необходимо предусмотреть перед меткой перехода для цикла?

Лабораторная работа 8. Организация вычислительного процесса с переходами без возврата

Цель работы: Познакомиться с основными командами процессора, осуществляющими операции перехода (безусловно и по условию). Закрепить знание мнемонической записи команд. Научиться составлять вычислительные инструкции на языке Ассемблер с использованием переходов.

Основные теоретические сведения

Организация не линейного вычислительного процесса осуществляется с помощью команд передачи управления. Команды передачи управления можно подразделить на следующие группы:

– команды безусловной передачи управления;
– команды условного перехода;
– команды управления циклами.

Таблица 1

Мнемоника команд передачи управления

Обозначение	Выполняемая функция
Команды безусловной передачи управления	
CALL	Вызов подпрограммы
RET	Возврат из подпрограммы
INT	Вызов программного прерывания
INTO	Вызов прерывания переполнения
IRET	Возврат из прерывания
JMP	Безусловный переход
Команды условного перехода	
Jxx	Группа команд условного перехода
JCXZ	Переход по счетчику
Команды управления циклами	
LOOP	Цикл с заданным количеством повторений
LOOPE/LOOPZ	Цикл до тех пор, пока не нуль
LOOPNE/LOOPNZ	Цикл до тех пор, пока нуль

Команды перехода в системе команд процессора занимают одно из центральных мест. Рассмотрим основные команды передачи управления:

Безусловный переход

1. *Команда JMP*

Мнемоника	Функция команды	Синтаксис	Примечание
JMP	Выполняет безусловный переход, прямо или косвенно определенный операндом.	JMP n	n – регистр, адрес ячейки памяти или числовая константа Перенаправляет процессор к указанной инструкции, не запоминая адрес возврата.
Использование регистра флагов		Регистр флагов не используется.	

Существуют четыре типа команды JMP:

– прямой внутрисегментный переход;
– косвенный внутрисегментный переход;
– прямой межсегментный переход;
– косвенный межсегментный переход.

При прямом переходе адрес передачи управления находится непосредственно в коде команды, а при косвенном переходе – в регистре или в памяти. При внутрисегментном переходе используется только одна компонента адреса: смещение команды, к которой осуществляется переход, относительно начала текущего кодового сегмента, а при межсегментном переходе – обе компоненты адреса, т.е. сегмент, в котором располагается команда, к которой осуществляется переход, и ее смещение относительно начала этого сегмента.

Примеры использования команды JMP:

JMP 0BD4 - прямой внутрисегментный переходк команде, располагаемой по адресуCS:0BD4h

JMP 0234:0BD4 - прямой межсегментный переход ккоманде, располагаемой по адресу0234:0BD4*h*

JMP AX - косвенный внутрисегментный переход ккоманде, располагаемой по адресу *CS*:*AX*

JMP word ptr [BX + 123] - косвенный внутрисегментный переход

JMP dword ptr [BX + 123] - косвенный межсегментный переход

Безусловный переход на ячейку с указанным адресом используется очень часто. По сути, выполняет то же самое действие, что и оператор goto в языках программирования

Например:

```
0C9D:0100 MOV   AX,0001
0C9D:0103 MOV   BX,0001
0C9D:0106 MOV   CX,0001
0C9D:0109 ADD   AX,0001
0C9D:010C ADD   BX,AX
0C9D:010E  JMP   0109
```

Условный переход

Условный переход **–** условный переход выполняется в зависимости от сути проверяемого условия. Ассемблер содержит целую группу команд условного перехода. Переход выполняется, если соответствующее условие выполнено. Каждое условие является каким-либо состоянием флагов. Обычно перед командой условного перехода выполняется команда **CMP**, хотя это совсем не обязательно.

2. Команда CMP

Мнемоника	Функция команды	Синтаксис	Примечание
CMP	Команда предназначена для сравнения своих операндов	CMP X, Y	X - может иметь любой метод адресации, кроме непосредствен-ной Y – может иметь любой метод адресации, если первый операнд имеет регистровую адресацию; может иметь регистровую или непосредственную адресацию, если первый операнд адресует данные в памяти Действия, выполняемые командой CMP, аналогичны командеSUB, но результат вычитания ни куда не помещается, а изменяются только флаги.
Использование регистра флагов		Меняет значение флагов OF, SF, ZF, AF, PF, CF (см. ниже).	

Команда CMP имеет следующие форматы машинных кодов:

1) сравнение операнда в памяти или в регистре и операнда в регистре
2) сравнение непосредственного операнда с содержимым аккумулятора
3) сравнение непосредственного операнда с операндом в регистре или в памяти

Примеры использования команды *CMP*:

CMP DX, [BX + 12] - сравнение содержимого регистра и ячейки памяти

CMP [BX], DX - сравнение содержимого ячейки памяти и регистра

CMP AX, 123 - сравнение регистра с числовой константой

CMP word ptr [123], 12 - сравнение содержимого ячейки памяти с числовой константой

В результате вычитания **ОПЕРАНД1 – ОПЕРАНД2** значение операндов не меняется, команда меняет флаги (FLAGS) в 16-битном регистре, где каждый бит имеет определенное значение:

11	10	?	?	7	6	4	2	0
OF	DF	IF	TF	SF	ZF	AF	PF	CF

CF (Carry Flag) –флаг переноса. Содержит значение «переносов» (0 или 1) из старшего разряда при арифметических операциях, некоторых операциях сдвига и циклического сдвига

PF (Parity Flag) –флаг четности. Проверяет младшие восемь бит результатов операций над данными. Нечетное число бит приводит к установке этого флага в 0, а четное - в 1. Не следует путать флаг четности с битом контроля на четность.

AF (Auxiliary Carry Flag) - дополнительный флаг переноса. Устанавливается в 1, если арифметическая операция приводит к переносу четвертого справа бита (бит номер 3) в регистровой однобайтовой команде. Данный флаг имеет отношение к арифметическим операциям над символами кода ASCII.

ZF (Zero Flag) - флаг нуля. Устанавливается в качестве результата арифметических команд и команд сравнения. Как это ни странно, ненулевой результат приводит к установке нулевого значения этого флага, а нулевой - к установке единичного значения. Кажущееся несоответствие является, однако, логически правильным, так как 0 обозначает «нет» (т.е. результат не равен нулю), а единица обозначает «да» (т.е. результат равен нулю). Команды условного перехода JE и JZ проверяют этот флаг.

SF (SIgn Flag) - знаковый флаг. Устанавливается в соответствии со знаком результата (старшего бита) после арифметических операций: положительный результат устанавливает 0, а отрицательный - 1. Команды условного перехода JG и JL проверяют этот флаг.

TF (Trap Flag) - флаг пошагового выполнения. Этот флаг вам уже приходилось устанавливать, когда использовалась команда T в отладчике DEBUG. Если этот флаг установлен в единичное состояние, то процессор переходит в режим пошагового выполнения команд, т.е. в каждый момент выполняется одна команда под пользовательским управлением.

IF (Interrupt Flag) –флаг прерывания. При нулевом состоянии этого флага прерывания запрещены, при единичном - разрешены.

DF (DIrection Flag) - флаг направления. Используется в строковых операциях для определения направления передачи данных. При нулевом состоянии команда увеличивает содержимое регистров SI и DI, вызывая передачу данных слева направо, при нулевом - уменьшает содержимое этих регистров, вызывая передачу данных справа налево

OF (Overflow Flag) - флаг переполнения. Фиксирует арифметическое переполнение, т.е. перенос вниз старшего (знакового) бита при знаковых арифметических операциях.

Например: Команда CMP сравнивает два операнда и воздействует на флаги AF, CF, OF, PF, SF, ZF. Однако, нет необходимости проверять все эти флаги по отдельности. В следующем примере проверяется: содержит ли регистр BX нулевое значение:

```
0100  CMP BX,00       - сравнение BX с нулем
0103   JZ 0150        - переход на ячейку с адресом 0150 если нуль
……..                 - (действия при не нуле)
0150  MOV AX, BA ..   - точка перехода при BX=0
```

Если BX содержит нулевое значение, команда CMP устанавливает флаг нуля ZF в единичное состояние, и возможно изменяет (или нет) другие флаги. Команда JZ (переход, если нуль) проверяет только флаг ZF. При единичном значении ZF, обозначающее нулевой признак, команда передает управление на адрес, указанный в ее операнде.

При необходимости значение флагов можно устанавливать принудительно с помощью специальных команд ассемблера (таблица1).

Таблица1

Команды ассемблера управления флагами

stc – установить CF в 1	**clc** – установить CF в 0
cld – установить DF=0	**std** – установить DF=1
cli – запретить прерывания (IF=0)	**sti** – разрешить прерывания (IF=1)
pushf – поместить FLAGS в стек	**popf** – загрузить FLAGS из стека

Команды условного перехода

В общем случае формат команд условного перехода можно представить следующим образом:

Jcc адрес

Jcc – одна из команд условного перехода (таблица 2).

Таблица 2

Команды условного перехода

Код команды	Реальное условие	Условие для CMP	Код команды	Реальное условие	Условие для CMP
JA	CF=0 и ZF=0	если выше	**JG**	ZF=0 и SF=OF	если больше
JAE **JNC**	CF=0	если выше или равно если нет переноса	**JGE**	SF=OF	если больше или равно
JB **JC**	CF=1	если ниже если перенос	**JL**	SF<>OF	если меньше
JBE	CF=1 ZF=1	если ниже или равно	**JLE**	ZF=1 | SF<>OF	если меньше или равно
JE **JZ**	ZF=1	если равно если ноль	**JNE** **JNZ**	ZF=0	если не равно если не ноль
JO	OF=1	если есть переполнение	**JNO**	OF=0	если нет переполнения
JS	SF=1	если есть знак	**JNS**	SF=0	если нет знака
JP	PF=1	если есть четность	**JNP**	PF=0	если нет четности

Например: Команда CMP производит сравнение двух значений (например, значение в регистре AX и значение в регистре BX), результатом её работы является сброс и установка определённых бит в регистре флагов.

Команда JE проверяет установлен ли в регистре флагов флаг равенства и если он установлен выполняет переход на метку, если же флаг не установлен процессором будут выполняться команды расположенные ниже JE.

CMP AX, BX - переход по условию: если AX = BX, то переходим

JE адрес по указанному адресу, если нет, выполняется следующая команда...

Интересной особенностью ассемблера является то, что для выполнения условного перехода необязательно выполнять сравнение, ведь условный переход производится на основании текущего состояния регистра флагов, а значит с его помощью можно обработать результаты выполнения любой команды, устанавливающей флаги.

Например можно обработать получение в сумме 0:

ADDAX,BX - складывает значения в регистрах AX и BX

JNZ адрес - если результат не равен 0 - выполняется переход на указанный адрес, при этом весь код между JNZ и указанным адресом выполняться не будет

адрес *команда* - это указанный адрес

Аналогия ассемблера и ЯВУ

Ассемблер	***Аналогия в PASCAL***
JMP<адрес>	goto<метка>
JZ< адрес >	if<выражение>=0 thengoto<метка>
JNZ< адрес >	if<выражение> 0 thengoto<метка>
JC< адрес >	if<выражение1> меньше <выражения2>goto<метка>
JNC< адрес >	if<выр.1> больше или равно <выр.2>goto<метка>
JS< адрес >	ifsgn(<выр.1>)=-1 thengoto<метка>
JNS< адрес >	ifsgn(<выр.1>)=1 thengoto<метка>

Пример:

Задача: Найти минимум из 3 чисел и разместить его в регистре AX.

Например: 6, 3, 1

```
0100   MOV  AX, 6
0103   MOV  BX, 3
0106   MOV  CX, 1
0109   CMP  AX, BX
010C   JNC   0113
010E   MOV DX, AX
0111   JMP  0116
0113   MOV DX, BX
0116   CMP  DX, CX
```

0119 JC 011E
011B MOV DX, CX
011E MOV AX, DX

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом.
2. Письменно в тетради (отчете) ответить на данное задание: предположим, что регистры AX, BX, CX и DX - содержат данные. Определите команды CMP (где необходимо) и команды безусловного перехода для следующих проверок:
   а) значение в DX больше, чем в CX?
   б) значение в BX меньше, чем в AX?
   в) CX содержит нуль?
   г) значение в BX равно или меньше, чем в AX?
   д) значение в DX равно или больше, чем в CX?
3. Согласно номеру варианта выполнить индивидуальное задание:

Вариант 1

1. Большее из трех натуральных чисел умножить на 10, среднее по величине — на 5, меньшее — на 2.
2. Вычислить значение функции

$$Y=\begin{cases}0, \text{если } X\le 0\\ 5X-MOD(X,7)+22, \text{если } 0<X\le 20\\ 1, \text{если} > 20\end{cases}$$

Вариант 2

1. Проверить, удовлетворяют ли числа M и N соотношениям |M|<10 и |N| > 5. В случае, если оба соотношения справедливы, то очистить регистр AX, в противном случае записать туда 1.
2. Вычислить значение функции

$$Y=\begin{cases}5X^2, \text{если } X\le 1\\ 3X^2+9X-13, \text{если } 1<X\le 66\\ 0, \text{если } X>66\end{cases}$$

Вариант 3

1. Результаты вычислений по формулам y = 8*A — 4*B и z = |A + 4*B| записать в ячейки памяти. Большее из них поместите в регистр AX.
2. Вычислить значение функции

$$Y=\begin{cases}11X-DIV(X,10)+11, \text{если } X\le 2\\ 11X+DIV(X,10)-11, \text{если } 2<X\le 50\\ 11X+MOD(X,10), \text{если } X>50\end{cases}$$

Вариант 4

1. Найдите min {max (A, B), max (C, D)}.
2. Вычислить значение функции

$$Y=\begin{cases}X^2-DIV(X,22)-MOD(X,22), \text{если } X\le 0\\ X^2+DIV(X,22)+MOD(X,22), \text{если } 0<X\le 22\\ X^2, \text{если } X>22\end{cases}$$

Вариант 5

1. Проверьте, попадает ли точка C{x, y) в квадрат {a < x < b; c < y <d). Если попадает, то ее абсциссу занесите в регистр AX, иначе — в ячейку памяти.
2. Вычислить значение функции

$$Y=\begin{cases}0, \text{если } X \le 0\\ 3X - MOD(X,5)+2, \text{если } 0 < X \le 10\\ 1, \text{если} > 10\end{cases}$$

Вариант 6

1. Даны три числа. Занесите их в ячейки памяти в порядке возрастания.
2. Вычислить значение функции

$$Y=\begin{cases}2X^2, \text{если } X \le 1\\ X^2+4X-1, \text{если } 1 < X \le 15\\ 0, \text{если } X > 15\end{cases}$$

Вариант 7

1. Из трех чисел найти наибольшее и вычесть из него все остальные.
2. Вычислить значение функции

$$Y=\begin{cases}4X - DIV(X,10)+1, \text{если } X \le 2\\ 3X + DIV(X,10)-8, \text{если } 2 < X \le 10\\ 11X + MOD(X,10), \text{если } X > 10\end{cases}$$

Вариант 8

1. Записать числа a и b в память. Вычислить c =|a – 7b| и сравните c и d. Если c > d, то очистить регистр AX, иначе — BX.
2. Вычислить значение функции

$$Y=\begin{cases}X^2 - DIV(X,2^2) - MOD(X,2^2), \text{если } X \le 0\\ X^2 + DIV(X,2^2) + MOD(X,2^2), \text{если } 0 < X \le 5\\ X^2, \text{если } X > 5\end{cases}$$

Вариант 9

1. Заданы длины трех отрезков. Определить, могут ли эти отрезки служить сторонами треугольника. Если могут, то в регистр AX занести 1, иначе — 2.
2. Вычислить значение функции

$$Y=\begin{cases}0, \text{если } X \le 0\\ 15X - MOD(X,5)+25, \text{если } 0 < X \le 10\\ 1, \text{если} > 10\end{cases}$$

Вариант 10

1. В ячейках Rl, R2, R3 находятся числа. Записать адрес ячейки памяти, в которой находится наибольшее значение в регистр AX.
2. Вычислить значение функции

$$Y=\begin{cases}X^2, \text{если } X \le 1\\ AX^2+9X-C, \text{если } 1 < X \le B\\ 0, \text{если } X > B\end{cases}$$

Контрольные вопросы

1. Каков синтаксис команд условного перехода?
2. Какие флаги анализируют команды безусловного перехода?
3. Как формируется машинный код команды безусловного перехода ассемблера?
4. Что такое близкий и дальний переходы в ассемблере?
5. Как различить в командах прямой и косвенный переходы?

Лабораторная работа 9. Организация и использование стековой памяти для организации вычислительного процесса

Цель работы: Познакомиться с особенностями организации и работы стековой памяти, стеком как неявным методом адресации. Изучить основные команды процессора обращения к стеку. Закрепить знание мнемонической записи команд. Научиться составлять вычислительные инструкции на языке Ассемблер при использовании стековой памяти.

Основные теоретические сведения

Принципы организации стека

Стек – это определенный динамический способ хранения данных, при котором в каждый момент времени доступ возможен только к одному из элементов, а именно к тому, который был занесен в стек последним. Главный принцип стека: «*первым пришел – последним ушел*» (в англоязычной литературе применяется более строгий термин LIFO: LastIn – First Out).

При описании работы стека принято говорить, что он имеет фиксированное основание (нижнюю границу, дно) и подвижную вершину, через которую, собственно, и происходит запись и чтение данных.

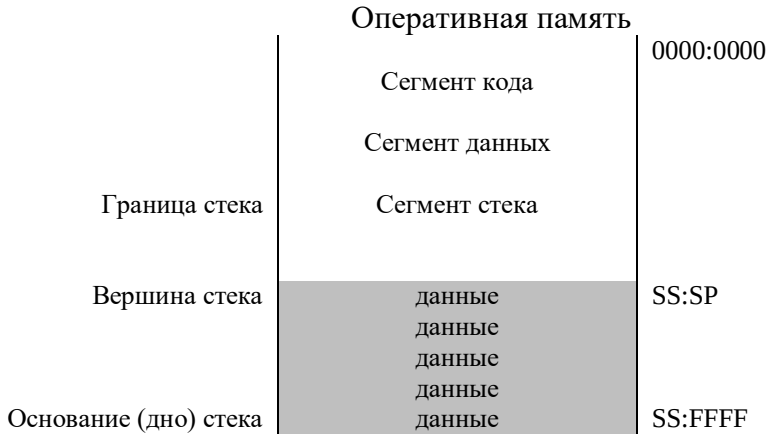


Рис.1.

В начальный момент времени обе границы совпадают. По мере записи данных область, занятая информацией (на рисунке закрашена серым цветом), расширяется. Вершина стека при этом смещается вверх. При извлечении данных из стека происходит противоположный процесс.

Когда стек свободен от данных (вершина и основание совпадают), попытка считывания данных является грубой логической ошибкой. В другом предельном случае, когда данных чрезмерно много (вершина совпадает с верхней границей области памяти, отведенной под стек), некорректной, напротив, становится запись.

Для использования стековой памяти необходимо хранить адрес сегмента стека (основания), и текущего положения вершины стека - ***указателя***.

Для работы со стеком предназначены три регистра:

1) **SS** – сегментный регистр стека. Обеспечивает доступ к сегмент у стека;
2) **SP/ESP** – регистр указателя стека. Компонента смещения логического адреса вершины стека;
3) **BP/EBP** – регистр указателя базы кадра стека.

Учитывая, что стек может расти как в сторону увеличения, так и в сторону уменьшения адресов, и то, что существуют равноправные договоренности о последовательности операций при записи/чтении (например, сначала менять адрес, а потом читать данные или наоборот), возможны несколько вариантов организации стека в ОЗУ. Тем не менее, системные программисты практически всегда строят стек единообразно:

- при заполнении стека значение указателя уменьшается (алгоритм –(R) - значение указателя уменьшается, а затем происходит запись данных),
- при извлечении данных – растет (алгоритм (R)+ - сперва считываются данные, а затем наращивается указатель стека)

Например**:** Пусть SP = 2006 и выполняется инструкция для работы с двухбайтовыми данными. Тогда при записи в стек значение SP будет уменьшено до 2004, и по этому адресу будет произведена запись требуемых данных и т.д.

Если считывать данные, то они будут извлечены из адреса 2000, а указатель стека увеличится до 2002и т.д. (фактически это означает возврат указателя в исходное состояние) (рис.2.).

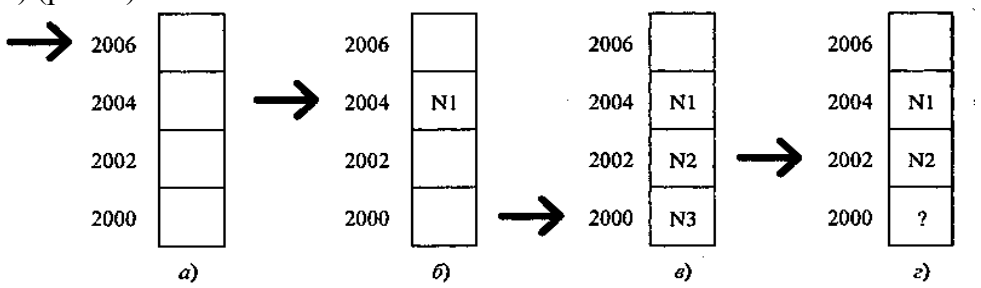


Рис.2.

На возможности компьютерной реализации стека могут также влиять принципы организации системы команд процессора. Так, в машинах семейства PDP в качестве регистра для стековой адресации может использоваться *любой* из регистров процессора, в то время как для процессоров IBM-совместимых компьютеров существует единственный регистр для работы со стеком – SP .

Над стеком можно определить следующие операции:

- добавление нового элемента в вершину стека (общепринят термин– «заталкивать» в стек);
- извлечение элемента из вершины стека («выталкивать» из стека);
- унарные операции по изменению верхнего элемента стека;
- бинарные операции с двумя извлеченными верхними элементами; результат возвращается обратно в вершину стека.

Команды для работы со стеком

Команды для работы со стеком относятся к группам команд:

- пересылки данных,
- передачи управления (см. следующую лабораторную работу)

Таблица 1

Команды для работы со стеком

Обозначение	Выполняемая функция
Команды пересылки общего назначения	
PUSH	Помещение значения операнда в стек
POP	Извлечение значения из стека и помещение его в операнд
Команды пересылки содержимого регистра флагов	
PUSHF	Сохранение содержимого регистра флагов в стеке
POPF	Извлечение слова из стека и помещение его в регистр флагов

1. Команда PUSH

Мнемоника	Функция команды	Синтаксис	Примечание
PUSH	Помещения в стек	PUSH *операнд*	*Операнд* может иметь любую адресацию, кроме непосредствен-ной, или быть сегментным регистром. Команда уменьшает содержи-мое регистра *SP* на 2, затем пересылает данное, адресуемое своим операндом в элемент стека, находящийся по адресу SS:SP.
Использование регистра флагов		Содержимое регистра флагов не изменяется.	

Замечание: команда PUSH всегда помещает в стек 16-разрядные данные, т.е. при регистровой адресации ее операнда должны быть использованы только 16-разрядные регистры.

Команда *PUSH* имеет следующие форматы машинных кодов:

1) сохранение в стеке содержимого регистра
2) сохранение в стеке содержимого ячейки памяти
3) сохранение в стеке содержимого сегментного регистра

Примеры использования команды PUSH:

PUSH AX - помещение в стек содержимого регистра AX
PUSH [BX + 1] - помещение в стек содержимого ячейки памяти
PUSH CS - помещение в стек содержимого сегментного регистра CS

2. Команда POP

Мнемоника	Функция команды	Синтаксис	Примечание
POP	Пересылка данных, находящееся по адресу SS:SP, в указанный операнд	POP*операнд*	*операнд* может иметь любой метод адресации, кроме непосредственной, или быть сегментным регистром Эта команда пересылает слово, находящееся по адресу *SS:SP*, в *операнд*, а затем увеличивает содержимое регистра *SP* на 2.
Использование регистра флагов		Содержимое регистра флагов не изменяется.	

Замечание: команда POP всегда извлекает из стека 16-разрядные данные, т.е. при регистровой адресации ее операнда должны быть использованы только 16-разрядные регистры.

Команда *POP* имеет следующие форматы машинных кодов:

1) извлечение из стека в регистр
2) извлечение из стека в ячейку памяти
3) извлечение из стека в сегментный регистр

Примеры использования команды *POP*:

POP AX - извлечение данных из стека в регистр AX
POP [BX + 12]; - извлечение данных из стека в ячейку памяти
POP DS - извлечение данных из стека в сегментный регистр DS

3. Команда PUSHF

Мнемоника	Функция команды	Синтаксис	Примечание
PUSHF	Пересылка данных, регистра флагов в стек	PUSHF	Не имеет операндов и подобно команде *PUSH* сохраняет в стеке содержимое регистра флагов.
Использование регистра флагов		Содержимое регистра флагов не изменяется.	

4. Команда POPF

Мнемоника	Функция команды	Синтаксис	Примечание
POPF	Извлекает из стека данные, помещает их в регистр флагов	POPF	Не имеет операндов и подобно команде *PUSH* сохраняет в стеке содержимое регистра флагов.
Использование регистра флагов		Содержимое всех флагов изменяется.	

Начальная установка указателя

Начальное значение указателя алгоритмом никак не определяется. Его установка является самостоятельной проблемой, а программист должен задумываться, каково это значение и можно ли рассчитывать на то, что оно уже было ранее корректно установлено.

При запуске отладчика Debug адрес, который операционная система заносит в указатель стека SP, соответствует как правило «верхней» области памяти (с максимально возможными адресами). Для большинства практических задач этим вполне можно удовлетвориться.

В противном случае можно воспользоваться командой **MOV SP,<const>,** которая занесет в указатель стека требуемую константу.

Стек – как метод неявной адресации

Как следует из описанных выше принципов, конкретный адрес при обращении к памяти берется из указателя стека. Для записи в стек содержимого регистра R_i достаточно указать команду PUSH R_i, никак не ссылаясь на адрес информации в ОЗУ. Подобные способы обращения к данным, когда адрес данных не указывается, а подразумевается, принято называть ***неявными***.

Стек является одним из образцов неявной адресации, важным преимуществом стека по сравнению с адресной организацией памяти является то, что его элементами можно манипулировать, не адресуясь к ним явно.

Преимуществом неявной адресации:

1. короткий формат команд,
2. ее практически полная независимость от конкретных адресов ОЗУ,
3. позволяет реализовать многие структуры алгоритмов или данных, которые без него осуществить необычайно сложно (подпрограммы, рекурсивные функции, и т.д.).

Использование стека в вычислительной технике

В вычислительной технике стек получил широчайшее распространение, причем как для организации хранения данных, так и для реализации вычислительного процесса.

1. ***Использование стека для временного хранения данных***. Очень часто некоторые промежуточные результаты необходимо где-то временно сохранить, прежде чем они будут использованы в дальнейшей обработке.

Например, некоторому фрагменту программы в машинных командах требуется использовать несколько регистров процессора для хранения промежуточных результатов, которые по окончании его работы уже не будут представлять интереса. Поскольку регистров у процессора не так много, лучшим решением будет не трогать их значения без крайней необходимости. Стек позволяет совместить оба этих, казалось бы несовместимых, требования.

Рассмотрим простой пример для процессоров с системой команд, принятой в IBM-совместимых компьютерах. Пусть некоторый фрагмент должен использовать для промежуточных данных регистры BX и CX. Надежное решение проблемы такое:

```
PUSH BX
PUSH CX
<здесь любые манипуляции с BX и CX>
POP CX
POP BX
```

После завершения указанного фрагмента, постоянство значений сохраняемых регистров гарантируется независимо от длины и сложности фрагмента.

2. ***Использование стека в некоторых алгоритмах***. Многие алгоритмы существенно упрощаются при использовании стека для хранения данных.

Например: алгоритм формирования десятичных цифр произвольного двоичного числа N выполняется по формулам:

$$C_i := N \bmod 10;$$
$$N := N \operatorname{div} 10;$$

Алгоритм генерирует цифры C_i «задом наперед», т.е. сначала единицы, потом десятки и т.д., в то время как печатать требуется в строго обратном порядке. Если получающиеся при вычислениях цифры помещать в стек, то при считывании они будут автоматически возвращаться именно в нужной последовательности.

3. ***Использование стека для вычисления выражений***. Стек является идеальным средством для вычисления произвольных арифметических и логических выражений.

Например: требуется вычислить выражение 2*(3+4). Ход вычислений с помощью стека иллюстрирует следующая таблица:

Команда	Стек
PUSH 2	2

PUSH 3	3 2
PUSH 4	4 3 2
ADD	7 2
MUL	14

При использовании стековой формы вычислений скобки исчезают, поэтому такую форму записи выражений называют ***обратной бесскобочной***, или ***польской,*** ее автором является известный польский математик Ян Лукасевич. Достоинством польской системы представления арифметических выражений является полная однозначность ее интерпретации. Такая система ввода выражений использовалась раньше во многих программируемых калькуляторах.

4. ***Использование стека для передачи параметров и создания локальных переменных***. При входе в процедуру в стеке выделяется место под локальные переменные, а по завершении работы это место освобождается. Данный алгоритм сочетается с требованиями рекурсии: при каждом вызове в стеке создаются свои собственные копии переменных, которые никак не мешают друг другу.

На практике процесс создания в стеке структур переменных выглядит несколько сложнее, поскольку согласно принципам программирования вложенным процедурам не запрещается пользоваться переменными процедур вышележащих уровней. Поэтому трансляторы организуют ***фреймы локальных переменных*** так, чтобы обеспечивать возможность доступа от фреймов одного уровня к другому.

Описанный выше механизм требует выделения достаточного объема стековой памяти. При неправильной организации (бесконечной) рекурсии программа «вылетает» именно в связи с достижением границы стека. Переменные-параметры (их значения или адреса: в первом случае говорят о передаче параметра по значению, а во втором – по ссылке) также передаются при обращении к подпрограмме через стек.

5. ***Использование стека для реализации вложенных структур***. Вложенными могут быть не только подпрограммы, но и другие алгоритмические структуры, в частности, развилки и циклы. Поэтому и здесь стеку находится применение.

Вложенность конструкций так же, как система открывающихся и закрывающихся скобок, а значит, и здесь стековая память подходит идеально. При этом любые из перечисленных выше применений могут произвольным образом комбинироваться друг с другом.

Возможно построение вычислительной машины на основе стековых принципов. Хорошим примером стековой организации может служить виртуальная Java-машина.

6. ***Использование стека для организации механизма подпрограмм***. (См. лабораторную работу 8)

Реализация стека в разных вычислительных машинах

Стек имеет аппаратную реализацию. Для IBM-совестимых и других компьютеров организацию и средства доступа к стеку мы уже рассмотрели.

В серверных RISC-процессорах существует еще одна весьма своеобразная и интересная аппаратная реализация некоторой разновидности стека. Ее часто ***регистровым окном***. Основная идея состоит в том, что в любой момент времени процессор «видит» только одно регистровое окно, причем регистры в нем всегда пронумерованы единообразно. Окно можно аппаратно переключать на другие регистры с

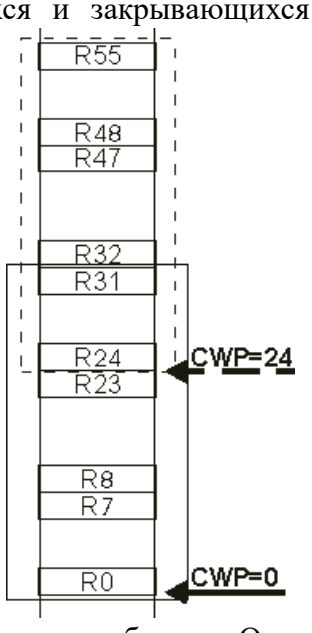


Рис.3.

помощью специального регистра **CWP** (Current Window Pointer) – *указателя текущего окна*.

Например, на рисунке 3 показано, что при установке значения CWP = 24 (новое положение окна показано пунктиром) регистр R24 приобретает имя R0, R25 – R1 и т.д.

Регистровое окно делится на три области:

- область регистров параметров (R0-R7);
- область локальных регистров (R8-R23);
- область временных регистров (R24-R31).

Важно подчеркнуть, что области параметров и временных регистров при переключении окон перекрываются, что дает возможность передавать данные из одного окна в другое. Например, в наших обозначениях, код, занесенный при CWP = 0 в регистр R24, после переключения окна будет доступен под именем R0, а в R31 – в качестве R7. Зато регистры R0-R23 в результате переключения окажутся надежно сохранены, поскольку станут недоступны процессору.

Команды для работы с кадром стека

В системе команд процессоров x86 существуют также специальные команды для работы с кадром стека подпрограммы: ENTER и LEAVE.

Команда ENTER обычно размещается в начале подпрограммы. У неё два непосредственных операнда: первый операнд — размер памяти, выделяемой под локальные переменные, второй операнд — уровень вложенности. Второй операнд этой команды позволяет организовывать вложенные области видимости, как в некоторых языках высокого уровня. В ассемблере эти возможности практически не используются.

Команда LEAVE не имеет операндов и аналогична по действию двум командам:

```
MOV SP, BP
POP  BP
```

Следует отметить, что компиляторы эти команды практически не используют. Дело в том, что эти команды на современных процессорах выполняются гораздо медленнее, чем пролог из 3-х команд. Самый быстрый вариант такой:

```
                Процедура с локальными переменными
PUSH  BP        - сохранение BP
MOV BP,SP       - копирование указателя стека в BP
SUB   SP,n      - выделение памяти для локальных переменных
...
LEAVE           - освобождение памяти, восстановление BP
RET             - возврат из процедуры
```

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом.
2. С помощью Debug разберитесь, как работает следующая последовательность команд:

```
MOV SP,172
PUSH AX
PUSH BX
MOV SP,170
POP BX
```

3. Используя команды *PUSH и POP* промоделировать вычислительный процесс по польской системе.
4. Напишите программу обмена при посредстве стека содержимого регистров CX и DX
5. Сохранить в стеке содержимое всех регистров и регистра флагов, заменить их любыми произвольными значениями, затем восстановить их.

Контрольные вопросы

1. Каков синтаксис команд условного перехода?
2. Какие флаги анализируют команды безусловного перехода?
3. Как формируется машинный код команды безусловного перехода ассемблера?

4. Что такое близкий и дальний переходы в ассемблере?
5. Как различить в командах прямой и косвенный переходы?

Лабораторная работа 10-11. Постоянная память. основы работы с BIOS Setup Utility

Цель работы: Познакомиться с особенностями постоянной и полупостоянной памяти. Познакомиться с программой BIOS, ее системой звуковых сигналов при тестировании системы. Научиться устанавливать настройки BIOS.

Основные теоретические сведения

Постоянная память

Системная постоянная память (ПЗУ) занимает сравнительно небольшой объем, информация в которой сохраняется и после выключения питания компьютера. Однако ее значение для компьютера очень велико. Существует множество типов энергонезависимой памяти: ROM, PROM, EPROM, EEPROM, Flash Memory, различающихся по своим свойствам, обусловленным способом построения запоминающих ячеек, и сферам применения.

Запись информации в энергонезависимую память, называемая программированием, обычно существенно сложнее и требует больших затрат времени и энергии, чем считывание. Основным режимом работы такой памяти является считывание данных, а некоторые типы после программирования допускают только считывание, что и обусловливает их общее название ROM (Read Only Memory - память только для чтения) или ПЗУ (постоянное запоминающее устройство).

Самые первые постоянные запоминающие устройства выполнялись на магнитных сердечниках, где информация заносилась их прошивкой проводниками считывания. С тех пор применительно к программированию ПЗУ укоренилось понятие «прошивка».

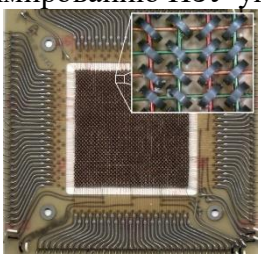


Рис.1.

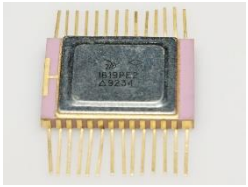


Рис.2.

По возможности программирования микросхемы ПЗУ различают:

- Микросхемы, программируемые при изготовлении (масочные ПЗУ) – ROM (рис.2)
- Микросхемы, программируемые однократно после изготовления перед установкой в целевое устройство (прожигаемые ПЗУ, программируемые на специальных программаторах (рис.3) – PROM (Programmable ROM) (рис.4).
- Микросхемы, стираемые и программируемые многократно (перепрограммируемые):
  a. EPROM (Eraseable PROM) - стираемые ПЗУ с помощью ультофиолетовых лучей) (рис.5);
  b. EEPROM (Flash Memory) - стираемые ПЗУ с помощью электрических сигналов (рис.6)

Рис.3.

Рис.4.

Рис.5

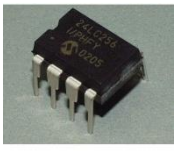


Рис.6.

ПЗУ, как устройство имеет достаточно сложную организацию и состоит из:

- Непосредственно постоянной памяти (ROM),

- Перепрограммируемой постоянной памяти (Flash Memory),
- Память CMOS RAM, питаемой от батарейки,
- Видеопамяти и некоторых других видов.

Постоянная память (ROM, Read Only Memory — память только для чтения) - энергонезависимая память, используется для хранения данных, которые никогда не требуют изменения. Содержание памяти специальным образом «зашивается» в устройстве при его изготовлении для постоянного хранения. Прежде всего в постоянную память записывают программу управления работой самого процессора, программы запуска и остановки компьютера, тестирования устройств.

Перепрограммируемая постоянная память (Flash Memory) — энергонезависимая память, допускающая многократную перезапись своего содержимого. Важнейшая микросхема Flash-памяти — модуль BIOS (рис.7, 8). Роль BIOS двоякая: с одной стороны это неотъемлемый элемент аппаратуры, а с другой стороны — важный модуль любой операционной системы. ROM BIOS (ROM Basic Input/Output System - базовая система ввода/вывода) содержит функционально полный набор программ нижнего уровня для управления устройствами ввода/вывода. Поэтому даже до загрузки в оперативную память исполняемых программ с диска компьютер имеет возможность обслуживать клавиатуру, дисплей, подавать звуковые сигналы, общаться с дисками и т.д. Важно отметить, что большинство современных видеоадаптеров, сетевых адаптеров, модемов, дисковых контроллеров, принтеров имеют собственную систему BIOS, которая дополняет или даже заменяет системную BIOS во время основной работы. Вызов программ BIOS осуществляется через программные или аппаратные прерывания, для чего BIOS формирует соответствующую таблицу векторов прерываний. BIOS материнской платы отвечает за инициализацию (подготовку к работе), тестирование и запуск всех ее компонентов. Операционная система и прикладные программы работают с аппаратным обеспечением компьютера посредством BIOS, которая переводит понятные пользователю команды операционной системы на язык, понятный компьютеру.

Полупостоянная память CMOS RAM - разновидность постоянного запоминающего устройства с невысоким быстродействием и минимальным энергопотреблением от батарейки. Используется для хранения информации о конфигурации и составе оборудования компьютера, а также о режимах его работы

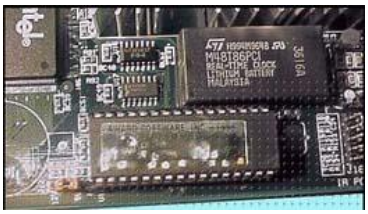


Рис.7. Интегральные схемы BIOS и CMOS

Содержимое CMOS изменяется специальной программой Setup, находящейся в BIOS (рис.8). Иногда можно столкнуться с ситуацией, что компьютер перестал стартовать из-за неправильных настроек BIOS, например, слишком завышена частота FSB шины или слишком большой множитель процессора и т.д. В таком случае, необходимо вернее обнулить CMOS память, в которой хранятся эти настройки (сбросить настройки BIOS). Для этой цели на материнской плате имеется переключатель Clear CMOS (могут использоваться другие названия: Clear RTC, CRTC, CLRTC, CL_CMOS, CL_RTC, CCMOS и т.д.) (рис.9).

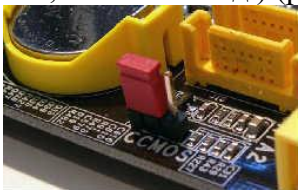


Рис.8. *Рис. 9*

Использование Debug для просмотра служебных областей BIOS

Проверка параллельных и последовательных портов

Первые 16 байт области данных BIOS содержат адреса параллельных и последовательных портов. Поэтому с помощью следующей команды можно проверить эти порты:

-D 40:00

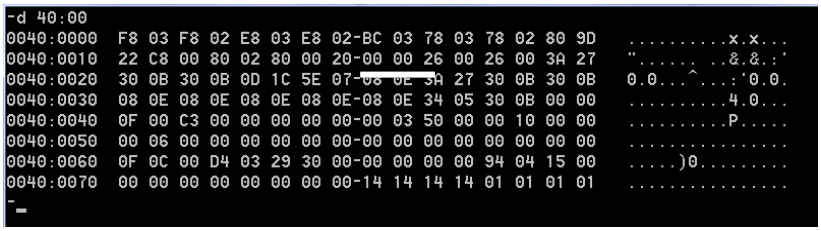


Рис.10

Первые выведенные восемь байтов указывают на адреса последовательных портов COM1 - COM4. Следующие 8 байтов указывают на адреса параллельных портов LPT1 - LPT4.

Например, если на компьютере есть один параллельный порт, то первые два байта будут 7803 (рис.10). Адрес порта записывается в обращенной последовательности, т.е.0378.

Проверка оборудования

Первые два байта, располагающиеся в BIOS по адресу 410h, содержат информацию об установленном в системе оборудовании. Эти байты можно найти командой:

-D 40:10

Предположим, первые два байта 23 44. Расшифруем их для получения информации об установленных устройствах. Для этого необходимо обратить эти байты (44 23), затем перевести их в двоичную систему счисления:

Значение бита	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0
Позиция бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Дешифруем полученные данные:

Биты	***Устройство***
15, 14	Число параллельных портов (01 = 1 порт, ...)
11, 10, 9	Число последовательных портов (..., 010 = 2 порта, ...)
7, 6	Число дисководов (00 = 1 дисковод, 01 = 2, 10 = 3, 11 = 4)
5, 4	Начальный видеорежим (01 = 40x25 цветной, 10 = 80x25 цветной, 11 = 80x25 монохромный)
1	Присутствие математического сопроцессора (0 = нет, 1 = есть)
0	Наличие привода для дискет (0 = нет, 1 = есть)

Проверка состояния регистра клавиатуры

В области данных BIOS по адресу 417h находится первый байт, который хранит состояние регистра клавиатуры.

Для этого необходимо: выключить Num Lock и Caps Lock, затем набрать команду:

-d 40:17 Первый байт будет равен 00.

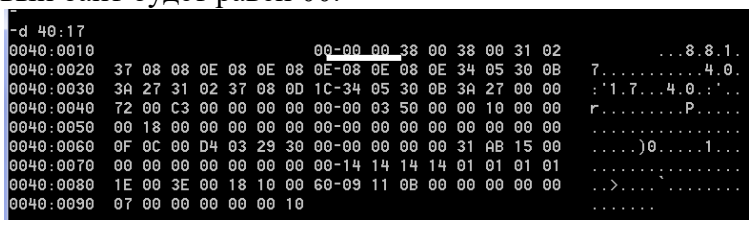


Рис.11

Включить Num Lock и Caps Lock, снова выполнить команду. Теперь первый байт равняется 60 (рис.12).

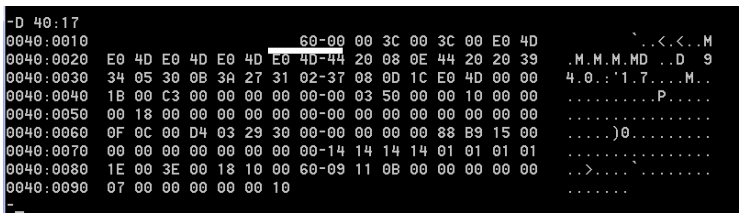


Рис.12

Проверка состояния видеосистемы

По адресу 449h в BIOS находится первая область видеоданных. Для проверки следует набрать: -d 40:49

Первый байт показывает текущий видеорежим 03 - цветной, второй - число столбцов (50 - режим с 80 столбцами). Число строк можно найти по адресу 484h (40:84) (рис.13).

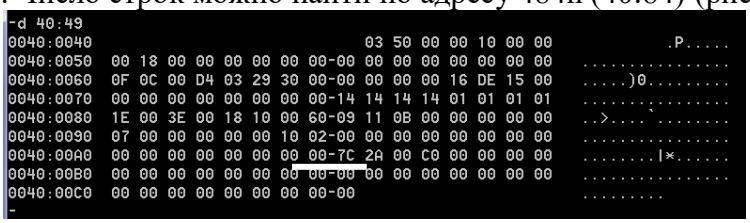


Рис.13

Проверка данных о BIOS

- *Проверка копирайта BIOS и серийного номера.* Сведения об авторских правах на BIOS встроены в ROM BIOS по адресу FE00:0. Строку с копирайтом можно легко найти в ASCII-последовательности, а серийный номер - в виде шестнадцатеричного числа. Хотя, строка с указанием авторских прав может быть длинной и не умещаться в выведенную область памяти. В этом случае следует просто ввести еще раз команду D.
- *Проверка даты производства BIOS.* Эта дата также записана в ROM BIOS начиная с адреса FFFF:5. После выполнения соответствующей команды в ASCII-последовательности будет находиться дата, записанная в формате мм/дд/гг.
- *Проверка фирмы производителя.* Данную информацию можно получить по адресу F000:E000
- *Проверка информации о системе*. Дата создания *BIOS*, чипсета и т.д. можно получить по адресу: F000:EC6C
- *Информация о таймере BIOS*. Находится по адресу 0000:046C (Отчетливо видно, что значения все время меняются)

Для программиста удобно просматривать сегменты команд, данных и стека (CS, DS и SS, соответственно):

-d DS:0000

-d CS:100

BIOS Setup Utility

Среди программ, содержащихся в BIOS, имеется программа настройки параметров полупостоянной памяти BIOS Setup Utility, которая позволяет изменять данные, хранящиеся в памяти CMOS, с помощью системы меню. Для обеспечения правильной работы операционной системы и прикладных программ с помощью BIOS Setup Utility вводятся параметры всех компонентов компьютера, начиная от оперативной памяти и рабочей частоты процессора и заканчивая режимом работы принтера и других периферийных устройств. Правильно настроив содержимое BIOS компьютера, можно увеличить производительность его работы до 30%.

Замечание: неосторожные действия пользователя, как правило, не могут привести к физическому повреждению компьютера — он может лишь перестать загружаться. Современные BIOS имеют довольно обширные средства автоконфигурирования, поэтому роль пользователя в установке «правильных» параметров можно свести к минимуму. В последнее время в программе установки параметров появился пункт

«Загрузить оптимизированные параметры». Выбор этого пункта позволяет пользователю установить параметры «параметры по умолчанию» для имеющегося оборудования.

Программа установки параметров BIOS Setup Utility недоступна пользователю во время работы компьютера. Вход в BIOS Setup Utility обычно выполняется путем нажатия клавиши [Del] во время загрузки компьютера. Так же встречаются версии BIOS, вход в настройки которой выполняется с использованием других клавиш или их сочетаний.

Microsoft Virtual PC

Для изучения основ работы в CMOS Setup Utility можно использовать свободно-распространяемое программное обеспечение ***Microsoft Virtual PC*** (http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=04d26402-3199-48a3-afa2-2dc0b40a73b6), поскольку в данной виртуальной машине эмуляция BIOS Setup реализована наиболее полноценно (версия AMI BIOS 2.10), хотя и не имеет таких широких возможностей настройки, реализованных для реального BIOS.

Программа Microsoft Virtual PC предназначена для создания полностью виртуального компьютера. Она позволяет выделить часть места на жёстком диске, создать в этой области виртуальную машину, установить на неё отдельную операционную систему, необходимые программы и безбоязненно экспериментировать с этим компьютером, не нанося вред реальному компьютеру.

Подготовка к выполнению работы

При подготовке к выполнению лабораторной работы необходимо добавить в Virtual PC новую виртуальную машину, но не устанавливать на нее операционную систему, так как для доступа к BIOS Setup Utility не требуется дополнительное программное обеспечение.

1. Запустить программу Virtual PC (*Пуск — Все программы — Microsoft Virtual PC*).

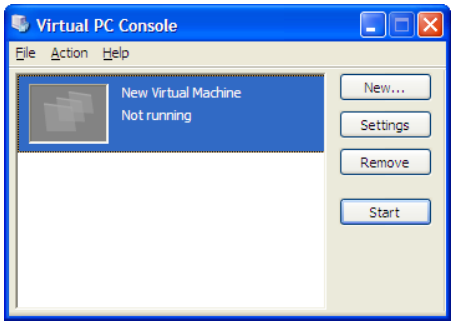


Рис. 14. Интерфейс виртуальной машины Virtual PC

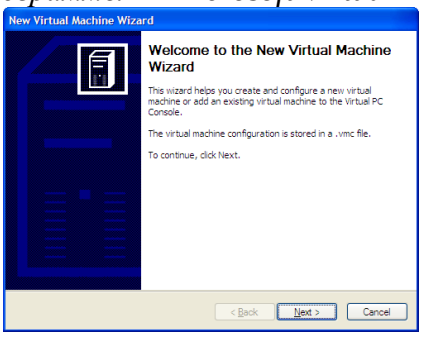


Рис. 15. Мастер создания и конфигурирования виртуальной машины

2. Добавить новую виртуальную машину. Для этого нажать на кнопку «**New**…». Появится мастер создания виртуальной машины (рис.14).

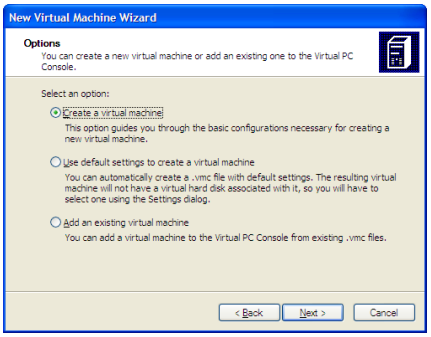


Рис.16. Мастер создания и конфигурирования виртуальной машины

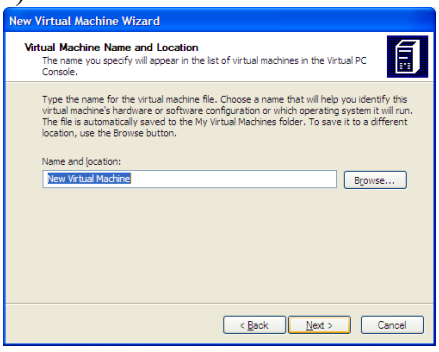


Рис. 17. Мастер создания и конфигурирования виртуальной машины

3. Нажать кнопку «**Next**» («**Далее**») (рис.15).
4. Выбрать «**Create a virtual machine**». Нажать «**Next**» (рис.16).

5. Ввести название виртуальной машины. Например, «**For BIOS ONLY**». Нажать кнопку «**Next**» (рис.17).
6. Выбрать тип операционной системы оставить «**Other**». Нажать кнопку «**Next**».
7. Выбрать «**Use the recommended RAM**». Нажать кнопку «**Next**» (рис.19).
8. Выбрать «A new virtual hard disk», что означает создание нового виртуального жесткого диска. Нажать «Next» (рис.20).
9. Оставить предложенные параметры, нажать «**Next**» (рис.21).
10. Нажать «**Finish**» (рис.21). Таким образом, в списке виртуальных машин появится новая — «**For BIOS ONLY**» (рис. 23).

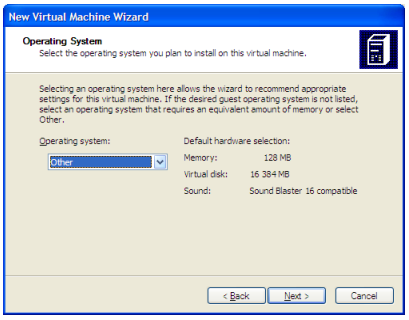


Рис. 18. Мастер создания и конфигурирования виртуальной машины

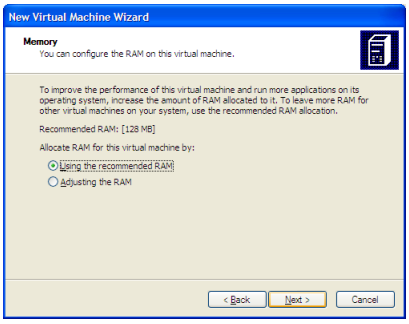


Рис.19. Мастер создания и конфигурирования виртуальной машины

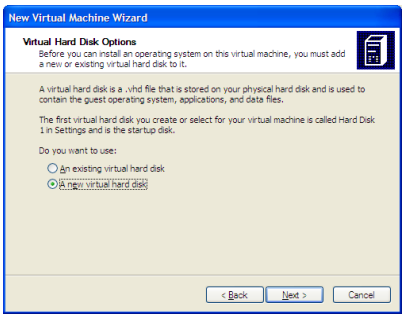


Рис.20. Мастер создания и конфигурирования виртуальной машины

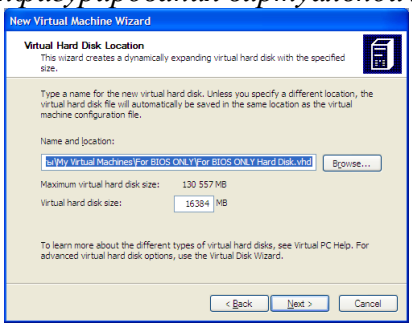


Рис.21. Мастер создания и конфигурирования виртуальной машины

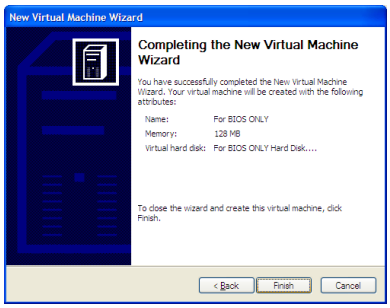


Рис. 22. Мастер создания и конфигурирования виртуальной машины

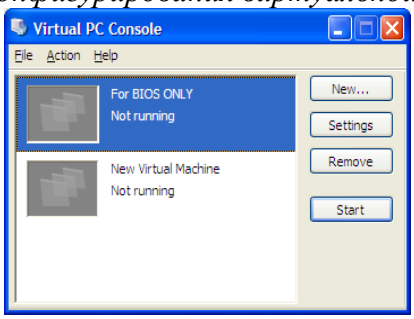


Рис. 23. Список виртуальных машин

11. Чтобы запустить виртуальную машину следует нажать на кнопку «**Start**». Как только начнется запуск, сразу следует нажать клавишу «**Delete**» на клавиатуре. Обычно при загрузке компьютера после нажатия именно этой клавиши происходит вход в настройки BIOS.
12. После загрузки BIOS Setup Utility экран может иметь следующий вид (рис.24, 25):

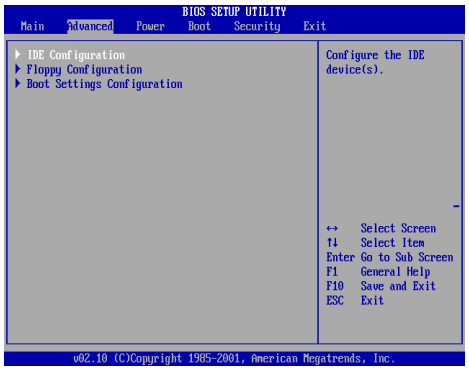


Рис.24. Интерфейс AMI BIOS

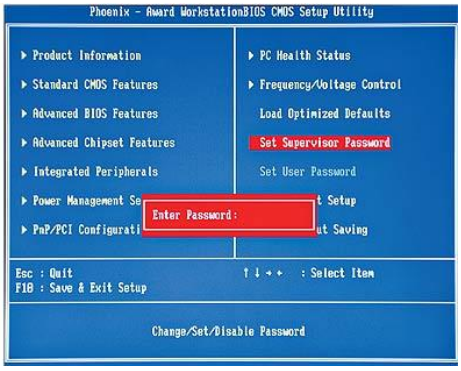


Рис.25. Интерфейс Award BIOS

Примечание: Чтобы скопировать содержимое окна, необходимо в главном меню виртуальной машины выбрать пункт Edit — Select All, а затем Edit — Copy.

Основные настройки BIOS

В большинстве компьютеров используются BIOS компаний AMI, Award и Phoenix. Вследствие большого разнообразия моделей материнских плат существует много различных версий BIOS. Поэтому настройки BIOS на разных компьютерах могут иметь различный вид. В рамках лабораторной работы рассмотрен Award BIOS.

Главное меню. При входе в BIOS пользователь сразу попадает в главное меню. Как правило, там имеется несколько разделов, где и настраиваются различные параметры. В зависимости от версии BIOS, справа или внизу экрана расположена строка подсказки. Прежде чем вносить какие-либо изменения в BIOS, на всякий случай следует записать старые настройки, чтобы при необходимости вернуться к ним. На некоторых компьютерах можно также сохранить отображаемое на экране окно BIOS нажатием соответствующей клавиши, если это невозможно, следует сфотографировать экран.

Многие настройки BIOS можно включить (рядом с ними появится слово Enabled) или отключить (соответственно Disabled). В других настройках, как правило, предлагается возможность выбора опций из списка.

Из главного меню BIOS можно перейти в различные подменю (чаще их называют разделами). В рамках лабораторной работы рассмотрены наиболее важные из них. В некоторых из этих разделов доступны полезные функции, которые следует активировать.

Load Optimized Defaults. Этот пункт предназначен для возврата настроек BIOS в первоначальное состояние. К этой возможности следует прибегать в том случае, когда компьютер после изменения BIOS работает со сбоями и при этом не были сохранены настройки, предшествовавшие корректировке (рис.26).

Рис.26.

Рис.27.

Standart CMOS Features. Данный пункт позволяет установить текущие настройки (рис.27):

- *Date/Time*. Здесь можно установить дату и время. Но делать это средствами операционной системы удобнее.
- *IDE Channel 0/1/2.* В этом разделе указаны обозначения установленных на компьютере дисковых накопителей и оптических приводов.
- *Video*. Значение этого параметра должно быть EGA/VGA. В противном случае размер шрифта в настройках BIOS значительно увеличится, и не будет возможности прочитать сообщение полностью.
- *Halt On*. Здесь можно указать, как BIOSy следует реагировать на ошибки при тестировании оборудования сразу после включения компьютера. Как правило, при возникновении ошибки на экране появляется короткое сообщение, и для продолжения загрузки компьютера требуется нажатие клавиши. Если требуется, чтобы загрузка системы производилась вне зависимости от результатов проверки, следует выбрать опцию «All, But Disk/Key». В этом случае запуск компьютера произойдет, даже если в ходе тестирования будут обнаружены ошибки.

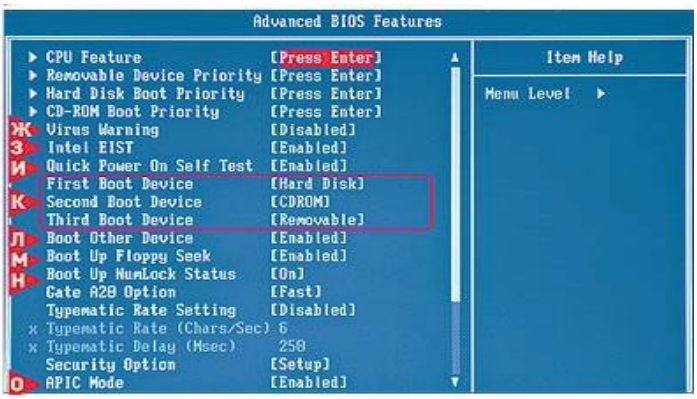


Рис. 28.

Advansed BIOS Features. Содержит дополнительные настройки BIOS (рис.28):

- *VirusWarning.* Если активировать эту функцию, то BIOS будет предупреждать о том, что некоторая программа пытается записать данные в загрузочную область жесткого диска. Эта функция призвана защитить компьютер от вирусов, однако на практике она не обеспечивает должной защиты, и даже может препятствовать установке операционной системы. К тому же большинство RAID-контроллеров производит запись данных о конфигурации массива на каждом жестком диске, что позволяет восстановить информацию в случае выхода из строя одного из дисков. Рекомендация: данную опцию лучше не активировать.
- *Intel EIST*. Эта настройка должна быть активирована. Она отвечает за функции энергосбережения процессоров Intel. Функция для процессоров AMD называется Cool & Quiet.
- *Quick Power On Self Test.* При активации данной функции, BIOS произведет очень подробное тестирование всех компонентов компьютера. Необходимость в этом возникает крайне редко, например, в том случае, если компьютер «зависает» перед запуском операционной системы.
- *First/Second/Third Boot Device*. Этот пункт содержит информацию о том, в какой последовательности BIOS обращается к дискам при загрузке операционной системы. В обычных условиях работы в качестве первого загрузочного диска должен быть установлен жесткий диск. В этом случае BIOS не будет проверять, находится ли в дисководе или оптическом приводе носитель. Это экономит время при запуске.

- *Boot Other Device*. Функция используется для компьютеров, подключенных к разветвленным корпоративным сетям. Если она активирована, операционную систему можно запустить со специального сервера.
- *Boot Up NumLock Status.* Функция позволяет активировать режим использования клавиш цифрового блока клавиатуры в качестве навигационных по умолчанию.
- *APIC Mode*. В режиме APIC компоненты материнской платы и модули расширения могут использовать до 16 дополнительных прерываний. Функция должна быть активирована. В противном случае вероятны сбои в работе некоторых модулей, например видеоадаптера и звуковой карты.

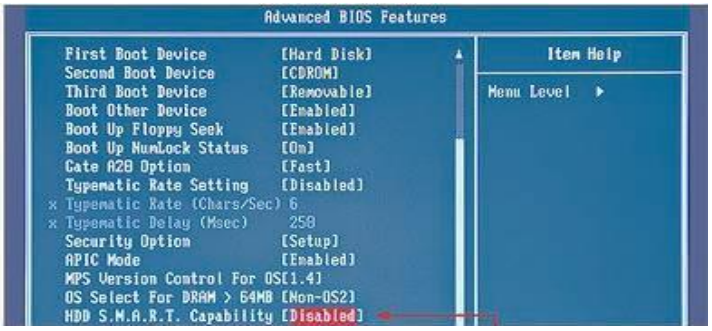


Рис.29.

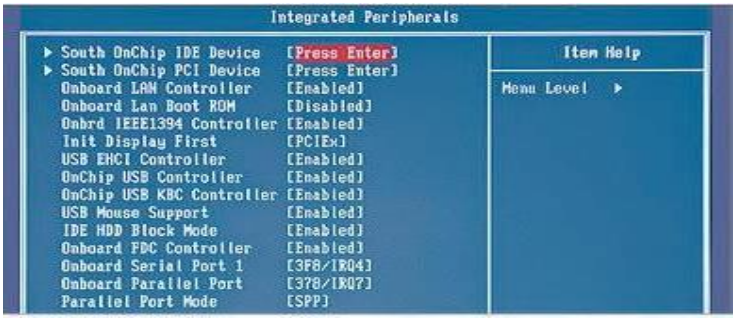


Рис.30.

- *HDD S.M.A.R.T Capability* (Возможность S.M.A.R.T диагностики) — позволяет разрешать/запрещать возможность диагностики состояния жесткого диска в соответствии с требованиями стандарта S.M.A.R.T. При разрешении параметра и нарушении нормального функционирования жесткого диска BIOS выдает на экран соответствующее сообщение до появления таблицы с характеристиками компьютера. Следует учитывать, что разрешение этого параметра снижает производительность компьютера на несколько процентов.

Integrated Peripherals. В этом разделе собраны параметры для различных интегрированных периферийных устройств, которые поддерживаются южным мостом чипсета: контроллеров гибких и жестких дисков, звуковых и сетевых адаптеров, последовательных, параллельных и USB-портов и др. Состав настроек этого раздела зависит от состава периферийных устройств в конкретной системе. Здесь можно отключить ненужные компоненты материнской платы (рис.30).

Power Management System. В разделе устанавливаются параметры электропитания и режимы энергосбережения. Можно настроить автоматический переход компьютера в условия пониженного энергопотребления, а также заставить его возвращаться в рабочее состояние при наступлении определенных событий.

- *ASPI Suspend Type*. В режиме S1 процент экономии энергии невелик. Оптимальным является режим S3. В этом случае компьютер в «спящем режиме» практически не будет потреблять электроэнергию (рис.31.)

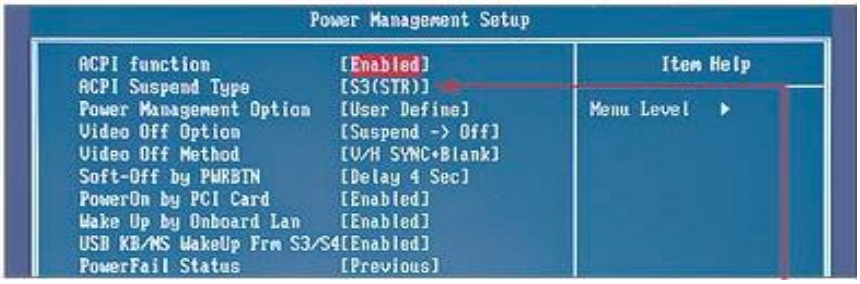


Рис.31.

PNP/PCI Configurations. Параметры этого раздела управляют способом распределения ресурсов между периферийными устройствами, регулировкой прерываний. Обычно эту функцию поручают системе, оставляя настроенное по умолчанию автоматическое распределение ресурсов (рис.32).

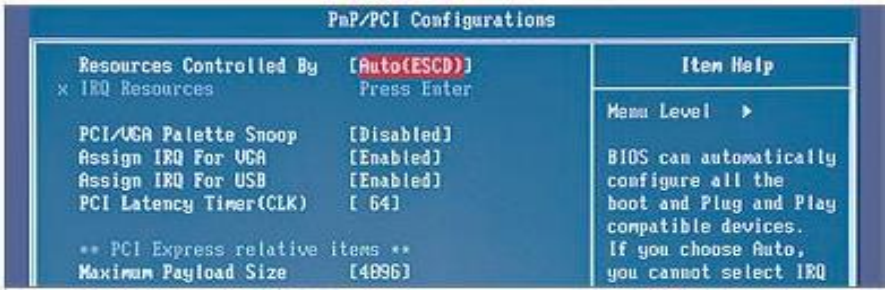


Рис.32.

PC Health Status. Все современные системные платы оснащены датчиками контроля рабочих температур, напряжений и скоростей вращения вентиляторов. Их текущие показания отображаются в отдельном разделе BIOS Setup с названием PC Health Status или Hardware Monitor. Показания датчиков используются в автоматических системах защиты, выставляются соответствующие параметры (рис.33).

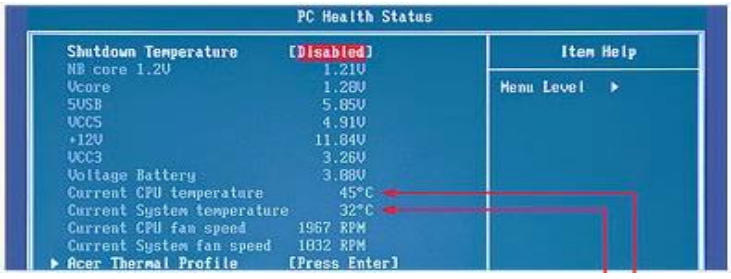


Рис.33. Температура системы и процессора

Frequency / Voltage Control. В этом разделе устанавливаются рабочие частоты и напряжения для процессора, чипсета, оперативной памяти, видеоадаптера и др. При параметрах по умолчанию все частоты и напряжения в современных компьютерах настраиваются автоматически, что обеспечивает надежную работу системы. Если параметры этого раздела изменить вручную, можно выполнить разгон, то есть заставить процессор, память и другие компоненты работать на более высоких частотах (рис.34).

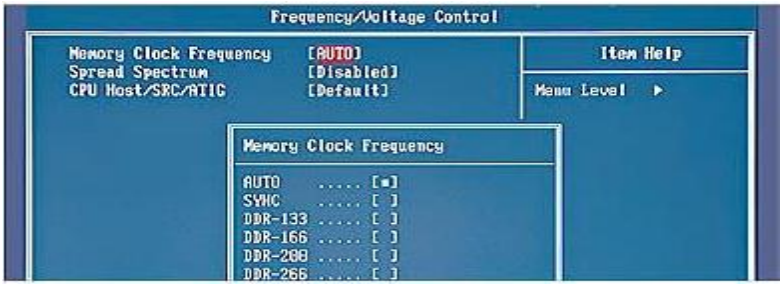


Рис.34.

Load Fail-Safe Defaults (Load BIOS Setup Defaults). Команда сбрасывает все настройки BIOS до значений по умолчанию. При этом устанавливаются наиболее безопасные значения всех параметров, обеспечивающие высокую стабильность работы системы. При выборе этого пункта обычно появляется окно, в котором нужно подтвердить выбранное действие нажатием клавиши Y

Set Supervisor Password, Set User Password. Команды устанавливают, соответственно, административный и пользовательский пароль на вход в BIOS или на загрузку компьютера.

Save & Exit Setup. Сохранить параметры и выйти.

Exit Without Saving. Выйти без сохранения настроек.

Upgrade BIOS. Данная команда есть не во всех BIOS, она используется для обновления BIOS в домашних условиях. Перед ее выполнением нужно иметь новую версию BIOS.

Более подробную информацию о настройках BIOS можно найти на сайте http://www.razlib.ru/kompyutery_i_internet/ochen_horoshii_samouchitel_polzovatelja_kompyuterom_kak_samomu_ustranit_90_neispravnostei_v_kompyutere_i_uvelichit_ego_vozmozhnosti/p4.php

2. Система звуковых сигналов BIOS

При включении компьютера каждый раз запускается диагностическая программа Power-On-Self-Test (POST), которая выполняет проверку важнейших компонентов компьютера (начиная от центрального процессора и заканчивая контроллером клавиатуры). Результаты тестирования выдаются на динамик компьютера в виде специального звукового сигнала. Расшифровки этих сигналов, при выявлении дефектных компонентов компьютера, представлена ниже.

Award BIOS

Нет сигналов - неисправен или не подключен к материнской плате блок питания.
Непрерывный сигнал - неисправен блок питания.
1 короткий сигнал - ошибок не обнаружено.
2 коротких сигнала - незначительные ошибки. Необходимо проверить надежность контактов шлейфов в разъемах IDE/SATA-контроллеров на материнской плате и на жестких дисках.
3 длинных сигнала - ошибка контроллера клавиатуры. Возможно необходима замена материнской платы.
1 длинный и 1 короткий сигналы - обнаружены проблемы с оперативной памятью.
1 длинный и 2 коротких сигнала - обнаружены проблемы с видеоадаптером.
1 длинный и 3 коротких сигнала - ошибка инициализации клавиатуры.
1 длинный и 9 коротких сигналов - ошибка при чтении данных из микросхемы постоянной памяти.
1 длинный повторяющийся сигнал - неправильная установка модулей памяти.
1 короткий повторяющийся сигнал - проблемы с блоком питания.

AMI BIOS

1 короткий сигнал - ошибок не обнаружено.
Нет сигналов - неисправен или не подключен к материнской плате блок питания.
2 коротких сигнала - обнаружены проблемы с оперативной памятью.
3 коротких сигнала - ошибка при работе основной памяти (первых 64 Кбайт).
4 коротких сигнала - неисправен системный таймер.
5 коротких сигналов - неисправен центральный процессор.
6 коротких сигналов - неисправен контроллер клавиатуры.
7 коротких сигналов - неисправна материнская плата.
8 коротких сигналов - проблемы с видеоадаптером.
9 коротких сигналов - ошибка контрольной суммы содержимого микросхемы BIOS.
10 коротких сигналов - невозможно произвести запись в CMOS-память.
11 коротких сигналов - неисправна внешняя кэш-память.
1 длинный и 2 коротких сигнала - неисправен видеоадаптер.
1 длинный и 3 коротких сигнала - неисправен видеоадаптер.
1 длинный и 8 коротких сигналов - проблемы с видеоадаптером или не подключен монитор.

Phoenix BIOS

Phoenix BIOS обладает уникальной системой чередующих сигналов.
1-1-3 - ошибка записи/чтения данных CMOS.
1-1-4 - ошибка контрольной суммы содержимого микросхемы BIOS.
1-2-1 - неисправна материнская плата.
1-2-2 - ошибка инициализации контроллера DMA.
1-2-3 - ошибка при попытке чтения/записи в один из каналов DMA.
1-3-1 - обнаружены проблема с оперативной памятью.
1-3-3 - ошибка при тестировании первых 64 Кбайт оперативной памяти.
1-3-4 - ошибка при тестировании первых 64 Кбайт оперативной памяти.
1-4-1 - неисправна материнская плата.
1-4-2 - обнаружены проблемы с оперативной памятью.
1-4-3 - ошибка системного таймера.
1-4-4 - ошибка обращения к порту ввода/вывода. Ошибка может быть вызвана периферийным устройством, использующим данный порт для своей работы.
3-1-1 - ошибка инициализации второго канала DMA.
3-1-2 - ошибка инициализации первого канала DMA.
3-1-4 - неисправна материнская плата.
3-2-4 - ошибка контроллера клавиатуры.
3-3-4 - ошибка при тестирования видеопамяти.
4-2-1 - ошибка системного таймера.
4-2-3 - ошибка при работе линии A20. Неисправен контроллер клавиатуры.
4-2-4 - ошибка при работе в защищенном режиме. Возможно, неисправен центральный процессор.
4-3-1 - ошибка при тестировании оперативной памяти.
4-3-4 - ошибка часов реального времени.
4-4-1 - ошибка тестирования последовательного порта. Может быть вызвана устройством, использующим последовательный порт для своей работы.
4-4-2 - ошибка тестирования параллельного порта. Может быть вызвана устройством, использующим параллельный порт для своей работы.
4-4-3 - ошибка при тестировании математического сопроцессора.

IBM BIOS

1 короткий сигнал - ошибок не обнаружено.
Нет сигналов - неисправен блок питания.
Непрерывный сигнал - неисправен блок питания.
Повторяющиеся короткие сигналы - неисправен блок питания.
1 длинный и 1 короткий - неисправна материнская плата.

1 сигнал и пустой экран - неисправна видеосистема.
1 длинный и 2 коротких сигнала - неисправна видеосистема (Mono/CGA).
1 длинный и 3 коротких сигнала - неисправна видеосистема (EGA/VGA).
2 коротких сигнала - неисправна видеосистема (не подключен монитор).
3 длинных сигнала - ошибка контроллера клавиатуры.

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом и требованиями к отчету по лабораторной работе.
2. Используя debug выяснить:
    a. Дату производства микросхемы ПЗУ;
    b. Тип, версию, производителя и дату BIOS;
    c. Параметры системных часов;
    d. Установленное оборудование;
    e. Режим работы видиосистемы
3. Установите, чему равен первый байт в служебной области 40:17:
    a. при включенном Num Lock
    b. при включенном Caps Lock
4. Создать новую виртуальную машину (или использовать доступную реальную)
5. Войти в BIOS.
6. Выяснить и сравнить с ранее полученными результатами: тип и версию BIOS; дату создания BIOS; параметры системных часов и календаря;
7. Установленный максимально поддерживаемый размер памяти.
8. Определить параметры накопителей, подключенных к каналам стандартного IDE и SATA контроллера.
9. Определить текущий порядок опроса накопителей при загрузке.
10. Изменить порядок опроса накопителей при загрузке так, чтобы в первую очередь опрашивался CD ROM, затем жесткий диск. Остальные носители не опрашивать.
11. Включить режим подробного тестирования системы.
12. Установить режим автоматической активации цифровых клавиш на цифровом поле клавиатуры.
13. Отключить COM и LPT порты.
14. Установить режим максимальной экономии энергопотребления
15. Выяснить температуру ЦП и системы.
16. Установить пароль ***qwaszx*** на вход в BIOS Setup Utility.
17. Установите пароль ***123456*** на вход в виртуальную машину .
18. После фиксации всех заданий верните все измененные параметры BIOS в исходное состояние.

Контрольные вопросы

1. Чем отличается постоянная память от оперативной?
2. Какова структура ПЗУ?
3. В какой памяти сохраняются программы BIOS?
4. Какая информация сохраняется в энергонезависимой памяти?
5. С какой целью в адресном пространстве предусматривают адреса постоянной памяти?

Лабораторная работа 12-13. Обслуживание устройств внешней памяти

***Цель работы*:** Познакомиться с историей развития устройств для хранения информации. Изучить теоретические основы хранения информации на различных носителях. Познакомиться с программами обслуживания носителей информации. Научиться форматировать, тестировать накопители, осуществлять проверку надежности носителя, оптимизировать, реализовывать информационную безопасность.

Основные теоретические сведения

Основные сведения об устройстве внешней памяти компьютера

В 1945 г. Джон фон Нейман (1903-1957) выдвинул идею использования внешних запоминающих устройств для хранения программ и данных. Нейман разработал принципиальную схему ЭВМ, которой соответствуют и все современные компьютеры в плане организации памяти.

Внешняя память предназначена для долговременного хранения программ и данных. Устройства внешней памяти (накопители) являются энергонезависимыми, выключение питания не приводит к потере данных. Они могут быть встроены в системный блок или выполнены в виде самостоятельных блоков, связанных с системным через его порты. Важной характеристикой внешней памяти служит ее ***объем***. Объем внешней памяти можно увеличивать, добавляя новые накопители. Не менее важными характеристиками внешней памяти являются ***время доступа к информации*** и ***скорость обмена информацией***. Эти параметры зависят от устройства считывания информации и организации типа доступа к ней.

По *типу доступа* к информации устройства внешней памяти делятся на два класса:

- Устройства произвольного доступа. При произвольном доступе время доступа к информации не зависит от ее места расположения на носителе.
- Устройства последовательного доступа. При последовательном доступе время доступа зависит от местоположения информации.

Скорость обмена информацией зависит от скорости ее считывания или записи на носитель, что определяется, в свою очередь, скоростью вращения или перемещения этого носителя в устройстве.

По способу записи и чтения накопители делятся, в зависимости от вида носителя, ***на магнитные, оптические, полупроводниковые*** (flash-накопители).

По отношению к компьютеру накопители могут быть ***внешними*** и ***встраиваемыми*** (внутренними). Внешние накопители имеют собственный корпус и источник питания, что экономит пространство внутри корпуса компьютера и уменьшает нагрузку на его блок питания. Встраиваемые накопители крепятся в специальных монтажных отсеках (drive bays), что позволяет создавать компактные системы, которые совмещают в системном блоке все необходимые устройства. Сам накопитель можно рассматривать как совокупность носителя и соответствующего привода. Различают накопители со ***сменными*** и ***несменными*** носителями

В процессе работы вычислительной системы по мере необходимости производится оперативный обмен информационными массивами между устройствами внешней памяти и оперативной.

Развитие устройств внешней памяти

Первоначально в качестве носителей внешней памяти использовались перфоленты и перфокарты, где информация записывалась с помощью перфорационного устройства (рис.1, 2, 3).

Затем, начиная с ЭВМ 2 поколения стали использоваться устройства хранения информации на магнитных лентах и барабанах (рис.4, 5). Магнитная лента имела достаточную емкость, но обеспечивала последовательный доступ к информации. Магнитный барабан давал возможность произвольного доступа, но был ограничен по емкости

Рис.1

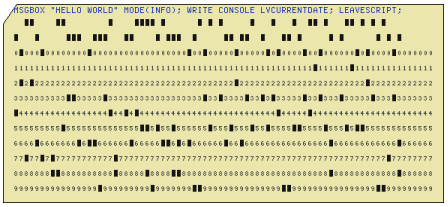


Рис.2.

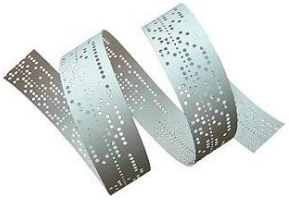


Рис.3.

Рис.4

Рис.5.

Принцип работы магнитных запоминающих устройств основаны на способах хранения информации с использованием магнитных свойств материалов. Как правило, магнитные запоминающие устройства состоят из собственно устройств чтения/записи информации и магнитного носителя, на который, непосредственно, осуществляется запись и с которого считывается информация. Информация на таких носителях записывается на дорожках с помощью магнитной головки. С помощью этого же устройства магнитная запись считывается. Аналогично действует современное устройство внешней памяти - стриммер (накопитель на магнитной ленте). На дорожки ленты записывается все тот же двоичный код: намагниченный участок - единица, не намагниченный - нуль. При чтении с ленты запись превращается в нули и единицы в битах внутренней памяти. Положительным качеством ЗУ на магнитных лентах является их большая емкость при сравнительно низкой стоимости хранения единицы информации. Легкая смена носителя информации позволяет беспредельно наращивать информационный архив

В 1956 году был выпущен первый серийный дисковый накопитель IBM 350 — первое устройство с подвижной головкой для чтения и записи. Он весил более тонны и был способен хранить 5 млн символов в 7-битовой кодировке на 50 «блинах» диаметром 24 дюйма, покрытых краской с окисью железа (рис.6).

Рис.6

Рис.7.

Следующим шагом было создание накопителей со сменными пакетами диаметром 14 дюймов. Эти практичные устройства позволяли многократно увеличивать объем хранимых данных на дисках без существенных затрат. С таких конструкций началось серийное тиражирование дисков, которыми комплектовались до середины 80-х годов мини-ЭВМ и мэйнфреймы (рис.7)

Носитель информации - алюминиевый диск (диаметр 300-350 мм, толщина 1,5-2 мм), покрытый магнитным слоем. В качестве магнитного покрытия использовался сплав никеля-кобальта-фосфора, кобальта-вольфрама. Дисковые носители намагничивались вдоль концентрических полей – дорожек расположенных по всей плоскости носителя.

С появлением ЭВМ четвертого поколения были и усовершенствованы носители информации: появились «винчестеры» (жесткие, Hard Disk) (рис.11) и гибкие (Floppy) диски (рис. 8, 9).

Мягкий магнитный диск (дискета) представляла собой полимерную основу (подложку, выполненную из полимерного немагнитного пластического материала, степень жесткости которого может быть различна в зависимости от реализации) с магнитным слоем. Носитель помещался в бумажный, пластмассовый или другой кожух-корпус.

Рис.8.

Рис.9.

В приводе флоппи-диска (гибкого диска, или просто дискеты) имеются два двигателя: один обеспечивает стабильную скорость вращения вставленной в накопитель дискеты, а второй перемещает головки записи-чтения. Скорость вращения первого двигателя зависит от типа дискеты и составляет от 300 до 360 об/мин. Двигатель для перемещения головок в этих приводах всегда шаговый. С его помощью головки перемещаются по радиусу от края диска к его центру дискретными интервалами. В отличие от привода винчестера головки в данном устройстве не «парят» над поверхностью флоппи-диска, а касаются ее. Для каждого из типоразмеров дискет (5,25 или 3,5 дюйма) использовались свои специальные приводы соответствующего форм-фактора. Емкость дискет 5,25 дюйма составляла 720 Кб или 1. 2 Мб. Емкость дискет 3,5 дюйма - 1,44 Мб. Каждый сменный дисковый магнитный носитель перед использованием необходимо было подготовить к приему данных - форматировать. Форматирование дискет производилось при помощи специального программного обеспечения. Форматирование – это разметка диска на дорожки и сектора. В ходе разметки головки дисковода в определенных местах диска метки дорожек и секторов. Объем хранимой на диске информации зависел от того, сколько дорожек можно разметить на его поверхности и числом секторов на одной дорожке (рис.10). Емкость сектора – 512 байт.

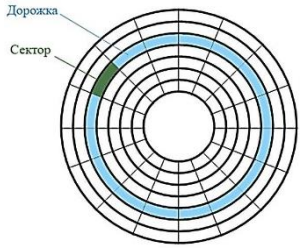


Рис.10.

Рис.11.

Дисководы для работы с флоппи-дисками, как правило, располагались внутри системного блока, однако, выпускались и внешние варианты. Надежность данных носителей оставляла желать лучшего, т.к. они были подвержены вредным воздействиям температурных, гидрометрических, магнитных, механических и др. факторов.

Накопитель на ***жестком диске*** относится к более совершенным и сложным устройствам, его диски вмещают многие Гб информации, передаваемой с огромной скоростью (рис.11). Содержат как механические, так и электронные компоненты. Типовой винчестер состоит из гермоблока и платы электроники. В гермоблоке размещены все механические части, на плате - вся управляющая электроника. Количество дисков может быть различным - от одного до пяти, количество рабочих поверхностей 2 – 10, что определяет емкость жесткого диска. Принцип записи такой же как у дискет. Цифровая информация преобразуется в переменный электрический ток, поступающий на магнитную головку, а затем передается на магнитный диск, но уже в виде магнитного поля. Головки перемещаются с помощью шагового двигателя и как бы "плывут" над поверхностью диска, не касаясь его. На поверхности дисков в результате записи информации образуются намагниченные участки, в форме концентрических окружностей. Они называются магнитными дорожками. Совокупность дорожек, расположенных друг под другом на всех поверхностях, называют *цилиндром*. Все головки накопителя перемещаются одновременно, осуществляя доступ к одноименным цилиндрам с одинаковыми номерами.

Так как головка при поиске информации перемещается только поперек диска, она вынуждена "ждать", пока диск повернется и сектор с запрашиваемыми данными окажется доступным для чтения. Это время зависит только от скорости вращения диска и называется

временем ожидания информации (latency). Общее время доступа к информации определяется временем поиска нужной дорожки на диске и временем позиционирования внутри этой дорожки.

Рис.12

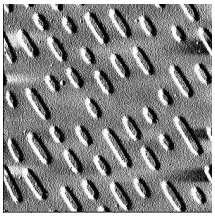


Рис.13

Первые ***оптические диски*** (компакт-диски) появились в 1972 году и продемонстрировали большие возможности по хранению информации. Объемы хранимой на них информации позволяли использовать их для хранения огромных массивов данных (рис.12). Оптические диски имеют высокую надежность и долговечность, что позволило использовать их для архивного хранения информации. Компакт-диск представляет собой поликарбонатную подложку толщиной 1,2 мм и диаметром 120 мм, покрытую тончайшим слоем металла (алюминий, золото, серебро и др.), защищенного слоем лака, на который обычно наносится графическое представление содержания диска. Принцип считывания через подложку позволяет весьма просто и эффективно осуществить защиту информационной структуры и удалить её от внешней поверхности диска.

Информация на диске записывается в виде спиральной дорожки из питов (углубление), выдавленных в поликарбонатной основе. Одна физическая дорожка может быть разбита на несколько логических. Каждый пит имеет примерно 100 нм в глубину и 500 нм в ширину. Длина пита варьируется от 850 нм до 3,5 мкм. Промежутки между питами называются лендом (пространство). Питы рассеивают или поглощают падающий на них свет, а подложка — отражает (рис.13). Различают диски:

- только для чтения - CD-ROM. Процесс изготовления диска и записи информации – единый технологический процесс (штамповка). Для штамповки существует специальная матрица-прототип (мастер-диск) будущего диска, которая выдавливает дорожки на поверхности. После штамповки, на поверхность диска наносят защитную пленку из прозрачного лака.
- для однократной записи - CD-R. Внешне похожи на накопители CD-ROM и совместимые с ними по размерам дисков и форматам записи. Позволяют выполнить одноразовую запись и неограниченное количество считываний. Запись данных осуществляется с помощью специального программного обеспечения. Скорость записи современных накопителей CD-R составляет 4x-48x.
- для многократной записи - CD-RW. Используются для многоразовой записи данных, причем можно как просто дописать новую информацию на свободное пространство, так и полностью перезаписать диск новой информацией (предыдущие данные уничтожаются). Как и в случае с накопителями CD-R, для записи данных необходимо установить в системе специальные программы, причём формат записи совместимый с обычным CD-ROM. Скорость записи современных накопителей CD-RW составляет 2x-24x

DVD (цифровой многоцелевой диск) - оптические диски подобны CD. Это носитель информации, выполненный в форме диска, имеющего такой же размер, как и компакт-диск, но более плотную структуру рабочей поверхности, что позволяет хранить и считывать больший объём информации за счёт использования лазера с меньшей длиной волны и линзы с большей числовой апертурой.

У DVD существуют несколько модификаций:

- DVD-R - самая простая из них отличается тем, что отражающий слой расположен на слое половинной толщины (0,6 мм) (в отличие от CD-R. Емкость такого диска достигает 4,7 GB. Если оба слоя несут информацию (в этом случае нижнее отражающее покрытие

полупрозрачное), то суммарная емкость составляет 8,5 GB (некоторое уменьшение емкости каждого слоя вызывается необходимостью сократить взаимные помехи при считывании дальнего слоя). Используются так же двухсторонние двухслойные диски. В этом случае их емкость составляет 17 GB. Главная причина увеличения емкости диска – уплотнение записи информации, за счет перевода лазера из инфракрасного диапазона (длина волны 780 нм) в красный (с длиной волны 650 нм или 635 нм), что дает двукратное уплотнение дорожек и укорочение длины отражающих питов.

- DVD-RW – формат, позволяющий осуществлять перезапись запись информации после предварительного стирания всего содержимого.
- HD DVD – формат был получен модификацией DVD. В HD-DVD используется такое же расстояние до пишущего слоя, как и в DVD, но применяется голубой лазер, волна у которого короче, поэтому и плотность записи данных выше. В итоге получена ёмкость 15 Гбайт на слой против 4,3 Гбайт у существующих DVD.
- Blu-ray Disc – формат оптических дисков последнего поколения, получил свое название от коротковолнового 405 нм "синего" (технически сине-фиолетового) лазера, который позволит хранить намного больше данных чем на DVD, который имеет те же физические объемы, но использует красный лазер большей длины волны (650 нм). Диски HD-DVD и Blu-ray несовместимы между собой.

CD 0,7 Gb DVD 4,7 Gb Blu-ray 25 Gb

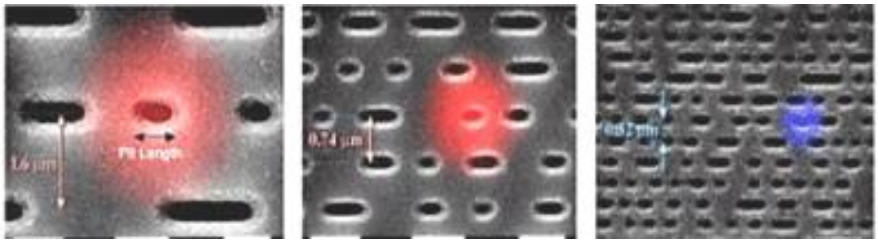


Рис.13

В данный момент наиболее активно в качестве носителей внешней памяти используется ***флэш-накопители*** (рис.14): во-первых, для хранения программных модулей ОС, во-вторых, для хранения пользовательской информации. Флэш-память представляет собой разновидность твердотельной полупроводниковой энергонезависимой перезаписываемой памяти. В настоящее время выпускается два основных типа флэш-памяти: NOR (логика ячеек NOT OR) и NAND (логика ячеек NOT AND). В качестве элементарных ячеек хранения информации используются полевые двух затворные транзисторы с плавающим затвором. Такой транзистор способен сохранять заряд, позволяя определить его наличие. При записи заряд помещается на плавающий затвор. При наличии заряда на плавающем затворе характеристики транзистора изменяются таким образом, что при обычном для операции чтения напряжении токопроводящего канала не возникает. Соответственно по изменению вольтамперных характеристик транзистора можно сделать вывод о наличии или отсутствии заряда. Слово «флэш» в названии памяти возникло благодаря тому, что операция записи требует подачи на сток и управляющий затвор высокого напряжения (отсюда flash, «молния»). Процедуры записи-стирания вызывают износ ячейки флэш-памяти, именно поэтому у таких микросхем ограничен ресурс циклов перезаписи.

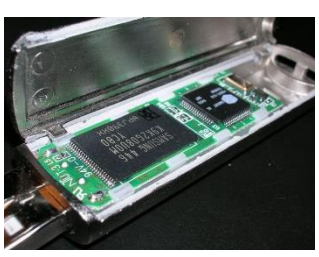


Рис.14

Структура хранения данных на носителях
(таблица размещения файлов)

Магнитный диск, как запоминающее устройство, обладает прямым методом доступа, что позволяет позиционировать считывающие головки на тот сектор дорожки, где содержится начало требуемого файла. Дальнейшее считывание информации зависит от ее расположения в блоке. Реализация данного метода доступа возможна за счет того, что сразу вслед за загрузочным сектором на логическом диске находятся секторы, содержащие таблицу размещения файлов FAT (NTFS). Рассмотрим структуру и содержание данной служебной области диска.

Весь диск разбивается операционной системой на участки одинакового размера - кластера. Кластер может содержать несколько секторов. Для каждого кластера в таблице FAT есть своя индивидуальная ячейка, в которой хранится информация об использовании данного кластера. Все свободные кластеры отмечены нулями. Т.о., таблица размещения файлов - это массив, содержащий информацию о кластерах. Размер этого массива определяется общим количеством кластеров на логическом диске. Если файл занимает несколько кластеров, то эти кластеры связаны в список. При этом элементы таблицы FAT содержат номера следующих используемых данным файлом кластеров. Конец списка отмечен в таблице специальным значением. Номер первого кластера, распределенного файлу, хранится в элементе каталога, описывающего данный файл (рис.15).

Из рисунка видно, что для файла autoexec.bat отведено три кластера, а для файла config.sys - два кластера. В каталоге кроме и другой информации указаны номера первых кластеров, распределенных этим файлам (11 и 12). В своей одиннадцатой ячейке таблица FAT содержит число 17 - номер второго кластера, распределенного файлу autoexec.bat. Ячейка с номером 17 содержит число 18. Это номер третьего кластера, принадлежащего файлу autoexec.bat. Последняя ячейка, которая соответствует последнему кластеру, распределенному этому файлу, содержит специальное значение - FFFF.

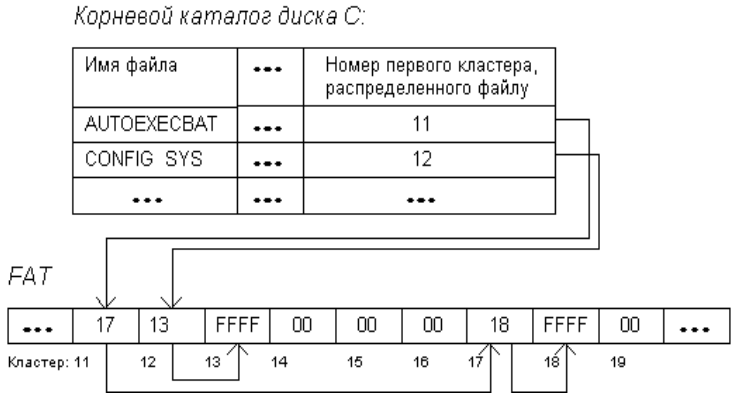


Рис. 15. Пример распределения кластеров для файлов autoexec.bat и config.sys

Таблица FAT может иметь 16 или 32-битовый формат. При этом в таблице для хранения информации об одном кластере диска используется, соответственно 16 и 32 бита. При этом, большой размер кластера приводит к неэффективному использованию дискового пространства. Это происходит из-за того что минимальный фрагмент дисковой памяти, выделяемый файлу, имеет достаточно большой размер (даже для файла размером 1 байт выделяется целый кластер). На диске может находиться несколько копий таблицы. Операционная система использует только первую копию, но обновляет вторую. Другие копии используются для утилит восстановления содержимого диска, таких как scandisk.exe. Количество копий FAT находится в поле fatcnt загрузочного сектора. Для чтения файла при помощи прерывания INT 25h необходимо установить соответствие между номерами кластеров и номерами секторов, в которых располагаются эти кластеры.

NTFS — стандартная файловая система для семейства операционных систем Microsoft с версии NT, заменила использовавшуюся файловую систему FAT. NTFS поддерживает систему

метаданных и использует специализированные структуры данных для хранения информации о файлах для улучшения производительности, надёжности и эффективности использования дискового пространства, имеет встроенные возможности разграничения доступа к данным для различных пользователей и групп пользователей, а также может назначать квоты (ограничения на максимальный объём дискового пространства, занимаемый теми или иными пользователями). NTFS использует систему журналирования для повышения надёжности файловой системы. Может поддерживать несколько корневых каталогов.

Сравнительная таблица файловых систем FAT32 и NTFS

Ограничения / возможности	*NTFS*	*FAT 32*
Размеры диска	2^{64} байт (16 экзабайт)	до 8 Тбайт
Размер тома	До 2 Тбайт	до 127 Гбайт.
Максимальный размер файла	≈ 16 Тбайт)	≤ 4 ГБ.
Максимальное количество файлов	4 294 967 295 (2^{32} — 1)	В FAT32 не более 268 435 444 (2^{28}−12)

Для флэш-накопителей традиционно используется FAT. exFAT — расширенная версия FAT, используемая для твердотельных флеш-дисков. Запатентована Microsoft как FAT64 — ограничение емкости носителя - 2^{64} байт (16 эксабайт).

UDF (Universal Disk Format, универсальный дисковый формат) — спецификация формата файловой системы, независимой от операционной системы для хранения файлов на оптических носителях. UDF позволяет дозаписывать файлы на CD-R или CD-RW дисках, один файл одновременно, без существенных потерь дискового пространства. Также UDF учитывает возможность выборочного стирания некоторых файлов на перезаписываемых носителях CD-RW, освобождая место на диске. Метаданные файловой системы, такие, как корневая директория, могут находиться где угодно на диске. UDF также используется для DVD и Blu-ray, так как имеет поддержку для дисков большого объёма — нет ограничения в 2 и 4 ГБ на размер файла.

Форматирование накопителей

Процесс форматирования заключается в создании (формировании) структур доступа к данным, например, структур файловой системы. При этом возможность прямого доступа к находящейся на носителе информации теряется, часть ее безвозвратно уничтожается. Некоторые программные утилиты дают возможность восстановить некоторую часть (обычно — большую) информации с отформатированных носителей. В процессе форматирования также может проверяться и исправляться целостность носителя. Существуют разные способы форматирования диска :

- ***Низкоуровневое форматирование***. Это базовая разметка области хранения данных, которая выполняется на заводе-изготовителе в качестве одной из заключительных операций изготовления устройства хранения данных. При этом процессе в области хранения данных создаются физические структуры: трэки (дорожки), секторы, сервометки — служебная информация, которая используется для позиционирования головок диска, при необходимости записывается программная управляющая информация. Впоследствии в подавляющем большинстве случаев эта разметка остаётся неизменной за все время существования носителя.
- ***Высокоуровневое форматирование.*** Этот процесс записывает (формирует) логические структуры, ответственные за правильное хранение файлов (файловые таблицы), а также, в некоторых случаях, загрузочные файлы для разделов, имеющих статус активных. Это форматирование можно разделить на два вида ***быстрое и полное:***
    - *Быстрое форматирование* — это способ форматирования, при котором новая таблица файлов создается без полной перезаписи или стирания информации жесткого диска. Быстрое форматирование занимает значительно меньше времени, чем обычное форматирование, при котором происходит полное стирание всей информации жесткого диска.

- *Полное форматирование* — это способ, при котором сначала производится верификация (проверка) физической поверхности носителя, при необходимости исправляются поврежденные сектора, т.е. участки поверхности, имеющие физические повреждения (маркируются как неисправные, что исключает в последующем запись в них информации), а уже потом производится запись таблицы файловой системы.

Форматирование носителей можно выполнить с помощью стандартных средств ОС Windows:

1. Вызвать контекстное меню для работы с носителем в окне «Мой компьютер»
2. Выбрать п. «Форматировать» (рис.16)
3. Выбрать требуемые параметры форматирования, кн. «Начать».

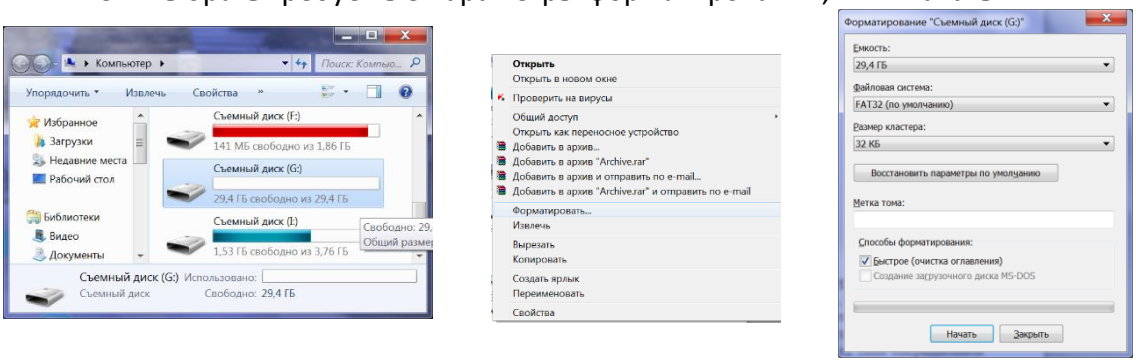


Рис.16

Определение метки тома

Метка тома – это имя носителя или его раздела, своего рода псевдоним, который присваивается пользователем физическому или виртуальному тому на каком-либо носителе компьютера. Операционной системе и прикладным программам для работы метка не нужна - они используют присвоенную диску литеру. А вот человеку удобнее работать с псевдонимом диска (меткой). Например, метки system и games не дадут перепутать, на котором из дисков хранятся игры, а на котором находятся файлы ОС, в отличие от литерных обозначений D и E.

Максимальная длина метки тома для файловой системы NTFS составляет 32 символа. Для файловой системы FAT метка тома может иметь длину до 11 символов. Метка тома может содержать пробелы, но не может содержать символы табуляции. На FAT метка тома не может содержать следующие символы (это ограничение не относится к NTFS): * ? / \ | . , ; : + = [ ] < > "

На FAT метка тома сохраняется в виде букв в верхнем регистре. В NTFS метка сохраняется так, как её ввёл пользователь. Если метка тома введена с нарушением вышеописанных ограничений, система может выдать сообщение об ошибке «указана недопустимая метка тома».

Метку можно установить:

- при форматировании накопителя (рис.16);
- при помощи «Проводника» Windows: в окне «Мой компьютер» вызвать контекстное меню, выбрать п. «Переименовать». Для завершения операции нажать клавишу Enter (рис.17);
- при помощи команды label, если изменить метку диска нужно из командной строки.

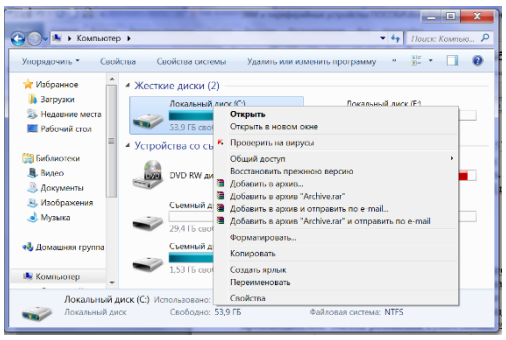


Рис.17.

Разделы диска

Разбиение диска на разделы - это процесс, который разбивает объём физического винчестера на логические диски (например, C:, D:…;sda1, sda2…; hda1, hda2…). Осуществляется с помощью встроенных служб самой операционной системы или соответствующими утилитами сторонних производителей. Метод разбиения существенно зависит от типа операционной системы. Разбиение диска на разделы не обязательно (в этом случае весь объем носителя будет состоять из одного раздела), но в виду очень больших объемов современных жестких дисков (до 4 Тб) их разбиение на логические разделы обычно осуществляется. Создавать разделы можно при форматировании дисков; с помощью специальных программ (например Fdisk); или служебными средствами Windows:

1. Вызвать контекстное меню значка «мой компьютер»;
2. Выбрать «Управление» («Управление компьютером»);
3. В левом окне выбрать «Управление дисками»;

Данная программа (рис. 18) в Windows 7 имеет больше возможностей, чем аналогичная в Windows XP, но меньше, чем специализированные программы (такие как Acronis Disk Director, например), но не нужно забывать, что это не специализированная программа, а встроенный в ОС функционал, не требующий установки программ или особенных знаний.

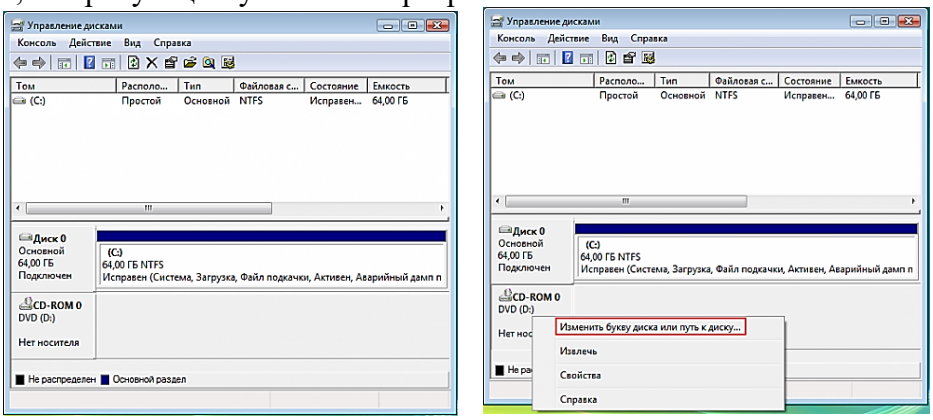
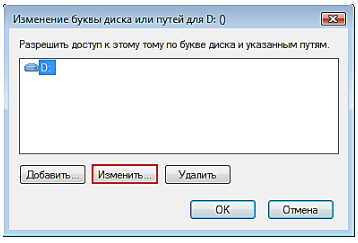


Рис.18.

Рассмотрим некоторые возможности программы:

1. Изменение стандартного имени диска

- В окне «Управление дисками» нажать правую клавишу мыши на требуемом диске, в контекстном меню выбрать пункт «**Изменить букву диска или путь к диску**» (рис.18)



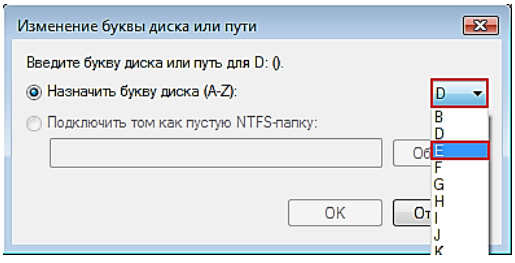


Рис.19. *Рис.20.*

- В открывшемся окне нажать кнопку **«Изменить» (рис.19);**
- Далее, выбрать из списка новую букву, которая будет назначена запоминающему устройству (рис.20);
- Нажать «**ОК», ответить «Да»** в окне с предупреждением системы

2. ***Создание нового раздела (выделение пространства из существующего)*** *на примере диска С.*
   - На диске нажать правую клавишу мыши, выбрать пункт «Сжать том» (рис.21), после чего на экране появиться сообщение (рис.22).
   - В следующем окне мастера управления разделами выводится информация об общем дисковом пространстве носителя, и доступном для выделения в другой раздел. Также можно указать количество мегабайт, на которые будет сжат текущий раздел (рис.23). нажать кнопку «Сжать»

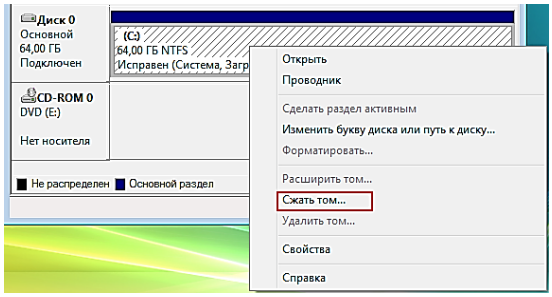


Рис.21.

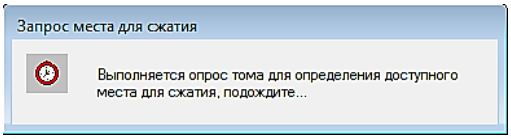


Рис.22.

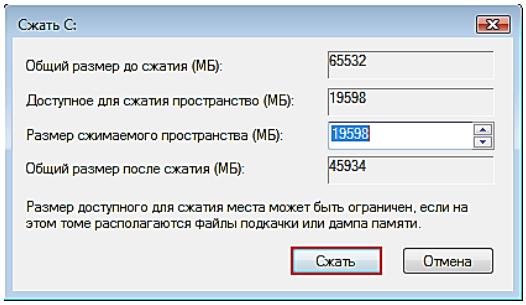


Рис.23.

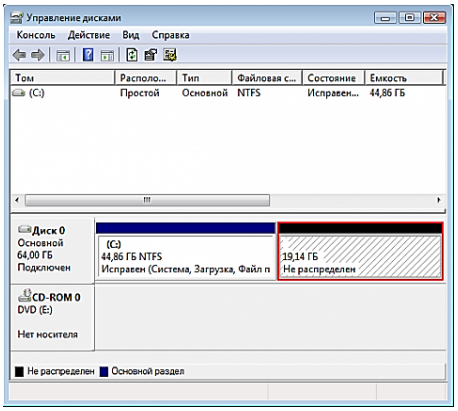


Рис.24

- После выполнения процедуры сжатия, которая займет некоторое время, в оснастке управления дисками появится новый, нераспределенный раздел (рис.24)
- Для создания нового тома и его форматирования надо нажать правую клавишу мыши на новом разделе и выбрать пункт «Создать простой том...» (рис.25).

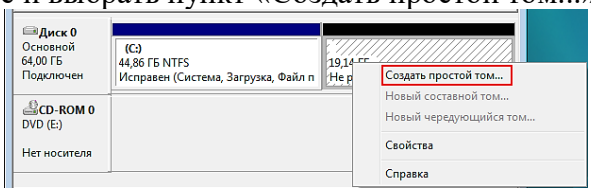


Рис.25.

- В окне мастера создания нового тома нажать «Далее», указать размер нового тома. По умолчанию предлагается использовать все доступное пространство в нераспределенном разделе, нажать «Далее» (рис.26).
- Указать имя диска (букву), нажать «Далее» (рис.27);

- В следующем окне выбрать требуется ли форматировать новый раздел. Если требуется, то указать параметры форматирования (рис.28).

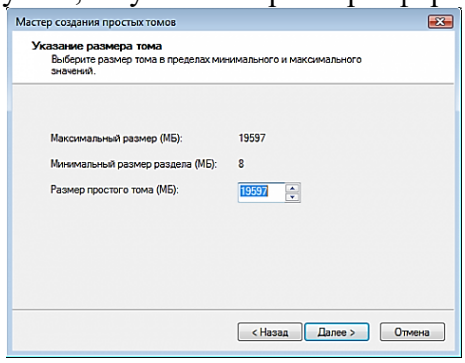


Рис.26.

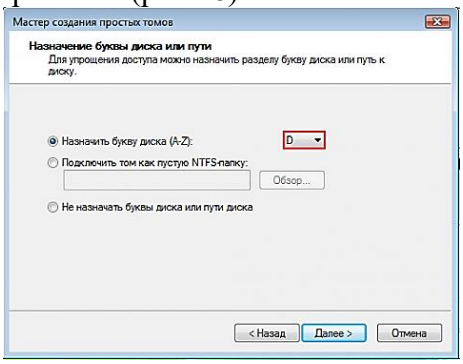


Рис.27.

- В окне «Завершение мастера создания простого тома» нажать кнопку «Готово» (рис.29).

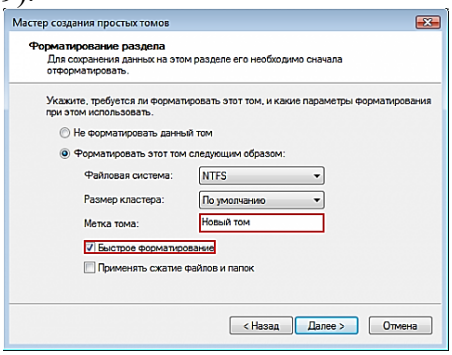


Рис.28.

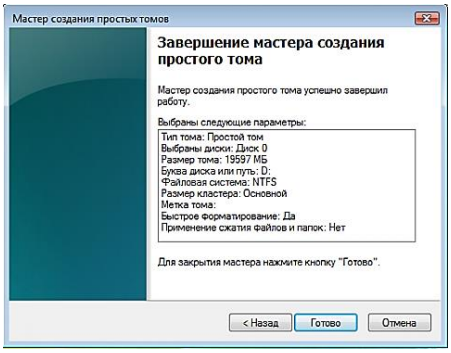


Рис.29.

Теперь вместо одного раздела на диске «С» есть два – «С» и «D» (рис.30).

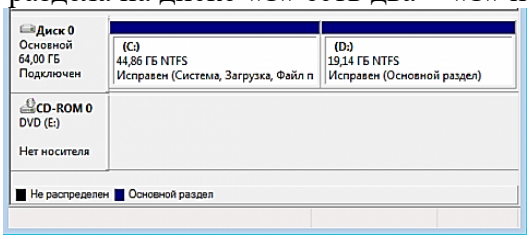


Рис.30.

3. Удаление раздела

Данная программа позволяет увеличить размер тома на диске за счет удаления другого раздела. Обязательным условием для этого является то, что раздел, следующий за первым должен быть основным, а не дополнительным, либо оба объединяемых тома должны быть расположены на одном дополнительном разделе. Данная процедура предусматривает «слияние» двух логических дисков, и без потери данных на последнем ничего не получится. Если требуется удалить том, то для этого:

- на удаляемом диске (разделе) нажать правой кнопкой мыши и выбрать пункт «Удалить том» (рис.31);

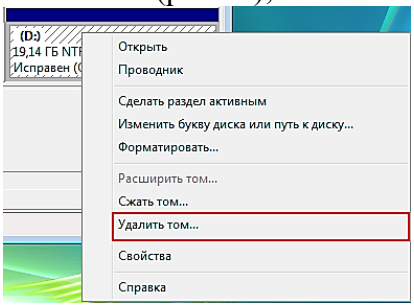


Рис. 31

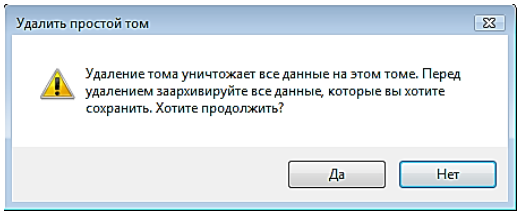


Рис.32.

- при этом на экран будет выведено предупреждение системы о том, что все данные на удаляемом разделе будут потеряны, нажать кнопку «Да» (рис.32);
- После этого том получит статус «Не распределен» (рис.33);

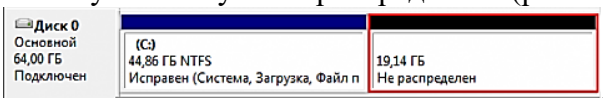


Рис. 33

- Далее, следует нажать правую клавишу мыши на диске и выбрать в контекстном меню пункт «Расширить том» (рис.34);

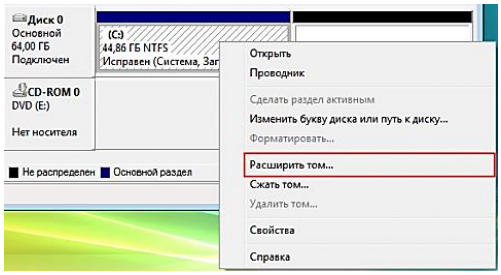


Рис. 34.

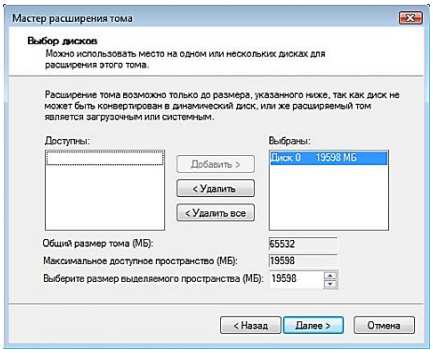


Рис.35

- Затем запустится мастер управления томом. В первом окне мастера нажать кнопку «Далее». В следующем окне указать или подтвердить перечень разделов, за счет которых будет расширен логический диск (рис.35).
- Закончить данную процедуру следует нажатием кнопки «Готово» (рис.36, 37).

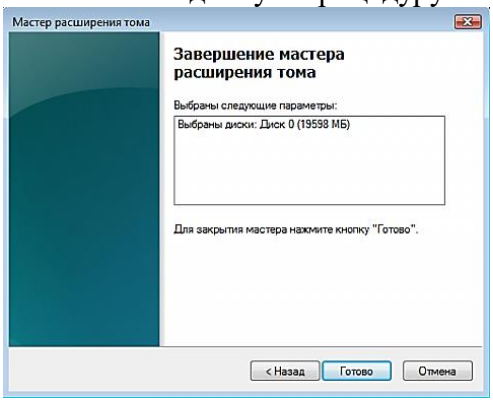


Рис. 36.

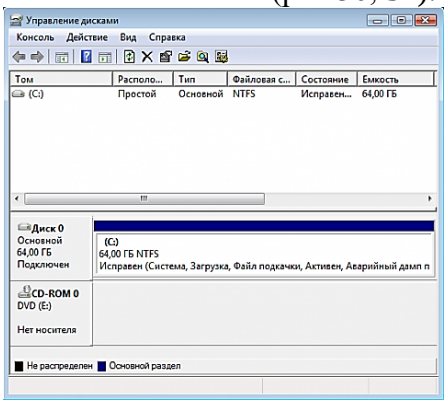


Рис.37

Таким образом, можно объединить разделы, и при этом в первом разделе все данные останутся нетронутыми.

Образ диска

Образ диска — файл, содержащий в себе полную копию содержания и структуры файловой системы и данных, находящихся на диске (компакт-диск, дискета, раздел жёсткого диска, флеш), причём неважно, был ли образ получен с реального физического диска или нет. Первоначально образы дисков использовались для резервного копирования и копирования дисков, при котором точное сохранение исходной структуры было необходимым и/или целесообразным. С появлением оптических носителей (CD, DVD) более часто встречающимся видом образов стали образы CD/DVD-диска, часто в форме .iso-файла, содержащего файловую систему ISO 9660. Формат ISO стал стандартом для образов дисков, но он не поддерживает много-сессионные данные. Помимо .iso существует ряд других форматов образа диска, таких как .img и .dmg, а также проприетарных: .vcd (Virtual CD) .nrg (Nero Burning ROM), .daa (Power ISO), .pqi (Drive Image), .vdf (Free VDF, VDF Crypt) и др.

Обычная программа резервного копирования сохраняет только файлы, к которым имеется доступ, а загрузчик и файлы, заблокированные операционной системой, могут быть не сохранены.

Образы дисков используются для распространения больших программных пакетов (например, дистрибутивов операционной системы GNU/Linux или BSD), для массовой установки программного обеспечения на компьютеры с одинаковой конфигурацией. Для этого на один компьютер устанавливаются все драйверы и необходимое программное обеспечение и снимается образ диска, который впоследствии устанавливают на оставшиеся компьютеры. Кроме того, существуют ситуации, когда создание образа может спасти компьютер, такие как: аппаратные сбои, Malware-угрозы (угроза от вредоносной программы), физические воздействия и пр.

Для получения образа используются специальные утилиты: CDBurnerXP,

Дефрагментация (оптимизация) диска

Дефрагментация — процесс обновления и оптимизации логической структуры раздела диска с целью обеспечения хранения файлов в непрерывной последовательности кластеров. После дефрагментации ускоряется чтение и запись файлов, а следовательно и работа программ, ввиду того, что последовательные операции чтения и записи выполняются быстрее случайных обращений (например, для жесткого диска при этом не требуется перемещение головки). Другое определение дефрагментации: перераспределение файлов на диске, при котором они располагаются в непрерывных областях.

Длинные файлы занимают несколько кластеров. Если запись производится на незаполненный диск, то кластеры, принадлежащие одному файлу, записываются подряд. Если диск переполнен, на нём может не быть цельной области, достаточной для размещения файла. Тем не менее, файл все-таки запишется, если на диске много мелких областей, суммарный размер которых достаточен для записи. В этом случае файл записывается в виде нескольких фрагментов.

На флеш-памяти время поиска не зависит от расположения секторов, и практически равно нулю, поэтому для них дефрагментация не требуется.

Дефрагментация чаще всего используется для таких файловых систем, как FAT для MS-DOS и Microsoft Windows, так как в программах для работы с ними обычно не предусмотрено никаких средств для предотвращения фрагментации, и она появляется даже на почти пустом диске и небольшой нагрузке.

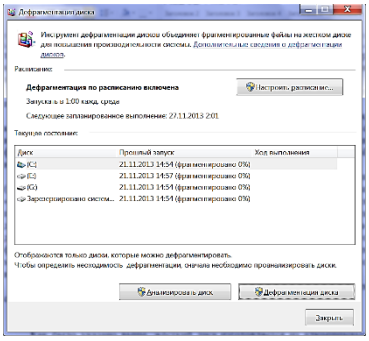


Рис.38

Помимо замедления компьютера в работе с файловыми операциями (таких как чтение и запись), фрагментация файлов негативно сказывается на «здоровье» жёсткого диска, так как заставляет постоянно перемещаться позиционирующие головки диска, которые осуществляют чтение и запись данных. Для устранения проблемы фрагментации существуют программы-дефрагментаторы, принцип работы которых заключается в «сборе» каждого файла из его фрагментов. Общим недостатком таких программ является их медленная работа — процесс дефрагментации обычно занимает очень много времени (до нескольких часов).

Дефрагментация диска может быть выполнена с помощью стандартных средств ОС Windiws (рис.38), Magical Defrag и др.

Проверка надежности дисков

Как было указано выше, жёсткий диск изготовляется из алюминиевых или стеклянных пластин, покрытых слоем ферромагнитного материала. Жёсткий диск это в первую очередь устройство работающее по принципу магнитной записи. Магнитные головки, считывающие,

записывающие или стирающие информацию с диска, парят над его поверхностью на высоте 10-12 нм и никогда не касаются поверхности магнитного диска, который легко повредить. Последним этапом изготовления запоминающего устройства является низкоуровневое форматирование, которое производится только один раз на специальном заводском оборудовании – серворайтер. В процессе этого форматирования в секторы записывается служебная информация (сервоинформация servo-служба, к примеру физический адрес сектора и адресный маркер, определяющий начало сектора) о разметке, которая нужна для нормальной работы жёсткого диска, для безошибочного позиционирования головок на эти дорожки и сектора при записи/считывании информации записанных. Вся служебная информация о номерах дорожек и секторов хранится в таблице, расположенной в закрытой и недоступной для средств ОС и BIOS служебной зоне. Так же в данной служебной зоне хранится паспорт диска, значения атрибутов SMART, а так же таблица-дефектов с информацией о невосстановимых или переназначенных сбойных секторах (бэд-блоках).

Если операционная система, испытывает проблемы с чтением данных с какого-либо сектора, то контроллер винчестера предпринимает ещё несколько дополнительных попыток прочитать данные, если они так же неудачны, данный сектор признаётся сбойным, в дальнейшем эта информация перезаписывается в нормальный сектор, находящийся на резервной дорожке, а проблемный сектор признаётся сбойным и выводится из обращения (Remapping). Факт признания сектора сбойным заносится в таблицу-дефектов с информацией о невосстановимых или переназначенных сбойных секторах.

Таблиц дефектов бывает две, одна начальная P-list (Primary-list), создаётся после конечных заводских испытаний, любой жёсткий диск уже при выходе с завода имеет несколько переназначенных бэд-блоков. А растущая таблица дефектов G-list (Grown-list), заполняется по мере использования жёсткого диска.

Бэд-блоки бывают нескольких видов:

– ***физические***, являются механическими дефектами магнитного покрытия поверхности жёсткого диска (осыпающийся магнитный слой рабочих пластин, сколы и т.д.). Т.е. сама структура сектора физически является неисправной, несомненно такой бэд-блок подлежит переназначению нормальным сектором с резервной дорожки. Очень часто это происходит из-за удара (например падение жёсткого диска на пол) или перегрева. Так же опасна вибрация жёсткого диска, если он ненадёжно закреплён. Пыльное помещение, курение, не смотря на установленный в жёстком диске фильтр, тоже играют огромную роль в образовании бэд-блоков, табачные смолы и пыль прилипают к поверхности жёсткого диска и мешают считыванию информации. Физические бэд-блоки невозможно исправить, можно только переназначить запасными секторами с резервных дорожек, что может сказаться на быстродействии запоминающего устройства.

– ***логические*** (ошибки логики сектора), в свою очередь делятся на *исправимые* и *неисправимые*. К неисправимым логическим ошибкам относятся ошибки сервоинформации, полученные при ударе, вибрации, люфте подшипников и т.д. Восстановить её можно только в заводских условиях. Исправимые логические бэд-блоки, возникают в том случае, если контрольная сумма сектора ECC (Error Correction Code - код коррекции ошибок, записываемая при записи в сектор пользовательской информации, данный код позволяет восстанавливать данные, если они были прочитаны с ошибкой) не совпадает с суммой пользовательских данных. Такое может произойти при внезапном отключение компьютера, из-за сбоев с электричеством (в этом случае информация в сектор жёсткого диска может быть записана, а контрольная сумма нет).

- ***программные*** бэд-блоки – это ошибки файловой системы (например неправильно помеченный сектор, принадлежащий двум файлам) можно убрать средствами операционной системы и форматированием.

Проверка надежности диска может быть выполнена с помощью свободного ПО: MHDD, HDDScan, Ashampoo HDD Control 2, Disk Director 11 Home и др.

HDDScan (рис. 39) — это бесплатная программа для низкоуровневой диагностики накопителей HDD в операционной системе Windows. Программа поддерживает диски IDE/SATA/SCSI, RAID массивы, внешние накопители USB/Firewire, флеш-карты. В программе реализован механизм проверки дисков и отправки отчётов по e-mail по расписанию. Также программа умеет сканировать поверхность, строить график скорости чтения, просматривать атрибуты SMART, настраивать AAM, APM (Power Management), etc. Более подробную информацию можно найти на сайте: http://ogoom.com/soft/portable/19386-hddscan-33-portable.html

Рис.39.

Восстановление удаленных файлов

Восстановление данных — процедура получения доступа к ранее удаленным данным на жестком диске или съемном носителе. К сожалению, такая возможность имеется не всегда. Это связано с принципом хранения информации на современных носителях на основе использования файловых таблиц. Когда происходит удаление файла, в том числе и из Корзины, то физически с ним ничего не происходит – удаляется только запись о нем в файловой таблице, а сам файл помечается как область для перезаписи. Именно благодаря этому можно попытаться восстановить нужные данные. Однако результат может быть отрицательным, если поверх нужного файла была записана какая-либо информация. На жёстких дисках всегда что-то стирается и что-то записывается, какие-то временные файлы, какие-то фоновые интернет-закачки и т.д. И, поскольку обычно файлы на диске хранятся небольшими фрагментами, то чем меньше свободного места на диске и чем больше пройдёт времени с момента удаления, тем больше вероятность, что нужные фрагменты будут затёрты новой информацией, а потеря любого фрагмента файла как правило равносильна потере всего файла. Поэтому у маленьких файлов гораздо больше шансов остаться целыми, чем у больших. Т.о. эффективность данной операции зависит от того записывалась ли на данный носитель новая информация или нет.

На сегодняшний день для этой цели существуют десятки платных и бесплатных приложений. У всех есть, как свои плюсы, так и недостатки.

R.saver - универсальная программа для восстановления удаленных файлов которая имеет несколько хороших алгоритмов поиска данных. Благодаря этому, приложение можно использовать как для поиска случайно удаленной информации, так и для восстановления файловой системы отформатированного диска. Она позволяет одинаково эффективно восстанавливать данные практически с любых файловых систем (NTFS, FAT, Ext (Linux), HFS (Mac OS), UDF (CD/DVD-диски) и др.). Причем, в отличие от других аналогичных приложений R.saver может возвращать не только отдельные файлы, но и целые папки, то есть полностью реконструировать файловую структуру! При этом программа не требует установки, и довольно проста в использовании (рис.40). Более подробную информацию можно найти на сайте: http://www.bestfree.ru/soft/file/data-recovery.php

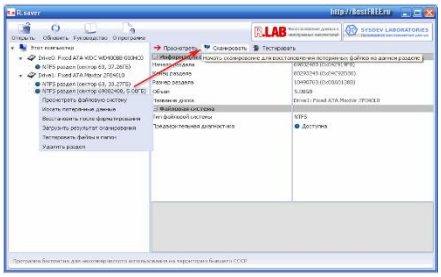


Рис.40.

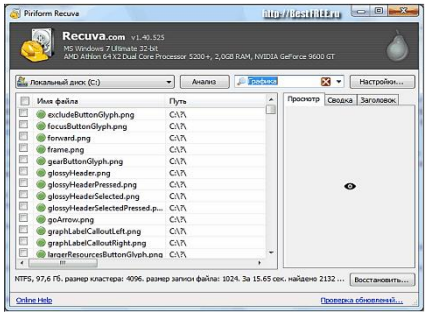


Рис.41.

Не плохими возможностями обладает программа ***Recuva***. Она сканирует весь винчестер или флешку и выдаёт список файлов, которые ещё можно восстановить. Обладает интуитивно понятным русскоязычным интерфейсом. Требует предварительной установки (рис.41). Более подробную информацию можно найти на сайте: http://www.bestfree.ru/soft/file/undelete.php

Антивирусная защита хранимой информации

Защита информации от вирусов на данный момент является едва ли не самой актуальной задачей в компьютерной индустрии. Ущерб от компьютерных вирусов может быть весьма значительным, поэтому лучшим решением является заблаговременная установка комплексной антивирусной защиты компьютера.

Компьютерный вирус — разновидность компьютерных программ, отличительной особенностью которой является способность к размножению (саморепликация). В дополнение к этому вирусы выполняют прочие действия, в том числе наносящие вред пользователю и/или компьютеру. По этой причине вирусы относят к вредоносным программам. Процесс внедрения вирусом своей копии в другую программу (системную область диска и т.д.) называется заражением, а программа или иной объект, содержащий вирус – зараженным. Сам вирус невелик, его размер редко измеряется килобайтами, но проблемы может создать большие от порчи конкретных типов файлов до полной потери информации на жестком диске и уничтожения содержимого BIOS материнской платы.

Рис.42.

Причины появления и распространения компьютерных вирусов обусловлены отсутствием соответствующих аппаратных средств защиты и противодействия со стороны операционной системы. Вирусы попадают в компьютер: вместе с программным обеспечением, которое запускает пользователь, что дает вирусу возможность размножаться и наносить вред; с использованием пиратских копий программ - это среда не только для распространения, но и «надежного» сохранения вирусов; с электронных досок объявлений (BBS), серверов FTP - это системы, которые позволяют обмениваться программным обеспечением; по электронной почте при открытии сообщений от неизвестных пользователей; при подключении запоминающих устройств: Flash -память, мобильный телефон и т.д. Классификация вирусов представлена на рис.42

Существует целый ряд программ, который обеспечивает комплексную антивирусную защиту: Антивирус Касперского, Doctor Web, Avast, Norton AntiVirus, McAfee VirusScan, Panda Antivirus и другие.

Брандмауэр

Брандмауэр - это программное или аппаратное обеспечение, которое блокирует атаки хакеров и не позволяет вирусам и вирусам-червям попадать на компьютер через Интернет. Брандмауэр Windows отслеживает весь сетевой трафик, передаваемый по соединениям, для которых он включен. Он следит за данными, отправляемыми с компьютера пользователя, и предотвращает попадание на него незапрошенных данных. Брандмауэр и антивирусная программа должны быть подключены до выхода в Интернет. Чтобы открыть брандмауэр Windows требуется:

1. Нажать кнопку «Пуск», «Панель управления».
2. В окне панели управления выбрать «Центр обеспечения безопасности», «Брандмауэр Windows» (рис.43).

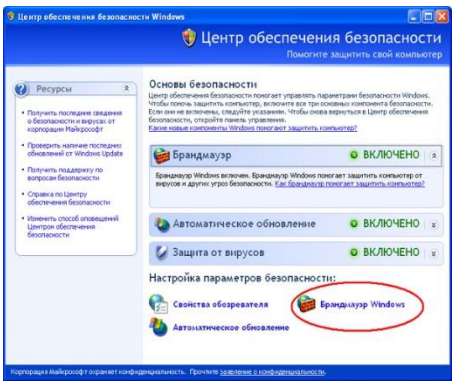


Рис.43.

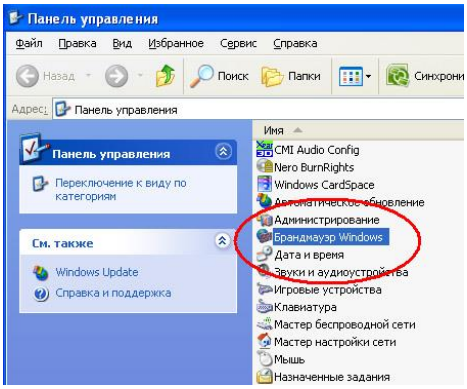


Рис.44

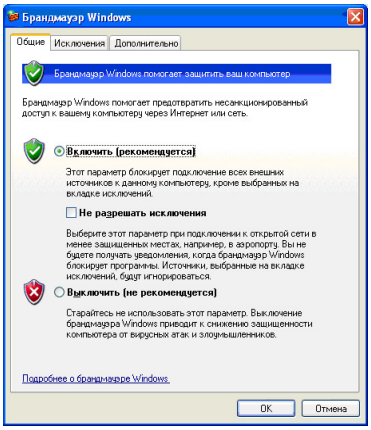


Рис.45

или

3. Нажать кнопку «Пуск», «Панель управления».
4. Выбрать Брандмауэр Windows (рис.44).

В открывшемся в любом случае окне (рис.45) можно проверить все настройки брандмауэра.

Бороться с вредоносными программами необходимо и домашним, и корпоративным пользователям, поэтому для них предназначены разные версии антивирусов.

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом и требованиями к отчету по лабораторной работе.
2. Получите у преподавателя носитель для учебных целей (флэш-накопитель), отформатируйте его с созданием FAT32 и меткой «УЧЕБНЫЙ»
3. Создайте на носителе второй раздел с половинным объемом.
4. Назначьте первому разделу имя «X», второму – «Y».
5. Изучите основы работы с программой создания образа. На диске «X» получите образ учебного CD.
6. Изучите основы работы с программой дефрагментации. Оптимизируйте учебную дискету.
7. Проверьте надежность дискеты и всех разделов флэш-накопитель.
8. На диск «Y» скопируйте с компьютера несколько папок и файлов различного типа.
9. Удалите с диска «Y» всю информацию.
10. Изучите основы работы с программой восстановления файлов. Восстановите содержимое диска «Y».
11. Проверьте, включен ли на компьютере брандмауэр. Зайдите на вкладку исключение. Определите, для каких программ создано исключение.
12. Выясните, какая антивирусная программа установлена на ПК. Каковы настройки этой программы. Когда последний раз обновлялись антивирусные базы. На какой срок действует лицензия.

13. Изучите основы работы с антивирусной программой. Проверьте все учебные носители на наличие вирусов и другого вредоносного ПО.
14. На флэш-накопителе удалите раздел «Y».
15. Почистите все носители от созданной вами информации.

Контрольные вопросы

1. Что такое кластер?
2. Что такое FAT, NTFS, UDF?
3. От чего происходит фрагментация диска?
4. Что такое форматирование диска и каковы его виды?
5. Что такое метка диска?
6. Что такое том?
7. Как происходит удаление данных с диска?
8. Что такое компьютерный вирус?
9. Как распространяются компьютерные вирусы?
10. Для чего используется брандмауэр?

Лабораторная работа 15. Работа с устройствами ввода/вывода. Обработка прерываний

Цель работы: Изучить суть процесса обработки прерываний, аппаратно-программную платформу реализации данных процессов. Познакомиться с особенностями взаимодействия процессора с периферийными устройствами. Закрепить знание мнемонической записи команд. Научиться составлять инструкции на языке Ассемблер для обработки прерываний и взаимодействия с внешними устройствами.

Основные теоретические сведения

Прерывание и его природа

Существует два типа взаимодействий между CPU и остальной аппаратной частью компьютера:

- передача команд аппаратным средствам,
- прием ответов от аппаратуры - прерывание.

Прерывание - инициируемый определенным образом процесс, временно переключающий микропроцессор на выполнение другой программы с последующим возобновлением выполнения прерванной программы.

Взаимодействие по принципу прерывания является наиболее тяжелым в обработке, т.к. прерывания возникают тогда, когда это удобно устройству, а не CPU. Аппаратные устройства обычно имеют весьма ограниченный объем ОЗУ, и если не считать поставляемую ими информацию немедленно, то она может потеряться.

Механизм прерываний позволяет обеспечить наиболее эффективное управление не только внешними устройствами, но и программами. Например, при нажатии клавиши на клавиатуре, инициируется посредством прерывания немедленный вызов программы, которая распознает нажатую клавишу, заносит ее код в буфер клавиатуры, откуда он в дальнейшем считывается программой или операционной системой. На время такой обработки микропроцессор прекращает выполнение выполняемой программы и переключается на так называемую процедуру обработки прерывания. После того как данная процедура выполнит необходимые действия, прерванная программа продолжит выполнение с точки, где было приостановлено ее выполнение. Некоторые операционные системы используют механизм прерываний не только для обслуживания внешних устройств, но и для представления своих «услуг». Так, операционная система MS-DOS взаимодействует с системными и прикладными программами преимущественно через систему прерываний.

Прерывания могут быть внешними и внутренними.

Внешние прерывания вызываются событиями внешними по отношению к микропроцессору. На рис.1 схематически изображена подсистема прерываний компьютера на базе микропроцессора Intel. У микропроцессора есть два физических контакта - ***INTR*** и ***NMI***,

предназначенные для формирования внешних по отношению к микропроцессору сигналов, возрастающие фронты которых извещают микропроцессор о том, что некоторое внешнее устройство требует внимания. Вход INTR (Interrupt Request) предназначен для фиксации запросов от периферийных устройств: системные часы, клавиатура, жесткий диск и т.д. Вход NMI (Non Maskeable Interrupt) - немаскируемое прерывание. Этот вход используется для событий, требующих безотлагательной обработки, или катастрофической ошибке. Внешние прерывания относятся, к не планируемым прерываниям.

Внутренние прерывания возникают внутри микропроцессора во время вычислительного процесса. К их появлению приводит:

- ненормальное внутреннее состояние процессора, возникшее при обработке некоторой команды программы. Такие события принято называть *исключительными ситуациями* (исключениями). Этот вид прерываний отчасти также можно отнести к не планируемым;
- обработка машинной команды *INT xx*. Этот тип прерываний называется *программным прерыванием*. Это - планируемые прерывания, так как с их помощью программист обращается в нужное время за обслуживанием своих запросов либо к операционной системе, либо к BIOS, либо к собственным программам обработки прерываний.

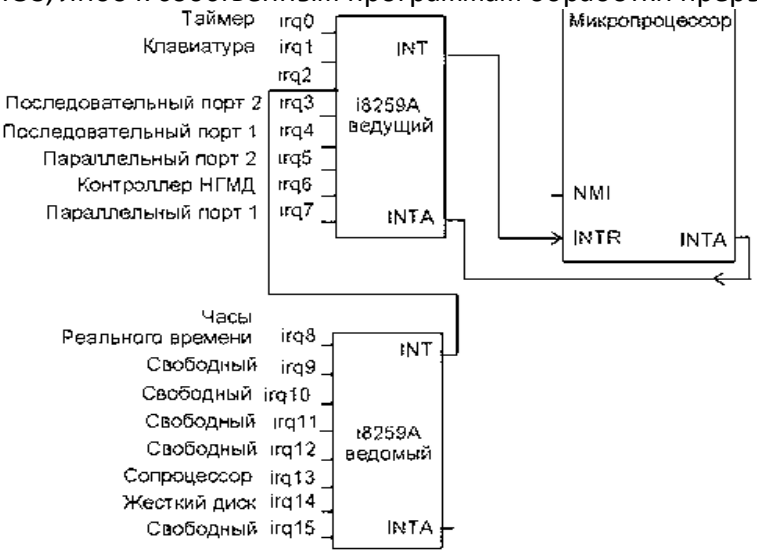


Рис.1.

Система прерываний

В общем случае ***система прерываний*** - это совокупность программных и аппаратных средств, реализующих механизма прерываний.

К *аппаратным* средствам системы прерываний относятся:

1. Выводы микропроцессора:
    - INTR - вывод для входного сигнала внешнего прерывания. На этот вход поступает выходной сигнал от микросхемы контролера прерываний 8259A;
    - INTA - вывод микропроцессора для выходного сигнала подтверждения получения сигнала прерывания микропроцессором. Этот выходной сигнал поступает на одноименный вход INTA микросхемы контроллера прерываний 8259A;
    - NMI - вывод микропроцессора для входного сигнала немаскируемого прерывания;
2. Микросхема программируемого контроллера прерываний 8259A, предназначенная для фиксирования сигналов прерываний от восьми различных внешних устройств;
3. Внешние устройства: таймер, клавиатура, магнитные диски и.д.

К *программным* средствам системы прерываний реального режима относятся:

1. Таблица векторов прерываний, в которой в определенном формате, в зависимости от режима работы микропроцессора, содержатся указатели на процедуры обработки соответствующих прерываний;
2. Флаги в регистре флагов:
   - IF (Interrupt Flag) – флаг прерывания. Предназначен для маскирования (запрещения) аппаратных прерываний, то есть прерываний по входу INTR. На обработку прерываний остальных типов флаг IF влияния не оказывает. Если IF = 1, микропроцессор обрабатывает внешние прерывания, если IF = 0, микропроцессор игнорирует сигналы на входе INTR;
   - TF (Trace Flag) - флаг трассировки. Значение флага TF = 1 переводит микропроцессор в режим покомандной работы, в котором после выполнения каждой машинной команды в микропроцессоре генерируется внутреннее прерывание с номером 1, и далее следуют действия в соответствии с алгоритмом обработки данного прерывания;
3. Машинные команды микропроцессора: INT, INTO, IRET, CLI, STI.

Контролер прерываний

Центральное место в схеме обработки аппаратных прерываний занимает программируемый контроллер прерываний (ПКП), выполненный в виде специальной микросхемы i8259A, которая может обрабатывать запросы от восьми источников внешних прерываний. В стандартной конфигурации вычислительной системы обычно используют две последовательно соединенные микросхемы i8259A, что позволяет увеличить количество возможных источников внешних прерываний до 15.

Основные функции контроллера прерываний:

- фиксирование запросов на обработку прерывания от источников, формирование единого запроса на прерывание и подача его на вход INTR микропроцессора;
- формирование номера вектора прерывания и выдача его на шину данных;
- организация приоритетной обработки прерываний;
- запрещение (маскирование) прерываний с определенными номерами.

Схематическое представление структуры и физических выводов микросхемы i8259A показано на рис.2.

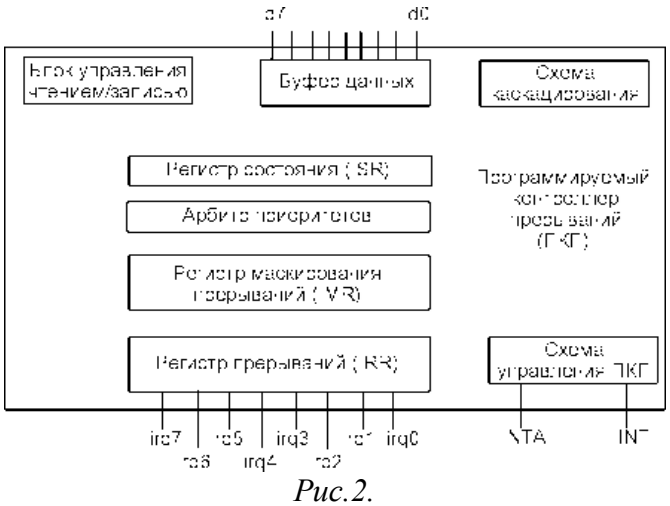


Рис.2.

Регистр запросов на прерывания IRR (Interrupt Request Register) - восьмиразрядный регистр, фиксирующий поступление сигнала на один из входов i8259A - irq0…irq7. фиксация выражается в установке соответствующего бита в единичное состояние;

Регистр маскирования прерываний IMR (Interrupt Mask Register) - восьмиразрядный регистр, с помощью которого можно запретить обработку запросов на прерывания, поступающих на соответствующие входы (уровни) irq0…irq7. Для запрещения (маскирования)

определенных уровней прерываний необходимо установить соответствующие биты регистра IMR. Эта операция осуществляется путем программирования порта 21h.

Регистр обслуживаемых прерываний ISR (Interrupt Service Register) - восьмиразрядный регистр, единичное состояние разрядов которого показывает, прерывания каких уровней обрабатываются в данный момент в микропроцессоре;

Арбитр приоритетов PR (Priority Resolver) - функцией данного блока является разрешение конфликта при одновременном поступлении запросов на входы irq0…irq7;

Блок управления - основной функцией данного блока является организация информационного обмена контроллера прерываний и микропроцессора через шину данных.

Принцип работы контролера

Рассмотрим прохождение и обработку сигнала прерывания от некоторого внешнего устройства.

Например, на вход irq0 поступает сигнал прерывания, что приводит к установке нулевого бита регистра IRR. Этот регистр связан с регистром маски IMR, состояние битов которого определяет, какие уровни прерываний запрещены (единичные биты) или разрешены к обработке (нулевые биты). Управление данным регистром осуществляется через порт 21h. Таким образом, если бит 0 в IMR равен нулю, то прерывание уровня 0 разрешено. Далее сигнал поступает к арбитру приоритетов, функция которого - разрешение конфликтов при одновременном поступлении запросов на несколько уровней. Обычно самый высокий приоритет у уровня irq_0, и далее уменьшается с возрастанием номера уровня. Если конфликта нет, то сигнал поступает на схему управления контроллером прерываний, которая формирует сигнал на выводе INT, который связан со входом микропроцессора INTR. Таким образом, сигнал на входе i8259A достиг микропроцессора.

Далее микропроцессор:

1. анализирует флаг IF. (IF = 1 - аппаратные прерывания разрешены, IF = 0 – запрещены);
2. если прерывания запрещены, то запрос на прерывание «повисает» до момента установки IF в единицу. Если прерывания разрешены, микропроцессор выполняет следующие действия:
   - сбрасывает флаг IF в ноль;
   - формирует сигнал подтверждения прерывания на выводе микропроцессора INTR. Этот вывод микропроцессора замкнут на одноименный вывод микросхемы i8259A.

Т.о., сигнал о прерывании прошел через микропроцессор и вернулся обратно в контроллер прерываний i8259A через вывод INTA. Данный вывод внутри контроллера прерываний замкнут на его схему управления, которая выполняет сразу несколько действий при поступлении этого сигнала:

1. сбрасывает бит в регистре IRR, соответствующий уровню прерывания irq_0;
2. устанавливает в 1 бит 0 регистра ISR, тем самым фиксируя факт обработки прерывания уровня 0 в микропроцессоре.
3. формирует с помощью блока управления номер вектора прерывания, используя буфер данных, и далее передает сигнал на выводы i8259A d0…d7. Выводы d0…d7 замкнуты на шину данных, по которой номер вектора поступает в микропроцессор. В микропроцессоре этот номер используется для вызова соответствующей процедуры обработки прерывания.

После того как номер прерывания по шине данных поступил в микропроцессор, последнему становится известно все об источнике прерывания. Если в это время придет другой сигнал о прерывании того же уровня, то он запоминается установкой бита в IRR, и обслуживание этого прерывания будет отложено. Если приходит прерывание другого уровня, то его дальнейшая обработка зависит от приоритета, который оно имеет по отношению к уже обрабатываемым прерываниям. Если приоритет выше, то текущая процедура обработки прерывания останавливается, и вызывается процедура обработки более приоритетного прерывания.

Обработка прерываний в реальном режиме

Обработка прерываний (как внешних, так и внутренних) в реальном режиме микропроцессора производится в три этапа:

1. Прекращение выполнения текущей программы.
2. Переход к выполнению и выполнение программы обработки прерываний.
3. Возврат управления прерванной программе.

Первый этап обеспечивает временное прекращение выполнения текущей программы таким образом, чтобы прерванная программа смогла продолжить свою работу. Любая выполняемая программа занимает отдельное место в оперативной памяти. Разделяемыми между программами ресурсами являются регистры микропроцессора, поэтому их содержимое требует сохранения. Обязательным для сохранения являются регистры CS, IP и Flags\eflags, и при возникновении прерывания они сохраняются автоматически. Пара CS:IP содержит адрес команды, с которой необходимо начать выполнение после возврата из программы обслуживания прерывания, а Flags\eflags - состояние флагов. Сохранение содержимого остальных регистров обеспечиваться на программном уровне. Для хранения значения регистров используется ***стек***. В конце первого этапа микропроцессор после занесения в стек регистров Flags, CS, IP сбрасывает бит флага прерываний IF в регистре Flags (но при этом в стек записывается предыдущее содержимое регистра Flags с еще установленным IF). Чем предотвращается возможность возникновения вложенных прерываний по входу INTR и порча регистров исходной программы вследствие неконтролируемых действий со стороны программы обработки вложенного прерывания. После того как необходимые действия по сохранению контекста завершены, обработчик аппаратного прерывания может разрешить вложенные прерывания командой STI.

Набор действий по реализации *второго этапа* заключается в определении источника прерывания и вызова соответствующей программы обработки. В реальном режиме микропроцессора допускается от 0 до 255 источников прерываний, что ограничено размером таблицы векторов прерываний. Эта таблица выступает связывающим звеном между источником прерывания и процедурой обработки. Таблица располагается в памяти, начиная с адреса 00000000h, каждый элемент таблицы занимает 4 байта и имеет следующую структуру: 1-е слово - значение смещения начала процедуры обработки прерывания (n) от начала кодового сегмента; 2-е слово - значение базового адреса сегмента, в котором находится процедура обработки прерывания. Определить адрес, по которому находится вектор прерывания с номером n, можно следующим образом:

смещение_элемента_таблицы_векторов_прерываний = n*4

Т.о., на втором этапе обработки прерывания микропроцессор выполняет следующие действия:

1. По номеру источника прерывания путем умножения на 4 определяет смещение в таблице векторов прерываний.
2. Помещает первые два байта по вычисленному адресу в регистр IP.
3. Помещает вторые два байта по вычисленному адресу в регистр CS.
4. Передает управление по адресу, определяемому парой CS:IP.

Далее выполняется сама программа обработки прерывания. Она, в свою очередь, также может быть прервана, например, поступлением запроса от более приоритетного источника. В этом случае этапы 1 и 2 будут повторены для вновь поступившего запроса.

Набор действий по реализации *третьего этапа* заключается в восстановлении контекста прерванной программы. Так же, как и на этапе 1, на данном этапе есть действия, выполняемые микропроцессором автоматически, и действия, задеваемые программой. Основная задача на этом этапе - привести стек в исходное состояние (очистить) и восстановить состояние регистров. Участок кода, предписывающий эти действия должен быть защищен от возможности искажения (например, в результате появления аппаратного прерывания) с помощью команды CLI. Т.о., последними командами в процедуре обработки прерывания должны быть STI и IRET, при обработке которых микропроцессор выполняет следующие действия:

1) STI - разрешить аппаратные прерывания по входу INTR;

2) IRET - извлечь последовательно три слова из стека и поместить их, соответственно, в регистры IP, CS, Flags.

В результате действий этого этапа управление возвращается очередной команде прерванной программы, которая должна была выполнится, если бы прерывания не было.

Аппаратные прерывания могут быть инициированы программно командой микропроцессора ***INT n***, где n - номер аппаратного прерывания в соответствие с таблицей векторов прерываний. При этом микропроцессор также сбрасывает флаг IF, но не вырабатывает сигнал INTA.

Команды обработки прерываний

Команды ассемблера для вызова прерываний и возврата в основную программу относятся к группе команд передачи управления.

Мнемоника команд передачи управления

Обозначение	Выполняемая функция
Команды безусловной передачи управления	
INT	Вызов программного прерывания
IRET	Возврат из прерывания

Команда INT

Мнемоника	Функция команды	Синтаксис	Примечание
INT	вызывает программное прерывание	INT n	*n* – непосредственный операнд со значением от 0 до 255
Использование регистра флагов	обнуляет флаг трассировки TF и флаг включения/ выключения прерываний IF для блокировки других действий, кроме обработки вызванного прерывания		

Команда выполняет косвенный межсегментный вызов подпрограммы-обработчика прерывания из 256 возможных векторов прерывания. Т.о., выполнение команды INT эквивалентно последовательному выполнению команд PUSHF, CLI (сбросить флаг IF) и CALL, если не обращать внимание на флаг TF.

Пример использования команды *INT*:

INT 3 - вызов прерывания с номером вектора 3

INT 21 - вызов прерывания с номером вектора 21*h*

Таблица значений ***n*** *(типов прерываний)*

n	Значение
0	возникает при делении на 0 или если частное от деления превышает разрядную сетку
1	действует в режиме “трассировки” (после выполнения каждой команды программы происходит останов)
2	немаскируемые технические прерывания
3	прерывания по команде INT, включенной в программу (вызывает останов и отображение содержимого регистров микропроцессора)
4	прерывание по команде INTO, включенной в программу (выполняется при условии, что при выполнении предыдущей команды произошло переполнение разрядной сетки)
8-15	аппаратные прерывания, инициируемые внешними устройствами;
16- 31	планируемые программные прерывания BIOS
32-255	программные прерывания DOS

В некоторых типах прерываний имеется много разновидностей (иногда более 10). Так, прерывание 33 (21H) имеет около 100 разновидностей (это прерывание наиболее часто используется в программах пользователя). В таких случаях вид прерывания (внутри типа) определяется содержимым регистра AH и называется ***функцией*** соответствующего прерывания.

Наиболее часто используемые функции прерывания 21H

Значение Ah	Операция	Дополнительные входные регистры	Выходные регистры	Примечание
Функции ввода с клавиатуры в память				

1h	Ожидание набора символа на клавиа-туре с последую-щим изображением его на экране (эхо).	Не используются	(AL) = ASCII-код символа	Символ отображается на экране
6	Чтение символа с клавиатуры	(DL)=0FFH	(AL) = символ	
Ah	Чтение клавиатурной строки в буфер (в память)	DX – начальный адрес буфера памяти, в который будет записана строка		Структура буфера: Первый байт-максимальная дли-на строки, включая Enter (указывает программист); Второй байт –фактическая длина (без Enter, заполня-ет система после ввода строки); С третьего байта – сама строка (рис.3)
Функции вывода из памяти на дисплей				
2h	Вывод символа на экран	(DL)= ASCII-код символа	не используются	
9	Изображение строки на экране дисплея	DX= начальный адрес строки, которая должна заканчиваться символом $	Не используются	
Функция завершения программы				
4Ch	Завершение программы, выход в операционную систему	**AL** – код завершения программы (=0)		

В общем виде вызов процедур 21h: ***mov ah, <номер функции>***
mov dl, <параметр>
int 21h

Следует иметь ввиду, что прерывание int 21h переопределяет содержимое регистра AL (левая часть AX), поэтому необходимо предусматривать сохранение нужных данных перед вызовом этого прерывания

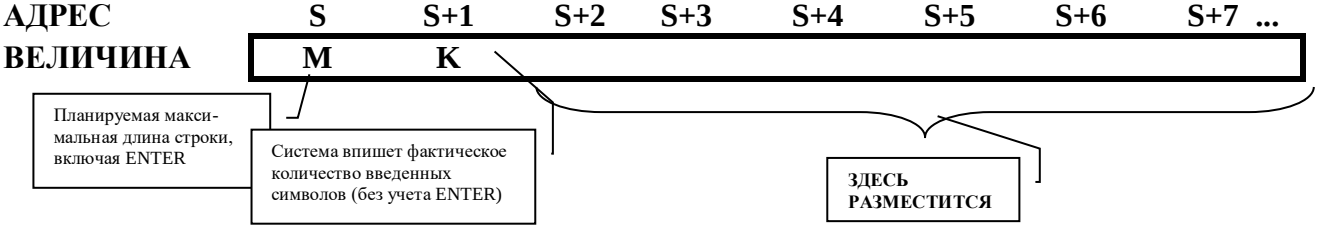


РИС.3 – СХЕМА БУФЕРА ДЛЯ ВВОДА СТРОКИ ФУНКЦИЕЙ AH

Команда IRET

Мнемоника	Функция команды	Синтаксис	Примечание
IRET	Осуществляет возврат из программы обработки прерывания	*IRET*	Не имеет операндов. Восстанавливает из стека значение регистров IP, CS и регистра флагов и передает управление основной программе.
Использование регистра флагов		Содержимое регистра флагов при выполнении команды *RET* не изменяется.	

Пример использования команды *IRET*:

```
IRET            ; возврат из прерывания
```

Команды отладчика для работы с программами, содержащими прерывания

Выполнение последовательности инструкций

Для выполнения последовательности инструкций в сегменте кода, начиная с той, на которую указывает IP, используется команда G.

Синтаксис команды: g <смещение останова >

Эту команду нужно использовать при выполнении прерывания, чтобы избежать его трассировки.

Сохранение программы в файле

Команда отладчика **N** позволяет создать файл с определенным именем.

Синтаксис команды n <имя.расширение>

Команда **W** позволяет записать число байт, указанное в CX, на диск в поименованный файл (в текущий каталог, там, где находится программа debug). BX при этом должен быть установлен в 0.

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом.
2. Используя прерывание 21, вывести на экран символ с кодом:

Вариант	*Код символа*	*Вариант*	*Код символа*
1	39	2	44
3	40	4	45
5	41	6	46
7	42	8	47
9	43	10	48

3. Используя прерывание 21, вывести на экран последовательность символов, используя РОН. Для определения кода ASCII соответствующего символа используйте таблицу в Приложении 1:

Вариант	*Код символа*	*Вариант*	*Код символа*
1	F3d9>(	2	S5|n:)
3	k8G+]	4	7x?Y!;
5	U0v:\”	6	2R}w*,
7	D8m.{-	8	G~b6-.
9	J4f<’#	10	3Am”%`

4. Вывести на экран текстовое сообщение:

Вариант	*Текст сообщения*
1	Микропроцессор
2	Компьютер
3	Аккумулятор
4	Микроконтроллер
5	Ассемблер
6	Информация
7	Преобразователь
8	Использование
9	Переключатель
10	Программатор

5. Записать на диск в файл str.com программу вывода на экран текстового сообщения, хранящегося в ОЗУ с адреса 200:

Вар-т	*Задание*
1	Ввести с клавиатуры два числа. Вывести на экран значение их произведения с комментариями
2	Ввести с клавиатуры два числа. Вывести на экран значение их суммы с комментариями
3	Ввести с клавиатуры два числа. Вывести на экран значение их разности с комментариями
4	Ввести с клавиатуры три числа. Вывести на экран значение их произведения с комментариями
5	Ввести с клавиатуры три числа. Вывести на экран значение их суммы с комментариями
6	Ввести с клавиатуры три числа. Вывести на экран значение их разности с комментариями
7	Ввести с клавиатуры два числа. Если они равны, вывести это сообщение на экран с комментариями
8	Ввести с клавиатуры два числа. Если первое больше, вывести это сообщение на экран с

	комментариями
9	Ввести с клавиатуры два числа. Если второе больше, вывести это сообщение на экран с комментариями
10	Ввести с клавиатуры два числа. Если они не равны, вывести это сообщение на экран с комментариями

Выполнить программу, сохранить данные в отчете.

Пример 1: Вывести на экран текст «Hello».

Для того, чтобы составить программу вывода строки, нужно воспользоваться прерыванием 09h. Эта функция выводит на экран последовательность символов до появления символа $ (24h). Смещение начала строки в сегменте данных должно быть указано в регистре DX.

```
-a
13D2:0100 mov ah, 09
13D2:0102 mov dx, 0108
13D2:0105 int 21
13D2:0107 iret
13D2:0108 db "Hello$"
-g
Hello
```

Пример 2: Сравнить числа X и Y. Выдать на экран соотвествующее сообщение. Написать программу в Debug и сохранить на носителе.

Для решения этого задания следует использовать две функции: вывод строки на экран и ввод данных с клавиатуры.

Чтобы обратиться к функции надо:

- поместить номер функции в регистр AH
- если есть подфункции, то номер подфункции поместить в AL
- загрузить остальные регистры согласно описанию функции
- подготовить необходимые буферы, строки ASCII и управляющие блоки
- вызвать прерывание INT 21H
- проверить индикатор ошибки, возвращенный прерыванием (флаг переноса)

- Функция ввода с клавиатуры: вход - AH = 02h; выход - AL = код символа, полученный из стандартного ввода. Данная функция считывает (ожидает) символ со стандартного входного устройства. Отображает этот символ на стандартное выходное устройство (эхо).
- Функция вывода строки на дисплей: вход - AH = 09h; DX = адрес строки, заканчивающейся символом ‘$’. Чтобы перейти на новую строку, следует включить в текст пару CR/LF (ASCII 0Dh и 0Ah).
- Завершение программы – прерывание INT 20h

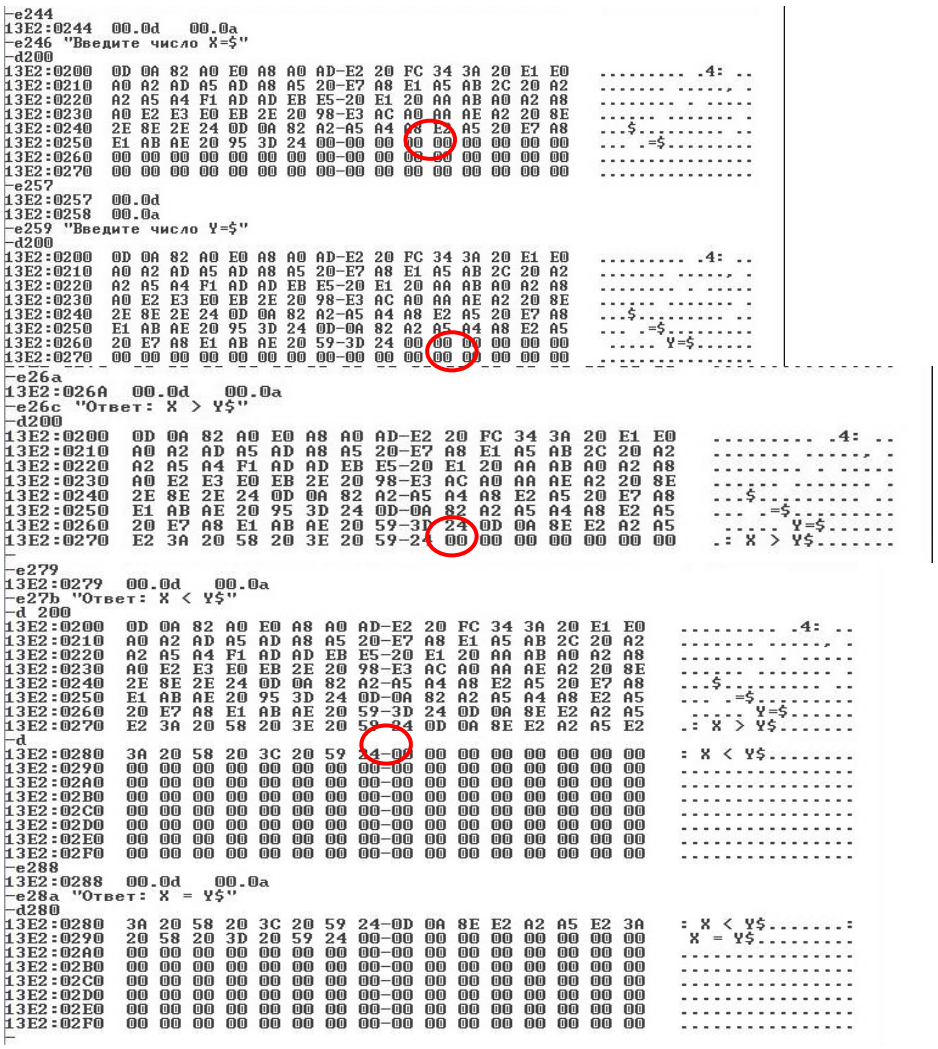
Решение:

1. Подготовка данных (строк) для вывода на экран

Разместим в ОЗУ строки диалога с пользователем для наглядности программы. Для перехода на новую строку введем перед каждой строкой диалога по два байта 0d и 0a (см. выше). Запускаем Debug и записываем в память строки. Запоминаем адреса начала строк, для последующего обращения к ним из программы:

```
-e200
13E2:0200  00.0d   00.0a
-e202 "Вариант №4: сравнение чисел, введённых с клавиатуры. Шумаков О.О.$"
-d200
13E2:0200  0D 0A 82 A0 E0 A8 A0 AD-E2 20 FC 34 3A 20 E1 E0   ......... .4: ..
13E2:0210  A0 A2 AD A5 AD A8 A5 20-E7 A8 E1 A5 AB 2C 20 A2   ....... .....,  .
13E2:0220  A2 A5 A4 F1 AD AD EB E5-20 E1 20 AA AB A0 A2 A8   ........ . .....
13E2:0230  A0 E2 E3 E0 EB 2E 20 98-E3 AC A0 AA AE A2 20 8E   ...... ....... .
13E2:0240  2E 8E 2E 24 00 00 00 00-00 00 00 00 00 00 00 00   ...$............
13E2:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13E2:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
13E2:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

Последний байт этой строки символ $, код которого 24h, расположился по адресу 13E2:0243. Ввод следующих строк начинается с адреса 13E2:0244. После ввода каждой строки просматриваем дамп памяти для определения адреса ввода последующей строки.



2. *Листинг программы*

mov ah, 09	подготавливаем функцию вывода строки на экран
mov dx, 200	в регистр dx вносим адрес выводимой строки "Вариант №4……"
int 21	вызываем прерывание для выполнения функции вывода строки на экран
mov dx, 244	в регистр dx вносим адрес строки «Введите число X=»
int 21	вызываем прерывание для выполнения функции вывода строки на экран
mov ah, 01	подготавливаем функцию ввода символа с клавиатуры
int 21	вызываем прерывание для выполнения функции
mov cx, ax	сохраним значение регистра ax (al=код введённого числа X) в регистре cx для последующего использования (сравнения)
mov ah, 09	готовим функцию вывода на экран
mov dx, 257	адрес строки "Введите число Y ="
int 21	вызываем прерывание для выполнения функции вывода строки на экран
mov ah, 01	готовим функцию ввода символа
int 21	вызываем прерывание: программа ожидает ввода символа с клавиатуры
mov bx, ax	копируем значение регистра ax (al = код числа Y) в регистр bx
mov ah, 09	установим заранее функцию вывода на экран
cmp cx, bx	сравниваем значения регистров cx и bx, в которые мы сохранили числа X и Y
JG ds:133	если число X (т.е. его 16–ричный код) больше числа Y (...), то переходим к адресу 133, где запрограммирован вывод соответствующей строки на экран
JL ds:13A	если число X (т.е. его 16–ричный код) меньше числа Y (...), то переходим к адресу 13A, где запрограммирован вывод соответствующей строки на экран

mov dx, 288 в dx вносим адрес строки о равенстве чисел X и Y (переходы jg и jl не совершились)
int 21 прерывание вывод строки на экран
int 20 завершение программы
mov dx, 26a в dx вносим адрес строки "X > Y" (сюда перейдёт по команде jg)
int 21 прерывание DOS, вывод строки на экран
int 20 завершение программы
mov dx, 279 в dx вносим адрес строки "X < Y" (сюда перейдёт по команде jl)
int 21 прерывание DOS, вывод строки на экран
int 20 завершение программы

```
13E2:0100 B80009        MOV     AX,0900
13E2:0103 BA0002        MOV     DX,0200
13E2:0106 CD21          INT     21
13E2:0108 BA4402        MOV     DX,0244
13E2:010B CD21          INT     21
13E2:010D B80001        MOV     AX,0100
13E2:0110 CD21          INT     21
13E2:0112 89C1          MOV     CX,AX
13E2:0114 B80009        MOV     AX,0900
13E2:0117 BA5702        MOV     DX,0257
13E2:011A CD21          INT     21
13E2:011C B80001        MOV     AX,0100
13E2:011F CD21          INT     21
13E2:0121 89C3          MOV     BX,AX
13E2:0123 B80009        MOV     AX,0900
13E2:0126 39D9          CMP     CX,BX
13E2:0128 7F09          JG      0133
13E2:012A 7C0E          JL      013A
13E2:012C BA8802        MOV     DX,0288
13E2:012F CD21          INT     21
13E2:0131 CD20          INT     20
13E2:0133 BA6A02        MOV     DX,026A
13E2:0136 CD21          INT     21
13E2:0138 CD20          INT     20
13E2:013A BA7902        MOV     DX,0279
13E2:013D CD21          INT     21
13E2:013F CD20          INT     20
```

3. Запись программы на диск

Для записи программы на диск необходимо знать её размер в байтах, и это значение необходимо внести в регистр CX. Последний байт внесённый до этого нами в область данных имеет адрес 296 (см. выше). Это значение и будет размером программы. Вводим это число в регистрCX:

```
-r cx
CX 0000
:296
```

Далее необходимо командой **n** ввести имя программы (придумать) и командой **w** произвести запись программы в файл:

```
-n schumak
-w
Запись 00296 байт
-_
```

4. Запуск и исполнение программы

Чтобы загрузить эту программу в Debug надо ввести её имя командой **n**, и загрузить командой L (файл программы должен находиться в каталоге программы debug):

```
-n schumak
-l
```

Для запуска программы вводим команду **g**:

```
-g

Вариант №4: сравнение чисел, введённых с клавиатуры. Шумаков О.О.
Введите число X=8
Введите число Y=8
Ответ: X = Y
Программа завершилась нормально
-g

Вариант №4: сравнение чисел, введённых с клавиатуры. Шумаков О.О.
Введите число X=3
Введите число Y=0
Ответ: X > Y
Программа завершилась нормально
-g

Вариант №4: сравнение чисел, введённых с клавиатуры. Шумаков О.О.
Введите число X=8
Введите число Y=9
Ответ: X < Y
Программа завершилась нормально
-
```

Примечание: в данной программе сравниваются числа от 0 до 9. Анализ правильности ввода (что введено именно число, а не буква) не производится. Данную проверку возможно было бы производить по анализу кодов введённых символов. Если коды введённых символов выходят за рамки диапазона (30h – 39h), то программа возвращается к вводу числа X или Y.

Контрольные вопросы

1. Что такое прерывание?
2. Опишите программно-технический комплекс, реализующий процесс обработки прерывания
3. Каков механизм обработки прерывания?
4. Какую информацию содержит таблица векторов прерывания?
5. Какие типы прерываний бывают?
6. Как осуществляется ввод информации с клавиатуры?
7. Как осуществляется вывод информации на экран дисплея?

Лабораторная работа 14. Основные устройства ввода-вывода информации

Цель работы: Познакомиться с назначением, принципами работы и основными характеристиками устройств ввода-вывода информации (монитор, клавиатура, мышь, принтер). Изучить виды и организацию устройств. Получить навык подбора конфигурации (спецификации, комплектации) ПК под решаемые задачи.

Основные теоретические сведения

Мониторы

Монитор - это устройство, предназначенное для визуального отображения текстовой и графической информации. По своему устройству современные мониторы бывают электронно--лучевые (рис.1), жидкокристаллические (рис.2), плазменные (рис.3), светодиодные (рис.4). Работой монитора руководит устройство - контоллер (видеокарта), который устанавливается на материнскую плату (стандартный SVGA, встроенный в северный мост, при этом материнская плата имеет VGA разъем) (рис.5), интегрирован в центральный процессор, или вставляется в специализированный слот (PCI Express) (рис.6). Работа монитора полностью зависит от возможностей и характеристик видеокарты.

Рис.1.

Рис.2.

Рис.3.

Рис.4.

Рис.5 *Рис.6*

Устройство мониторов

1. Устройство ЭЛТ(CRT) - монитора.

Принцип получения изображения на экране обычного электронно-лучевого монитора состоит в том, что луч, выпущенный из электронно-лучевой пушки, направляется отклоняющей системой в нужную точку экрана (которым является передняя стенка трубки), проходит через металлический трафарет (маску), и бьет в заданный участок экрана, фосфорное покрытие которого (люминофор) активизируется и излучает свет (рис.7).

Экран монитора представляет собой матрицу из крохотных трехцветных ячеек. Цвет каждой ячейки образуется смешением красного, зеленого и синего цветов различной интенсивности. При равной интенсивности всех трех цветов получается белый. Чтобы луч точно попал «в цвет», перед слоем люминофора ставится трафарет (маска), которая сужает пучок и сосредотачивает его на одном из трех участков люминофора. Разные фирмы - производители используют разные типы масок. Самой заслуженной является *точечная (теневая)* (рис.8). Мониторы с такой маской очень хорошо подходят для работы с текстовой информацией (контуры букв получаются очень гладкими), они дают натуральные и точные цвета, но не обеспечивают высокой яркости и контрастности.

Рис.7.

Для работы с графикой более подходят мониторы с *апертурной* решеткой (рис.9). Люминофор в таких трубках наносится в виде вертикальных полос, а маска представляет собой проволочную сетку. Из-за относительно редкой сетки изображение получается очень ярким, поэтому на мониторах применяют темное экранное стекло. Эти мониторы предпочтительны для редактирования изображений в графических пакетах прикладных программ.

Симбиозом двух перечисленных является *гнездовая (щелевая)* маска (рис.10). Ее отверстия представляют собой набор вертикальных отрезков (больше напоминающих эллипс), расположенных в шахматном порядке. В результате изображение на экране получается четким с натуральными, яркими контрастными цветами.

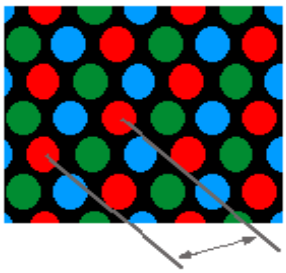


Рис. 8.

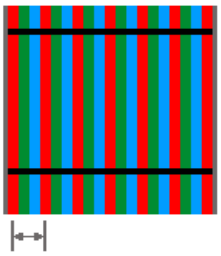


Рис.9.

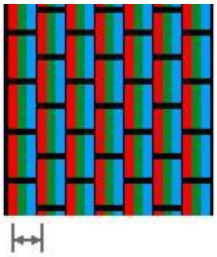


Рис.10.

Принцип действия CRT-монитора:

1) Монитор получает сигнал от компьютера и передает его на электронно-лучевую пушку, которая формирует луч, передающий совокупность сигналов: красный, зеленый, синий (RGB) на переднюю панель трубки. Цветное изображение формируется за счет наложения трех основных цветов, зависит от типа люминофора (каждому цвету соответствует свой тип люминофора) и от используемой цветоделительной маски. Именно цветоделительная маска отслеживает попадание того или иного цветного луча на соответствующий тип люминофора.

2) Луч направляется отклоняющей системой, проходит через цветоделительную маску, которая направляет его на люминофорное покрытие. В результате чего создается свечение, видимое глазу. Любое изображение на экране CRT-монитора состоит из множества дискретных точек люминофора, называемых пикселями. Такие мониторы называются растровыми. Растр формируется так, как показано на рис. 11. Электронный луч периодически сканирует весь экран, образуя на нем близко расположенные строки развертки. По мере движения луча по строкам видеосигнал изменяет яркость светового пятна и образует видимое на экране изображение. В цикле сканирования поверхности экрана, луч движется по зигзагообразной траектории от левого верхнего угла до правого нижнего (Прямой ход луча по горизонтали осуществляется сигналом строчной (горизонтальной) развертки, а по вертикали – кадровой (вертикальной) развертки). Перевод луча с права налево осуществляется с помощью сигналов обратного хода. Минимальное время прохождения луча по экрану от начальной точки к возврату в нее составляет от 20 мс до 1 секунды

При низком качестве люминофорного покрытия внутренней стороны экрана, появится мерцание изображения, так как такие частицы люминофора имеют короткий период послесвечения и гаснут еще до того, как луч снова их высветит. При использовании высококачественных и дорогих материалов такой эффект не наблюдается. Каждая точка светится ровно столько, сколько необходимо лучу для сканирования всего экрана.

Недостатки мониторов ЭЛТ: масса и габариты; энергопотребление (300-350 Ватт) и тепловыделение; излучения вредные для здоровья человека.

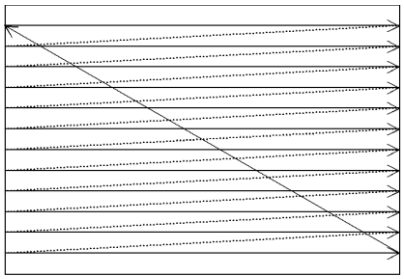


Рис.11

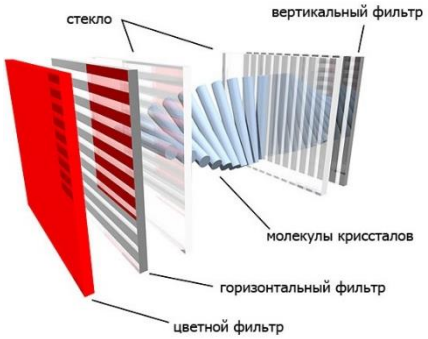


Рис.12

2. Устройство ЖК(LCD) - мониторов.

Жидкие кристаллы - это органические полимеры, которые могут проявлять свойства как жидкости, так и твердого вещества: молекулы ЖК столь же подвижны, как в жидкости, но сохраняют при этом строгую ориентацию по отношению к соседним молекулам (как в кристаллической решетке). Под воздействием тока свойства кристаллов меняются: в зависимости от подаваемого напряжения они меняют свою ориентацию, а следовательно, и

прозрачность. Изображение формируется благодаря изменению прозрачности отдельных элементов матрицы, через которую падает свет.

Если коротко, то устройство ЖК - мониторов следующее: два тонких кварцевых стекла, на внутренние стороны которых нанесена матрица электродов, где каждая «ячейка» соответствует одному пикселу (зерно экрана). Количество пикселов - всегда фиксированное число (т.к. зависит от процесса изготовления). На поверхность стекол наносится поляризующая пленка. Обе кварцевые пластины складываются вместе, между ними создается вакуум, и внутрь закачивается полимерный состав из жидких кристаллов (рис.12). Жидкокристаллическое вещество представляет собой набор овальных кристаллов, которые при высоком напряжении способны изменять свою ориентацию в пространстве (т.е. они все выстраиваются в одном направлении), заодно изменяя и угол поляризации проходящего через них света. По сути дела ЖК ячейка представляет собой электронно-управляемый светофильтр состоящий из трех элементов разных цветов (RGB), принцип действия которого основан на эффекте поляризации световой волны. Каждая ЖК-ячейка генерирует 1 пиксел изображения и управляется отдельным транзистором, находящимся в углу ячейки и отвечающим за нужное состояние этой ячейки.

Характеристики:

- Размер экрана (15”, 17”, 19” 21”, 23”, 25”);
- Ориентация (портрет и ландшафт);
- Поле обзора, характеризуется углами обзора. По вертикали 120° - 170°, по горизонтали 140° - 170°;
- Разрешение экрана. Строго фиксировано для каждого монитора. Разрешение определяется размером ЖК-экрана и размером отдельной ЖК-ячейки. Чаще всего используется разрешение 1024 x 768. Можно изменить разрешение экрана для ЖК-монитора, но при этом пострадает качество изображения.
- Частота обновления кадров (частота развертки) от 60 до 120 Гц.
- Яркость. Чем выше яркость, тем лучше: изображение будет более красочным, блики станут менее заметны, углы обзора увеличатся. Яркость всегда можно уменьшить с помощью регуляторов, а вот ее недостаток восполнить нельзя. Средняя яркость для ЖК - мониторов составляет 150-200 кд/м^2.
- Контрастность изображения на ЖК – экране показывает, во сколько раз изменяется его яркость при изменении уровня видеосигнала от максимального до минимального. Эту величину часто называют коэффициентом контрастности и обозначают в виде отношения (например, 150:1, где 150 – это значение яркости, а 1 - видеосигнал). Чем выше контрастность ЖК – экрана, тем более четкое изображение можно на нем получить. Приемлемая цветопередача обеспечивается при контрастности не менее 130:1, высококачественная цветопередача требует контрастности 300:1.
- Инертность ЖК – экрана характеризуется минимальным временем, необходимым для активизации его ячейки. Это время у современных ЖК - мониторов значительно уменьшилась по сравнению с первыми моделями. Инертность современных ЖК – мониторов составляет 15-70 мс, т.е. соответствует значениям аналогичных параметров обычных мониторов.
- Палитра. В отличие от традиционных, мониторы имеют ограниченную палитру, т.е. характеризуются ограниченным количеством воспроизводимых на экране оттенков цветов. Эта ограниченность объясняется тем, что ЖК – монитор является цифровым и требует выполнения дополнительного аналого-цифрового преобразования RGB – сигнала видеоадаптера перед подачей его на ЖК – ячейки. Типовой размер палитры современных ЖК – мониторов составляет 265 144 или 16 777 216 оттенков цветов.

Достоинства мониторов: нет излучений; энергопотребление на 70% ниже чем у CRT (не больше 35-50 Вт в рабочем режиме и 5-8 Вт в режиме ожидания); размер и масса.

Недостатки: зависимость качества изображения от внешнего света; ограниченный угол обзора; искажение цветопередачи; проблемные пикселы (наличие на некоторых ЖК – экранах проблемных, «битых» пикселов, яркость которых при смене изображения и даже при выключении монитора остается неизменной).

3. Плазменные мониторы

Вместо ЖК вещества используется ионизированный газ. Его молекулы обладают способностью излучать свет в процессе рекомбинации, т.е. восстановления электрической нейтральности. Для приведения газа в ионизированное состояние, т.е. в состояние плазмы используется высокое напряжение. При ярком свете изображение на экране плазменного монитора выглядит немного расплывчатым. В настоящее время выпускают модели с экраном 42”. Плазменные мониторы стоят дороже ЖК.

Рис.13.

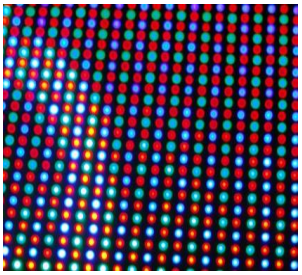


Рис.14.

3. Светодиодные мониторы

У данного вида мониторов каждой точкой, пикселем (pix) является один или несколько полупроводниковых светодиодов (рис.14). Светодиодные экраны по принципу построения делятся на два типа — кластерные и матричные. В кластерных экранах каждый пиксель, содержащий от трех до нескольких десятков светодиодов, объединён в отдельном светоизолированном корпусе. Такой конструктивный элемент называется кластером. Кластеры, образующие информационное поле экрана, закреплены при помощи винтов на лицевой поверхности экрана. От каждого кластера отходит жгут проводов, подключаемый, посредством электрического разъема, к соответствующей схеме управления (плате). Такой способ построения полноцветных светодиодных экранов постепенно отмирает, уступая место более технологичному матричному принципу. В матричных светодиодных экранах кластеры и управляющая плата объединены в единое целое — матрицу, то есть на управляющей плате смонтированы и светодиоды и коммутирующая электроника. В зависимости от размера и разрешения экрана, количество светодиодов, составляющих пиксель, может колебаться от трех до нескольких десятков. Светодиодные мониторы получают всё большее как рекламные и информационные таблоиды на улицах крупных городов, в качестве дорожных знаков.

Преимущества: высокая яркость; возможность сборки экрана больших размеров (до сотен метров в ширину и высоту); произвольное соотношение высота/ширина; надёжность (повреждение части экрана не ведёт к его неработоспособности в целом); возможность уличного круглогодичного использования.

К недостаткам можно отнести: довольно большой размер зерна экрана; низкое разрешение; самостоятельной сборки; высокая стоимость.

Клавиатура

Клавиатура - это на сегодня это не только основное устройство ввода информации в компьютер, но и управление им.

Каждая клавиша клавиатуры представляет собой крышку маленького переключателя (механического или мембранного). Под клавишами большинства клавиатур находятся два набора проводков - горизонтальных и вертикальных. Когда компьютер включен, микропроцессор клавиатуры посылает электрические импульсы по вертикальным проводкам в

поисках сигнала. Когда на клавишу нажимают, проводки под ней соприкасаются, замыкая электрическую цепь. Это указывает микропроцессору на то, что клавиша была нажата; микропроцессор проверяет горизонтальные проводки, чтобы выяснить местонахождение клавиши. Микропроцессор знает, какая клавиша нажата, т.к. активизируется только одна пара проводков.

Т.о., находящийся в клавиатуре небольшой процессор отслеживает состояние всех переключателей. При нажатии или отпускании каждой клавиши он посылает основному процессору соответствующее сообщение - прерывание.

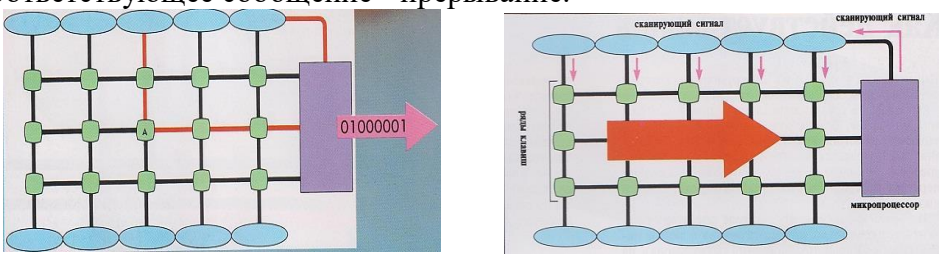


Рис.15.

Получив сигнал от клавиатуры, ЦПУ прерывает работу, и обрабатывает этот сигнал (ставит ему в соответствие код символа), после чего отправляет код в буфер клавиатуры и продолжает прерванную работу. Работающая программа, при необходимости, считывает из буфера требуемый код и интерпретирует его как текстовый или управляющий символ, или может проигнорировать его. Если удерживать клавишу в нажатом состоянии, ее сигналы будут поступать в буфер непрерывно и могут переполнить его (ели программа не будет успевать их обрабатывать). При этой ситуации компьютер еще некоторое время может выполнять какие - либо действия (например, перемещать курсор, печатать символ...), хотя пользователь уже не трогает клавиши. При «зависании» машины, после нескольких нажатий клавиш компьютер начинает реагировать коротким «писком». Это означает, что буфер клавиатуры переполнен, а заглядывать в него и выбирать имеющиеся там коды – некому.

Буфер клавиатуры имеет объем 32 байта и располагается в памяти начиная с адреса 0040:001E. Он может накапливать до 15 нажатий на клавишу, независимо от того, являются ли они однобайтными кодами ASCII или двухбайтными - буфер отводит по два байта для каждого нажатия клавиши (последнее нажатие, которое может быть максимально 16-м по счету, зарезервировано для клавиши <Enter>). Для однобайтных кодов первый байт содержит код ASCII, а второй - скан-код клавиши. Этот номер обычно совпадает со скан-кодом клавиши, но не всегда, поскольку некоторые клавиши могут комбинироваться с клавишами сдвига для генерации различных кодов.

Типы клавиатур

1. Механические - отличаются тем, что включают в себя печатную плату с встроенными металлическими контактами (рис.16).

2. Пленочные клавиатуры (мембранными) – в них входят три пленки, которые, в свою очередь содержат нанесенный диэлектрик посередине и контуры контактов. Сейчас уже не удивляет тот факт, что клавиатуру можно помыть под краном. Именно с пленочными клавиатурами можно провести подобную процедуру, так как в структуре этого вида клавиатуры находится резиновая прокладка между верхней пленкой и верхней частью устройства.

3. Гибкая клавиатура изготовлена из силиконовой резины, удобна тем, что она бесшумна, ее можно свернуть в трубочку, не боится ударов, падений, пыли, воздействий жидкости, часто имеет встроенную подсветку (рис.16).

4. Сенсорная клавиатура. Принцип работы сенсорной клавиатуры основан на использовании сенсорных датчиков, которые регистрируют колебания электроемкости. Руки человека имеют некий заряд энергии, что сразу фиксирует датчик. Сенсорная клавиатура характеризуется отсутствием физических кнопок и предполагает в дальнейшем отказ от использования манипулятора мышь. Поверхность сенсорной клавиатуры абсолютно гладкая, что полностью исключает загрязнение и востребовано в медицинских учреждениях.

5. Лазерная клавиатура – вариант беспроводного подключения клавиатуры. Маленькая коробочка с лазерным проектором ставится на стол и раскладка клавиатуры проецируется на любую ровную твердую поверхность. С помощью программного обеспечения, которое поставляется в комплекте, можно настроить звук клавиш при печатании, чувствительность и яркость. Увы, клавиатура частенько ошибается или вовсе не печатает нажатую букву (рис.17).

Рис.16.

Рис.17.

Рис.18.

Рис.19.

Подключение клавиатуры к компьютеру.

Обычно подключение клавиатуры происходит с помощью USB-порта, реже - с помощью разъема PS/2 - это шестиконтактный разъем, ранее применяемый для подключения клавиатуры и мыши. С развитием технического прогресса актуальной стала проблема дефицита портов USB. Ведь через USB-порты к компьютеру обычно подключается принтер, сканер, наушники, диктофон, фотоаппарат, и т.д.

: 

Рис.20.

Сегодня есть вариант беспроводного с помощью Bluetooth-подключения (беспроводные клавиатуры). Это устройство идет в комплекте с клавиатурой. Сейчас есть складные Bluetooth-клавиатуры, удобные для Ipad, iPhone (рис.20).

Принтеры

Принтер - это печатающее устройство, предназначенное для вывода информации на бумагу. Существует несколько классификаций принтеров. Самая распространенная - по принципу печати. По этой классификации принтеры бывают: матричные, струйные, лазерные,

Матричные

Благодаря деловому характеру применения - довольно распространенный тип принтеров: кассовые аппараты, принтеры для печати проездных документов и т.д. (рис.21).

Рис.21

Рис.22

Принцип печати следующий: печатающая головка содержит вертикальный ряд тонких металлических стержней (иголок). Обычно их бывает 9 или 24. Головка движется вдоль печатаемой строки , а стержни в нужный момент ударяют по бумаге через красящую ленту. Это обеспечивает формирование на бумаге символов и изображений. Для повышения качества

печати применяются повторные проходы головки. Применение разноцветной ленты и соответствующей программы позволяет осуществить некое подобие цветной печати. Созданы принтеры, позволяющие печатать одновременно два разных документа. Они имеют два тракта подачи бумаги. Скорость печати для текста - от 60 до 10 секунд на страницу. Рисунок печатается до 5 минут. Существуют специальные высокопроизводительные матричные принтеры для банков, телефонных компаний - скорость их печати несколько тысяч строк в минуту.

Основными достоинствами матричных принтеров являются :

- не дороги по цене,
- низкая стоимость печати страницы текста,
- возможность рулонной подачи бумаги,
- не высокие требования к характеру окружающей среды,
- способность печатать на плотном и толстом носителе,
- способность работать в тяжелых и экстремальных условиях.

К недостаткам этого вида относят:

- низкое качество печати,
- значительный шум при работе,
- мало пригоден для цветной печати.

Струйные

Сейчас этот тип принтеров находится в стадии расцвета. Принцип печати у них следующий: изображение формируется микрокаплями специальных чернил, выбрасываемых на бумагу через тюзы или сопла (полые трубочки) в печатающей головке, под действием выталкивающей мембраны (пьезоэлектрическая технология печати) или при помощи нагревающих элементов (термоэлектрическая технология печати). Головка движется по горизонтальной направляющей, а по окончании печати каждой полосы бумага продвигается по вертикали. Повышение качества печати зависит от размера капель красителя. При диаметре капель 40 микрон, на дюйм приходится порядка 600 капель. Для подготовки высококачественной цветной продукции используется термопереносная технология. Скорость печати для страницы текста: от 60 до 15 секунд на страницу, рисунок - до 2 минут (рис. 22).

К достоинствам данного типа относят:

- высокое качество печати достигающее фотографического,
- разработаны дешевые модели принтеров для широкого пользователя,
- при работе создают незначительный шум,
- наличие автоматической подачи бумаги,
- огромное количество, предлагаемых на рынке моделей.

К недостаткам следует отнести:

- довольна высокая стоимость заправочных чернил, а следовательно и печатной страницы,
- высокие требования к бумажному носителю,
- существуют ограничения на количество отпечатанных листов в месяц,
- принтеры требуют тщательного ухода и обслуживания.

Лазерные

Обеспечивают в настоящее время наилучшее качество печати, близкое к типографскому. Принцип печати*:* в лазерных принтерах используется принцип ксерографии: изображение наносится на бумагу со специального барабана, к которому электрически притягиваются частички краски (тонера). Отличие от копировального аппарата состоит в том, что печатающий барабан электризуется с помощью лазера по командам от компьютера. В настоящее время выпускаются лазерные принтеры коллективного пользования.

Эти принтеры находятся в постоянном развитии. Их характеризует ряд дополнительных свойств: возможность работать в сети, дистанционное управление, высокое быстродействие,

повышенная прочность, способность печатать долгое время без перерыва, возможность двухсторонней печати, наличие дополнительного оборудования для послепечатной доработки.

Рис. 23.

Рис. 24.

Скорость печати у них: для текста от 15 до 5 секунд на страницу, для рисунка - до минуты. Сетевые принтеры печатают 15 - 40 страниц текста в минуту. К достоинствам этого типа следует отнести:

- позволяют получать самое высокое качество печати,
- позволяет получать цветную печать,
- практически бесшумны при работе.
- у большинства принтеров есть два лотка подачи бумаги - для ручной и автоматической,
- огромное количество предлагаемых моделей.

К недостаткам относятся:

- довольно высокая стоимость принтера,
- требовательны к бумажному носителю - можно использовать бумагу только высокого качества,
- требуют тщательного ухода и обслуживания.

Плоттеры

Группа широкоформатных принтеров, используемая для полиграфической печати, деловой графики. Предшественниками широкоформатных принтеров и плоттеров были графопостроители - устройства планшетного типа, в которых над бумажным носителем помещался пишущий узел - подобие рапидографа. В нем закреплялись сменные флакончики с чернилами разного цвета. Развитие струйных технологий позволило использовать постоянную печатающую головку большого размера, способную наносить на бумагу цветные полосы всеми цветами сразу. В этих принтерах иногда используется технология термопечати. Данный вид принтеров широко используется при подготовке различной рекламы, в художественном дизайне.

Скорость печати плоттера: 5-7 страниц формата А4 при ширине носителя 1,5 метра.

К достоинствам плоттеров следует отнести:

- получение цветной печати очень высокого качества (порядка 1440 dpi),
- в качестве носителя можно использовать полимерную пленку,
- высокая производительность,
- скоростная печать,
- автоматическая и ручная настройка цвета,
- отпечатки могут находится на открытом воздухе до 2 лет, а в помещении до 5 без потери качества.

Недостатками можно считать:

- высокие требования к корректности цветопередачи,
- очень высокая цена принтера,
- для работы требуется компьютер большой мощности.

Дополнительные устройства

Манипуляторы

Для работы со многими современными программами, работающими в графическом режиме, практически обязательными является использование различных манипуляторов. К ним относят *мышь, трекбол, трекпоинт, сенсорная панель* (рис. 25)**.** Они позволяют быстро перемещать курсор по экрану, а так же указывать на те или иные элементы на экране. Как правила, имеют одну или две кнопки, выполняющие назначение клавиш [Enter] - ввод и [Esc] – отмены.

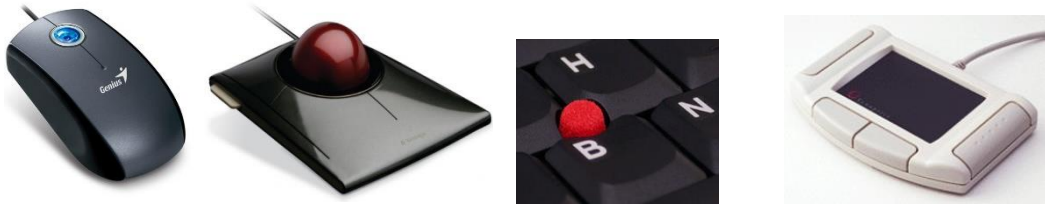


Рис.25

Акустические системы

Термин «Мультимедиа» переводится на русский язык как «многосредность», т.е. способность работать с информацией в различных видах: текст, графика, музыка, речь, другая звуковая информация, анимационные фильмы и т.д. Для работы с мультимедийной информацией компьютер оснащается звуковой картой, акустической системой (колонки, наушники, гарнитура) (рис.26).

Рис.25

Сканер

Сканер - это устройство, которое позволяет оптическим путем вводить черно-белую или цветную печатную графическую информацию с листа бумаги. Сканеры бывают планшетные, ручные и протяжные (рис.26). При сканировании текста изображение считывается с помощью специальных датчиков и в цифровом коде передается в компьютер, где по сложным алгоритмам с помощью специальных программ превращается в обычный текст, редактируемый на экране. Сканеры используются при работе на профессиональном уровне с текстом, графикой, в издательском деле

Рис.26.

Кроме выше перечисленных устройств разработано огромное количество других, позволяющих использовать компьютер для автоматизации труда в любой профессии.

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом и требованиями к отчету по лабораторной работе.
2. Изучить устройство различных моделей монитора, клавиатуры, мыши, принтера, сканера.
3. Познакомиться с прайс-листом компьютерной фирмы. Выяснить характеристики предлагаемых комплектующих.

4. Подобрать комплектацию компьютера для следующих целей:
    a. Офисный компьютер;
    b. Компьютер для образовательного учреждения;
    c. Компьютер для дома;
    d. Компьютер для профессиональной работы с графической и видеоинформацией.

Контрольные вопросы

1) Что представляют собой порты ввода/вывода?
2) Как устроена клавиатура?
3) Что представляют собой и как используется мышь?
4) Что такое тачпад?
5) Что такое трекпойнт?
6) Что такое трекбол?
7) Что такое принтер и какими они бывают?
8) Опишите основные параметры и характеристики сканеров.
9) Что отличает ЖК мониторы от электронно-лучевых трубок, как устроены ЖК мониторы?

Лабораторная работа 16. Определение характеристик аппаратной части компьютера. Тестирование компьютера

Цель работы: Познакомиться с программами, определяющими основные характеристики компонентов компьютера (EVEREST, AIDAи др.). Изучить основы тестирования персонального компьютера; познакомиться с тестовыми утилитами MadOnion 3DMark 2001, Sandra 2005 (2011). Научиться определять характеристики и тестировать вычислительные машины

Основные теоретические сведения

Краткие сведения о тестировании

Процесс тестирования заключается в определении производительности компьютера или какой-либо из его подсистем. Именно тестирование позволяет сравнивать компьютеры по производительности и подбирать оптимальное сочетание комплектующих для достижения наивысшей производительности. Это основная, не единственная задача тестирования, поскольку в процессе тестирования проверяется стабильность работы компьютера, выявляются слабые места, определяется корректность работы драйверов устройств, а также корректность самих устройств главной задачи тестирования является определение производительности компьютера, когда пользователь имеет реальную возможность оценить, насколько хорош или плох его компьютер при выполнении тех или иных задач, а также выяснить, как сказалась модернизация компьютера на его работе.

Разберемся с тем, что такое производительность компьютера и в чем ее можно измерять. Сами по себе устройства не имеет такого свойств как производительность, поэтому правильнее будет говорить о производительности аппаратной части при выполнении конкретного приложения или задачи. Кроме того, тестировать можно как компьютер в целом, так и отдельные его подсистемы. При этом компьютер можно условно разделить на несколько подсистем:

- процессорную;
- дисковую;
- видео;
- сетевую.

Такое деление весьма приблизительно, однако именно тестирование отдельных подсистем компьютера позволяет понять, насколько сбалансирован компьютер и какая из его подсистем является наиболее слабым местом.

Для тестирования компьютера в целом и его отдельных подсистем используются принципиально разные подходы и соответственно различные тестовые утилиты. Оценить работу компьютера в целом, то есть интегрально, можно на основе только реальных приложений, так как на результат теста влияет взаимодействие всех подсистем компьютера. Для этого используются тестовые утилиты, или ***бенчмарки,*** эмулирующие работу пользователя с реальными приложениями. Такие тесты запускают реальные программы и измеряют время их выполнения. При этом для создания наибольшей нагрузки на систему воспроизводится максимальная активность приложений.

Для тестирования отдельных подсистем компьютера больше подходят ***синтетические*** (или искусственные) тесты, которые не имеют отношения к реальным приложениям и *предназначены для создания стрессовой нагрузки на отдельные подсистемы компьютера*. Синтетические тесты эмулируют специализированные задачи, позволяющие сконцентрироваться на тестировании отдельных компонентов, причем результаты этих тестов не должны зависеть от конфигурации всего компьютера или, по крайней мере, влияние остальных компонентов на результаты теста должно быть сведено к минимуму. (Например, если синтетический тест направлен на определение производительности процессора, то результат не должен зависеть от модели используемого жесткого диска или видеокарты)

Тесты на реальных приложениях

С точки зрения пользователя, понятие производительности компьютера связывается со временем выполнения определенного тестового приложения. В этом смысле компьютер, выполняющий тот же объем работы за меньшее время, является более быстрым и, следовательно, более производительным.

Временной показатель производительности, который также называется ***временем отклика***, или ***временем выполнения*** — это задержка выполнения задания, включающая:

- работу процессора,
- обращения к диску,
- обращения к памяти,
- ввод-вывод,
- накладные расходы операционной системы.

Примером, когда именно время является основным показателем для оценки производительности компьютера, могут служить такие простые тесты, как инвертирование звуковых файлов из формата .wav в .mp3 или время сжатия файла с использованием архиваторов.

Естественно, что время выполнения определенной задачи — не единственный возможный показатель производительности компьютера. Как правило, в бенчмарках для определения рейтинга производительности используются собственные условные единицы измерения, которые профессионалы называют «попугаями» Так, если в процессе тестирования выполняется несколько задач, то для получения интегрального результата может использоваться довольно сложная схема, учитывающая значимость того или иного теста. Классическим примером такого подхода является популярный тест MadOnion 3DMmark, результат которого выдается в единицах 3Dmark's. Хотя такие единицы измерения и не имеют никакого физического смысла, они могут использоваться для сравнения различных систем друг с другом.

Другим подходом к расчету результата теста является использование относительных единиц измерения. Для этого выбирается некоторый эталонный компьютер, производительность которого, измеряемая бенчмарком, и считается равной единице (при этом собственно производительность измеряется как скорость выполнения тестовой программы), а производительность всех остальных компьютеров измеряется уже по отношению к производительности эталонного компьютера. Если в таком бенчмарке результат тестирования

равен 30 единицам, то это означает, что данный компьютер в 30 раз производительнее эталонного.

Для получения корректных результатов при использовании тестов на основе реальных приложений нужно соблюдать ряд требований, что порой сильно ограничивает возможность их использования в домашних условиях.

Правила тестирования

1. Правило первое: ***Тестирование необходимо проводить на «чистой» операционной системе,*** то есть для тестирования используется компьютер, на котором установлены операционная система, необходимые драйверы и сам тест. Больше никаких программ устанавливать нельзя. Как правило, столь жесткое ограничение является требованием самого теста, поскольку тесты на основе реальных приложений устанавливают и сами приложения, с которыми они работают. Если же на компьютере уже установлено нечто, то конфликта не избежать. Кроме того, даже если тест корректно устанавливается и запускается, пользовательские приложения, которые загружаются в фоновом режиме, могут отразиться на результатах тестирования.

2. Второй важный момент связан с самой операционной системой, а точнее с ее локализованной версией. Нередко установка русской версии операционной системы приводит к сбоям в тестировании. Отчасти проблема заключается в том, что тестовые утилиты могут иметь встроенные макросы для обработки результатов. Если, используется русская система мер, где в качестве разделителя целой и дробной части числа используется запятая, а макрос, обрабатывающий результаты, написан в программе, где в качестве такого разделителя используется точка, то результат будет некорректен. Отсюда следует второе правило: следует ***использовать «родные» версии операционных систем.***

3. Третье правило — ***предварительная дефрагментация жесткого диска.*** Поскольку при работе с реальными приложениями результат зависит, в том числе и от производительности жесткого диска, перед началом проведения тестирования необходимо обязательно выполнить процедуру дефрагментации. Такие известные тестовые утилиты, как Ziff Davis Business Winstone и Office Productivity, автоматически запускают процесс дефрагментации перед началом тестирования.

4. Четвертое правило тестирования касается ***количества повторений одного и того же теста.*** Хорошим тестом может считаться только такой бенчмарк, который показывает стабильные результаты с минимальным разбросом. Если результаты тестирования по нескольким запускам отличаются друг от друга более чем на 5%, то этим тестом лучше не пользоваться. Но даже если тестовая программа показывает стабильные результаты, тест необходимо запускать ***как минимум три раза.***

Самих тестов на реальных приложениях не много. Их можно условно разделить на:

- тесты, построенные на офисных приложениях,
- тесты, построенные на мультимедийных приложениях,
- игровые тесты.

Признанными и наиболее известными тестами являются:

- SYSmark (2002, 2004, 2007) Office Productivity;
- SYSmark (2002, 2004, 2007) Internet Content Creation;
- ZD Business Winstone 2001 v.1.0.2;
- ZD Content Creation Winstone 2002 v.2.0;
- 3Dmark.

Тестовый пакет SYSmark 2002, разработанный компанией BAPCO, состоит из двух частей: Office Productivity и Internet Content Creation. Тест Office Productivity предназначен для определения производительности компьютеров при работе с офисными приложениями. В этом тесте эмулируется работа пользователя с такими приложениями, как Dragon NaturallySpeaking Preferred 5, McAfee Virus Scan 5.13, McAfee Virus Scan 5.13, Microsoft Access 2002, Microsoft

Excel 2002, Microsoft Outlook 2002, Microsoft Word 2002, Netscape Communicator 6.0 и WinZip 8.0.

Тест Internet Content Creation предназначен для выявления производительности компьютера при работе с приложениями, использующимися для создания Интернет-контента. В этом тесте больший упор делается на мультимедийные приложения, а именно: Adobe Photoshop 6.0.1, Adobe Premiere 6.0, Macromedia Dreamweaver 4, Macromedia Flash 5 и Microsoft Windows Media Encoder 7.1.

Отличительной особенностью пакета SYSmark является то обстоятельство, что тестовые приложения запускаются не последовательно друг за другом, а параллельно, то есть одновременно эмулируется работа нескольких программ, некоторые из которых активны, а другие выполняются в фоновом режиме, что в полной мере соответствует реальной ситуации.

Альтернативой тестам из пакета SYSmark является тестовый пакет компании Ziff Davis (ZD), также состоящий из двух тестов: Business Winstone 2001 v.1.0.2 и Content Creation Winstone 2002 v.2.0.

Рассмотренные тесты относятся, к профессиональным тестам, и их использование в домашних условиях на рабочей машине, довольно сомнительно.

Для домашнего пользователя лучше подойдут такие тесты, которые можно запускать без переустановки ОС то есть на рабочем компьютере**.** Среди тестов, работающих с реальными приложениями, можно выделить широко известный бенчмарк ***MadOnion 3DMark***, который предназначен для тестирования видеоподсистемы, но прекрасно нагружает и процессорную подсистему компьютера. Единственное, от чего результаты теста практически не зависят, — это от дисковой подсистемы компьютера.

Тест построен на основе популярных игровых приложений и поддерживает технологию DirectX и позволяет производить различные настройки. Настроек у этого теста достаточно много, но когда речь заходит об официальных результатах, то, как правило (если это не оговаривается отдельно), тест запускается с настройками по умолчанию. Бенчмарк MadOnion 3DMark выдает результат в условных единицах, позволяет просмотреть подробный отчет по каждому тесту и сохранить его в текстовом виде. Отметим, что, несмотря на всеобщее признание, у этого теста есть и недостаток: разброс в результатах теста может достигать 5%, вследствие чего этот тест необходимо запускать как минимум три раза.

Другими распространенными тестами, построенными на реальных приложениях, являются игровые бенчмарки. Среди них наиболее популярны Unreal Tournament 2003, Quake III Arena, DroneZ mark, ChameleonMark, Codecreatures Benchmark Pro, Comanche 4 Demo Benchmark Test.

Все игровые тесты выдают результаты ***в кадрах в секунду***. Эти единицы измерения типичны для тестирования видеоподсистемы компьютера и определяют количество обрабатываемых кадров в секунду. При использовании игровых тестов следует иметь в виду, что их результаты сильно зависят от установленного разрешения экрана, глубины цвета и частоты строчной развертки, поэтому при использовании этих бенчмарков необходимо учитывать данное обстоятельство.

Синтетические тесты

Синтетические тесты позволяют ***эмулировать нагрузку на отдельную подсистему компьютера.*** Таких тестовых утилит довольно много, но, пожалуй, больше всего бенчмарков этого класса разработано для тестирования процессоров. Многие тесты измеряют производительность процессоров в единицах MIPS и Flops.

MIPS (Millions of Instructions Per Second) — это количество инструкций, выполняемых процессором в секунду. Необходимо сделать одно замечание: количество инструкций, выполняемых процессором за единицу времени не всегда отражает производительность процессора.

Например: программа, откомпилированная под CISC-архитектуру, выполняется за 1000 инструкций, а та же программа, откомпилированная под RISC-процессор, — за 2000 инструкций. Если обе программы выполняются за одно и то же время, то при

одинаковом объеме работы, сделанном процессорами, производительность RISC-процессора в единицах MIPS в два раза больше, чем CISC-процессора. Более того, единицы MIPS будут различаться не только для процессоров различной архитектуры, но и для разных программ, работающих на одном процессоре. Поэтому к MIPS следует относиться достаточно осторожно. Сравнивать процессоры в единицах MIPS корректно лишь в том случае, если они имеют одинаковую архитектуру, то есть различаются только тактовой частотой.

Аналогичным недостатком обладают и тесты, измеряющие производительность процессора в *Flops (Giga Floating Point Operations Per Second)*. Эти единицы показывают количество операций с плавающей точкой (в миллиардах), выполняемое процессором за одну секунду.

Описать все тесты довольно сложно, поэтому остановимся лишь на одном, который, можно считать одним из лучших: бенчмарк CPU RightMark (http://cpu.rightmark.org), который использует вычислительные задачи, такие как решение системы дифференциальных уравнений, соответствующих моделируемым физическим процессам взаимодействия системы многих тел, и решение задач из области трехмерной графики. Отличительной особенностью теста CPU RightMark является то, что результаты зависят от самого процессора, памяти и шины «память-процессор», а влияние остальных компонентов системы сведено к минимуму. Кроме того, этот тест обладает очень высокой стабильностью — разброс результатов менее 1%.

CPU RightMark содержит два программных блока, один из которых предназначен для расчета физической модели, то есть для решения системы дифференциальных уравнений, а второй отвечает за визуализацию (рендеринг) полученного решения — за прорисовку сцены. У каждого блока имеются различные варианты, оптимизированные под разные системы процессорных команд. Расчет физической модели возможен с использованием набора команд SSE2 (процессор Intel Pentium 4) либо с использованием набора команд для FPU. Блок визуализации состоит из двух частей — блока предварительной обработки и блока отрисовки (рендеринга). Первый блок откомпилирован с использованием набора команд сопроцессора x86, а второй имеет несколько вариантов, оптимизированных под различные наборы инструкций: FPU + GeneralMMX, FPU + EnhancedMMX и SSE + EnhancedMMX. Скорость работы блока визуализации отражает производительность процессора и памяти при выполнении геометрических расчетов с использованием действительных чисел одинарной точности.

Тестирование видеоподсистемы компьютера проводится в основном игровыми тестами, о которых мы уже упоминали. Однако следует иметь в виду, что видеоподсистему компьютера невозможно протестировать изолированно, поэтому результаты теста будут зависеть не только от установленного видеоадаптера, но и от процессора и памяти. А вот дисковую подсистему компьютера можно тестировать синтетическими тестами. Наиболее подходящим для этого среди всех известных нам тестов является тест IOmeter 99.10.20— разработка компании Intel, представляющая собой очень тонкий инструмент для измерения производительности жестких дисков.

Утилита IOmeter позволяет прецизионно настраивать тест на измерение производительности жестких дисков при выполнении ими специфических задач, причем дает возможность работать с неразбитыми на логические разделы жесткими дисками, что позволяет тестировать диски независимо от файловой структуры и свести к нулю влияние операционной системы.

При тестировании возможно создание специфической модели доступа (или паттерна), способной конкретизировать выполнение жестким диском специфических операций. В случае создания конкретной модели доступа возможно менять следующие параметры:

- размер запроса на передачу данных;
- процент доступа по данному запросу;
- случайное/последовательное распределение;

- распределение операций чтения/записи;
- количество отдельных операций ввода-вывода, работающих параллельно.

Последняя характеристика позволяет задавать загрузку тестируемого жесткого диска. Например, если задается всего одно обращение к диску, то при 100% случайном доступе можно измерить среднее время случайного доступа при работе с той или иной моделью доступа. Однако при работе с реальными приложениями более показательной является ситуация, когда одновременно осуществляется несколько обращений к диску. Так, при дефрагментации диска наблюдаются всплески с количеством одновременных операций ввода-вывода, превышающим 100.

В приведенном списке тестовых пакетов указаны далеко не все бенчмарки. Сравнивать производительность отдельных комплектующих или компьютера в целом корректно только в том случае, если для измерения производительности используется не один тест, а несколько разных. Только в этом случае можно получить объективную оценку сравнения и сказать, с какими задачами компьютер справляется лучше, а с какими — хуже.

Получение сведений о системе

Пользователю необходимо знать подробную информацию о своем компьютере и его характеристики в случае возникновения и диагностики неполадок системы, обновления драйверов или, модернизации компьютера. Эту информацию можно узнать различными способами: как с помощью специальных программ, так и стандартными средствами самой Windows.

1 способ - посмотреть информацию, которая возникает на мониторе при включении компьютера. Для этого:

- При включении и загрузке компьютера нажать клавишу «PAUSE» на клавиатуре.
- Просмотреть все сведения на экране монитора и записать их.

Из сводной таблицы сведений о конфигурации компьютера, которые выводятся на экран монитора, можно узнать:

- тип процессора, его тактовую частоту и идентификационный номер;
- объём и тип установленной оперативной памяти;
- объем кэш-памяти;
- сведения об установленных слотах памяти;
- информация о видеоадаптере;
- сведения о жестком диске, приводе компакт-дисков или DVD и пр.

Чтобы продолжить загрузку, следует нажать клавишу «Esc» («Escape») на клавиатуре.

2 способ - посмотреть в Главном Меню. Для этого:

- Открыть меню «Пуск» → «Мой компьютер» (щелкаем на нем правой кнопкой мыши) → «Свойства» → появится диалоговое окно «Свойства системы» → вкладка «Общие».

Здесь можно посмотреть номер версии Windows, общую информацию о процессоре (ЦП), его тактовую частоту, объем оперативной памяти.

Добраться до окна «Свойства системы» можно иначе: «Пуск» → «Панель управления» → «Производительность и обслуживание» → «Система». В результате появится окно «Свойства системы».

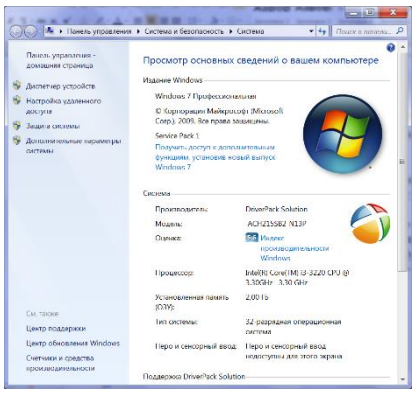


Рис. 1. Окно системы в Windows 7

Рис. 2. Окно системы в Windows XP

3 способ - с помощью утилиты ***msinfo32.exe*** «Сведения о системе». Для этого:

- «Пуск» → «Выполнить» → в появившейся командной строке набрать *msinfo32* – в появившемся окне «Сведения о системе» ознакомиться с полученной информацией.

Здесь можно найти следующие сведения: данные о версии Windows, процессоре, версии BIOS, полный и доступный объем физической памяти, объем файла подкачки и др. Вызвать это окно можно и через «Главное меню»: «Пуск» → «Стандартные» → «Служебные» → «Сведения о системе» (рис.3).

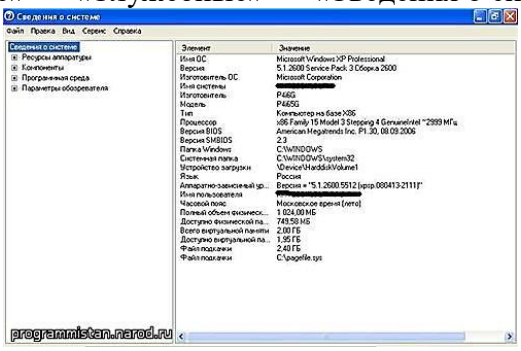


Рис.3.

Также для просмотра сведений о системе можно воспользоваться комбинацией клавиш «Ctrl» + «Shift» + «F1».

Программные средства получения подробной информации о компьютере

Есть программные способы просмотра подробных сведений о компьютере, его реальных характеристик и всевозможных параметрах системы:

1. С помощью программ тестов-информаторов ***Sisoft Sandra, PC Wizard, Everest, Aida*** и др., дающих не только полную информацию о ПК, но способных протестировать и проанализировать его работу.
2. Утилитой ***dxdiag.exe*** (средство диагностики DirectX от Microsoft Corporation), запущенной из «командной строки».

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом и требованиями к отчету по лабораторной работе.
2. Запустить специализированное программное обеспечение (EVEREST, AIDA или др.).
3. Определить основные характеристики устройств, входящих в состав ПК. Результат оформите в виде таблицы:

Устройство	***Характеристика***	***Значение***	***Путь к информации в программе***
Системная плата	Форм - фактор		
	Тип процессорного разъема		
	Наличие разъемов расширения системы (какие и сколько)		
	Разъемы ОЗУ (какие и сколько)		

	Чипсет (фирма - производитель)		
Процессор	Фирма - производитель		
	Тактовая частота		
	Числа транзисторов на кристалле		
	Количество контактов (pin)		
	Наличие кэш – памяти 1 и 2 уровня		
	Потребляемая мощность		
Оперативная память	Тип		
	Объем		
Постоянная память	Тип		
	Дата изготовления		
Накопители	Физические и логические HDD, их имя и объем		
	Оптические накопители (имя, вид, скорость)		
	Floppy – накопители (имя, стандартный объем носителей)		
Монитор и видеоадаптер	Тип монитора		
	Тип видеоадаптера		
	Наличие графического процессора		
	Наличие и объем видеопамяти		
	Максимально поддерживаемое разрешение		
Порты	Тип и количество		

4. С помощью теста MadOnion 3DMark протестировать видео подсистему ПК (тест необходимо запускать не менее трех раз). Тестирование производить с настройками по умолчанию. Сохранить результаты теста в виде документа (скриншота).
5. Протестировать видео систему в режиме 2D с помощью программы Tom2D_ru. Результаты сохранить.
6. С помощью универсального теста Sandra 2005 (2011) протестировать ПК:
   a. Собрать информацию о ПК (Information Modules/ Сводная информация);
   b. Протестировать процессор с помощью Benchmarking Modules / CPU Arithmetic Benchmark и CPU Multi-Media Benchmark. (арифметический и мультимедийный тесты)
   c. Протестировать оперативную память: Benchmarking Modules / Memory Bandwidth Benchmark (тест пропускной способности, тест кэш-памяти)
   d. Сохранить результаты теста в виде документа .html (мастер создания отчета (Create a Report Wizard).
7. Определить производительность CPU с помощью программы LinX:
   a. познакомиться с сутью и возможностями программы (прочитать файл Readme.txt)
   b. провести тест с максимальной загрузкой ЦП и ОЗУ, результаты тестирования сохранить.

Контрольные вопросы

1. Что такое производительность вычислительной машины?
2. В чем измеряется производительность?
3. Какими инструментами можно определить производительность компьютера и его подсистем?
4. Что такое «синтетические тесты», какого их назначение, примеры программ.

Лабораторная работа 17. Системные блоки. Комплектация системного блока

Цель работы: Познакомиться с видами корпусов системных блоков. Научиться комплектовать системный блок персонального компьютера.

Основные теоретические сведения

Корпус системного блока

Корпус для любого компьютера является не только «упаковочным ящиком», но и достаточно важным элементом, обеспечивающим размещение и жесткую фиксацию всех его устройств, обеспечение их электропитанием и защищающий довольно хрупкие «внутренности» от воздействия окружающей среды. Большинство корпусов стандартизированы - AT, ATX, microATX.

Первый индустриальный стандарт на корпуса для персональных компьютеров - AT просуществовал более 12 лет.

ATX является более современным, большинство материнских плат рассчитаны именно под него. Для него характерен более легкий доступ к внутренним узлам компьютера (зачастую без использования отвертки), улучшенная вентиляция внутри корпуса, возможность установки большего числа полноразмерных плат расширения, большие возможности по управлению энергопотреблением. ATX чаще всего для охлаждения нагнетает воздух внутрь компьютера, а не выбрасывает его наружу, что в целом менее предпочтительно при разгоне процессора/видеокарты или повышенной температуре в комнате.

MicroATX - малогабаритный вариант, хорошо подходящий для компактных базовых ПК с минимумом плат расширения (минимальными габаритами и доступной ценой).

Наиболее широкое распространение получили корпуса двух разновидностей: ***desktop***, располагающийся горизонтально на рабочем столе и применяемый по большей части в моделях ПК, производимых фирмами – «брэндами» и ***tower***, вертикально расположенный и более дешевый, массовый тип корпуса. Корпуса последнего типа подразделяются, в свою очередь, на micro-, mini-, midi- и big-tower, различающиеся по числу отсеков для 5,25" накопителей: соответственно micro-tower имеют 1 посадочное место под такие накопители, mini-tower - 2, midi-tower - 3 и big-tower - 4 и более.

Как правило, на корпусе системного блока располагаются несколько кнопок для управления компьютером (Reset, Power), светодиодные и цифровые индикаторы режимов работы (HDD, частота), замок для блокировки клавиатуры (Lock), встроенный динамик. Корпуса различных фирм могут несколько отличаться по дизайну и габаритам. Существуют специальные корпуса для мультимедиа-компьютеров, оснащенные стереоколонками, манипуляторами аудиовыхода, встроенные картридеры, для чтения флеш-карт, USB порты на передней панели. Для комфортной работы выпускаются корпуса с низким уровнем шума (low-noise), в которых применяются блоки питания с малошумящими вентиляторами.

Корпуса desktop

Чаще всего в корпусе такого типа размещаются горизонтально от 2 до 3 устройств формата 5,25" и вертикально 2 - формата 3,5", причем одно из них - с внешним доступом. Такие корпуса занимают достаточно большое пространство на рабочем месте, не всегда могут обеспечить удобный доступ к внутренним устройствам, да и иногда возникают проблемы с нормальным охлаждением процессора.

Рис.1.

Корпуса slim

Развитие идеи миниатюризации применительно к компьютерной области породило такое чудо, как предельно интегрированные системные платы формата FlexATX и их естественное продолжение - Slim, Super Slim. Корпуса тесные, возможности модернизации очень ограничены, но зато - внешне они выглядят оригинально и эксклюзивно.

Корпуса mini-tower

Довольно маленький по высоте корпус типа mini-tower использовался для материнских плат

формата BabyAT, сейчас встречается редко, так как с размещением в нем полноразмерных системных плат ATX могут возникнуть проблемы, поэтому используется для размещения малогабаритных плат форматов microATX и flexATX. Такие корпуса чаще всего используется в ПК самых простейших конфигураций и применяется в качестве офисных машин или сетевых терминалов.
Рис.3.

midi(middle)-tower

Самый распространенный на сегодня формат корпуса - midi(middle)-tower ATX, обеспечивает использование большого количества накопителей и практически всех типов системных плат при приемлемых габаритных размерах. Оптимально приспособленной для решения самого широкого круга задач, корпуса этого типа применяется практически везде. *Рис.4.*

Корпуса big(full)-tower

Являясь самыми крупногабаритными, корпуса типа big-tower обеспечивают размещение системных плат любых размеров и самого большого количества устройств формата 5,25", чаще всего 4-6. Кроме того, они обычно комплектуются блоками питания повышенной мощности.

Основная область применения корпусов - рабочие станции, небольшие серверы и компьютеры для продвинутых пользователей. Однако в связи с все ширящейся экспансией недорогих IDE RAID-контроллеров в массовые устройства, потребность в большом количестве посадочных мест для дисковых накопителей может вывести корпуса big-tower в разряд наиболее распространенных устройств, особенно если учесть, что современные высокоскоростные винчестеры в процессе работы ощутимо греются, и уже сейчас начали появляться устройства, монтируемые в 5-дюймовые отсеки и предназначенные для охлаждения 3-дюймовых HDD.

Рис.5.

Основные требования

Основными требованиями, которые предъявляется к корпусам, являются:

1. Совместимость с предполагаемым форм-фактором материнской платы и блоком питания.
2. Соответствие размеров. Корпус должен быть достаточно велик для размещения всех требуемых устройств - но в то же время достаточно мал, чтобы поместиться в отведенном для него месте.
3. Оптимальность конструкции. Сборка-разборка должна происходить просто, конструкция корпуса должна предусматривать свободный доступ ко всем компонентам.
4. Продуманность вентиляции. Схема вентиляции должна обеспечивать наиболее оптимальное охлаждение компонентов:
   - применение задних вытяжных вентиляторов существенно улучшает температурный режим;
   - все задние вентиляторы должны ориентировать воздушный поток в одну сторону, лучше всего - наружу;
   - применение втяжных фронтальных вентиляторов для вертикальных корпусов практически бесполезно (за редкими исключениями);
   - использовать вентиляторы диаметром менее 80…100 мм нежелательно;
   - передние и задние отверстия для циркуляции воздуха не должны перекрываться фальш-панелями и заглушками;
   - провода и кабели внутри корпуса не должны болтаться, как попало, лучше всего скрутить их в жгуты, перевязать и аккуратно уложить так, чтобы они не мешали потоку воздуха.
5. Качество исполнения. На корпусе не должно быть острых кромок; кроме того, должна обеспечиваться необходимая жесткость креплений.

Остальные параметры - цвет, дизайн и габариты - дело вкуса и личных предпочтений.

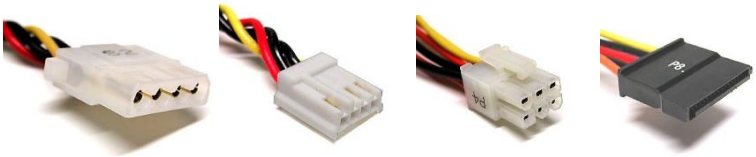
Блок питания

Источник (или блок) питания обычно смонтирован и поставляется вместе с корпусом системного блока, для которого он предназначен. Мощность источника питания компьютера должна полностью и даже с некоторым запасом обеспечивать энергопотребление всех подключенных к нему устройств. Чем больше устройств может быть установлено в системный блок, тем большую мощность должен иметь блок питания. Блоки питания обеспечивают следующие выходные напряжения: + 3,3 V (в блоках питания стандарта AT не используется); +/- 5 V; +/- 12 V.

Рис.6.

С помощью кабелей практически все устройства в компьютере подсоединяются к блоку питания, а блок - к розетке. Любой кабель включает в себя соединители (разъемы), находящиеся на концах кабеля, и изолированные друг от друга проводники, тем или иным образом соединяющие эти разъемы. Провода могут быть заключены в металлическую оболочку (экран), а кабель может быть покрыт пластиковой защитной оболочкой. Все кабели можно разделить на две большие группы: сигнальные кабели, предназначенные в основном для передачи информационных сигналов, и кабели питания (power cord), обеспечивающие только электропитание соответствующего устройства. Соединители (разъемы) бывают двух видов: розетки (female) и вилки (male).

Контактные выводы вилок выполнены обычно в виде штырьков, которые при соединении с однотипным разъемом (но уже розеткой) входят в соответствующие пазы ответных контактов. Контакты и в розетке, и в вилке могут быть также выполнены в виде плоских пружинных пластин. Большинство используемых разъемов сконструированы так, чтобы исключить возможность неправильного подключения.



для IDE для FDD для МП для SATA

Рис.7.

Для подключения к системной плате обычно используются два шестиконтактных разъема (реже один общий). Для питания накопителей предназначены четырехконтактные разъемы. Данные разъемы отличаются по размеру: large style и small style. Если разъемов не хватает, можно использовать специальные Y-разветвители.

Кроме этого, блоки питания ATX, помимо основного 20-контактного разъема питания для плат ATX, имеют дополнительный четырехжильный кабель для 5 и 12 В питания (рис.8), поскольку основной разъем уже не способен обеспечить повышенные требования к электропитанию системной платы.

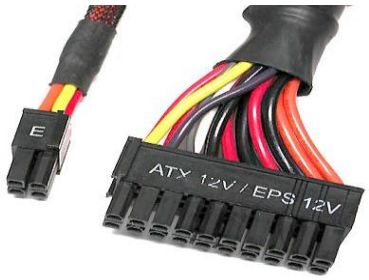


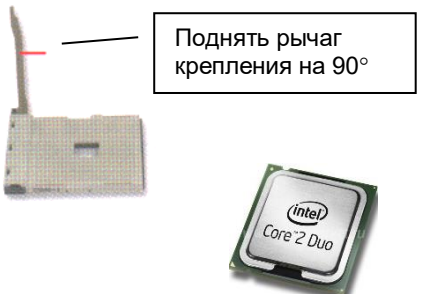
Рис.8.

В блоках питания системы ATX используются специальные управляющие сигналы Power_On и 5V_Standby. Первый из них обеспечивает включение системы программным путем, в том числе и от клавиатуры, а второй - предоставляет возможность поддерживать систему в "спящем" режиме вместо ее полного выключения. ATX-блоки, помимо перечисленных выше номиналов, вырабатывают также напряжение 3,3 V и подключаются к материнской плате через 20-контактный разъем, исключающий возможность неправильной установки. Кроме того, ATX-блоки, как правило, не имеют механического выключателя. Будучи подключенными к электрической сети, они находятся в состоянии пониженного потребления (standby), из которого могут быть включены по нажатию электронного выключателя на корпусе, либо по программной команде в ответ на какое-либо внешнее событие. Например, это может быть команда по сети (функция называется wake on LAN) или телефонный звонок, принятый и обработанный модемом (функция называется wake on Modem) - включаются данные функции из BIOS материнской платы. Выключение в состояние standby также может быть выполнено программно.

Стандартный алгоритм комплектации системного блока

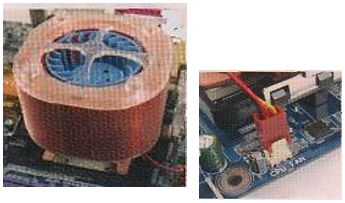
1. Установка процессора и теплоотвода на материнскую плату,
2. Установка модулей памяти,
3. Подготовка корпуса системного блока (снятие платформы для материнской платы) и установка блока питания,
4. Установка устройств IDE, SATA, флоппи-дисковода и др.,
5. Установка системной платы на платформу, крепление платформы к корпусу системного блока,
6. Установка плат расширения,
7. Подключение индикаторов и питания
8. Подключение разъемов корпуса

1.Установка процессора



1. На один из углов процессора нанесена метка в виде маленького золотого треугольника. Разместить процессор так, чтобы угол с меткой располагался рядом с рычагом крепления процессора.
2. Аккуратно установить процессор в разъем.
3. Убедится, что контакты процессора вошли в гнезда разъема.
4. Аккуратно нажать пальцем на середину процессора и опустить рычаг крепления в исходное положение.

2. Установка теплоотвода процессора



1. Перед установкой теплоотвода нанести на поверхность процессора равномерный слой термопасты.
2. Установить все компоненты теплоотвода (подробные инструкции обычно приведены в руководстве по установке теплоотвода).
3. Подключить провод питания теплоотвода к разъему CPU-FAN на системной плате.
4. Правильно установленный теплоотвод предотвращает перегрев процессора.

1. Установка модулей памяти

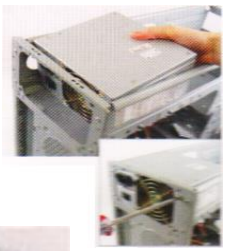
1. Убедится, что устанавливаемые модули памяти поддерживаются данной системной платой.
2. Оттянуть фиксаторы на концах разъема и удерживая модуль памяти за края вставить его вразъем до упора
3. Если модуль памяти установлен правильно, фиксаторы надежно закрепят его в разъеме.

Модуль памяти имеет выемку, благодаря чему он может быть установлены в разъем только в одном положении

4. Подготовка корпуса и установка блока питания

1. Перед установкой блока питания снять обе боковые стенки и крышку корпуса системного блока

2. Установить блок питания и прикрепить его к корпусу системного блока винтами



Последовательность действий по установке и местоположение блока питания для разных корпусов могут быть различными

5. Установка устройств IDE, SATA, FDD

Установка оптических накопителей DVD, CD

1. Снять переднюю панель корпуса.
2. Удалите заглушку отсека 5.25".
3. Установить оптический дисковод в отсек 5.25' и закрепить его винтами.

Установка флоппи-дисковода

4. Установить флоппи-дисковод в отсек 3.5" и закрепить его винтами.

Установка жестких дисков IDE, Serial ATA

5. Установить жесткий диск в отсек и закрепить его винтами

6. Установка системной платы

1. Снять одну боковую стенку корпуса (если она не снята)
2. Снять с корпуса панель вода/вывода и установить на ее место панель, поставляющуюся в комплекте с системной платой.
3. Установить системную плату в корпус: вставить ее в отверстие панели ввода-вывода и прикрепите панель к корпусу винтами.
4. Закрепить системную плату в корпусе винтами.
5. Если панель ввода-вывода корпуса несовместима с системной платой, то снять ее и заменить на панель, поставляющуюся в комплекте с системной платой

7. Установка плат расширения

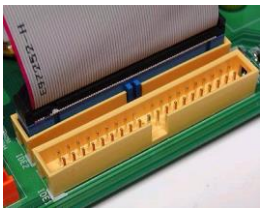
Перед установкой платы расширения убедиться, что системная плата поддерживает требуемый тип плат расширения

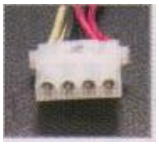
1. Снять металлическую крышку, закрывающую разъем, и установить плату расширения в разъем.
2. Убедившись, что плата вошла в разъем до упора, закрепить ее с помощью винтов

Обязательно убедитесь, что плата вошла в разъем до упора

8. Подключение накопителей

a) Подключить информационные кабели:
b) 1.К оптическому накопителю – шлейф IDE;
c) 2. К жесткому диску IDE - шлейф IDE;
d) 3. К жесткому диску SATA – кабель Serial ATA;
e) 4.К флоппи дисководу – шлейф флоппи – дисковода
f) Кабели, провода питания и аудиокабели могут вставляться в разъемы только в одном положении. Первый контакт обычно помечен красной полоской на шлейфе.
g) Рекомендуется подключать жесткий диск IDE к разъему IDE_1, а оптический накопитель - к разъему IDE_2.
h) К одному разъему IDE можно подключить два IDE-устройства. Перед установкой проверить настройки IDE-устройств – Master или Slave.
i) Если в компьютере установлены несколько жестких дисков IDE, войти в меню настроек BIOS и установить последовательность загрузки с IDE устройств.
j) Подключить кабели питания





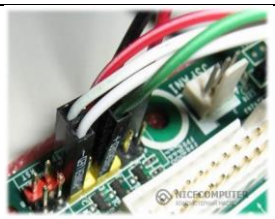
2. Подключение питания материнской платы

3. Разъем питания ATX: подключается к разъему ATX на системной плате

4. Разъем питания ATX_12B: подключается к разъему ATX_12V на системной плате

10. Подключение индикаторов

1. Подключить индикаторы на передней панели корпуса к разъему F_PANEL на системной плате
2. Подключите индикаторы питания, жесткого диска, динамика, питания компьютера и т.п. к разъему F_PANEL на системной плате
5. Наличие и расположение индикаторов в корпусах разных производителей может различаться



11. Установка передней панели

6. Установите на место переднюю панель корпуса и закрепите ее винтами

12. Подключение разъемов корпуса

2. Выполните практические задания

1. Ознакомиться с теоретическим материалом и требованиями к отчету по лабораторной работе.
2. Собрать системный блок из предложенных комплектующих.

Контрольные вопросы

1. Какие бывают корпуса для системных блоков
2. Какие бывают блоки питания.
3. Коков процесс сборки системного блока.

3.2. Тестовые задания

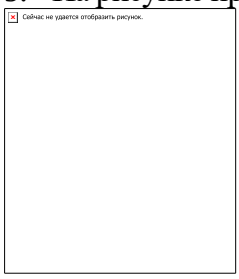
Тест по теме 1

1. Комплекс технических и программных средств, предназначенный для автоматизации подготовки и решения задач пользователей называется:
    a. Вычислительной машиной,
    b. Вычислительной системой,
    c. Персональным компьютером,
    d. Арифмометром
2. Совокупность взаимосвязанных и взаимодействующих процессоров или вычислительных машин, периферийного оборудования и программного обеспечения, предназначенная для подготовки и решения задач пользователей называется:
    a. Вычислительной машиной,
    b. Вычислительной системой,
    c. Персональным компьютером.
    d. Суперкомпьютером
3. Физическая структура вычислительной машины называется
    a. Организацией,
    b. Архитектурой
    c. Схемой,
    d. Устройством
4. Логическое построение вычислительной машины называется
    a. Организацией,
    b. Архитектурой
    c. Схемой,
    d. Устройством
5. При описании истории развития вычислительной техники используют следующие подходы:
    a. Технологический
    b. Исторический
    c. Хронологический
    d. Временной

6. Нестрогая классификация вычислительных машин по степени развития их аппаратных, а в последнее время, и программных средств называется:

a. Видом
b. Классом
c. Поколением
d. Типом

7. Простейшая часть вычислительной машины, выполняющая операции над двоичными цифрами (битами) и имеющая регулярную структуру называется:
   a. Узлом
   b. Устройством
   c. Блоком
   d. Элементом
8. Типовыми узлами вычислительной машины являются:
   a. Регистры
   b. Процессоры
   c. Микроконтроллеры
   d. Счетчики
9. На рисунке представлена схема ….



   a. Шифратора
   b. Дешифратора
   c. Регистра
   d. Триггера
10. Узел или устройство, предназначенное для временного хранения обрабатываемой, или управляющей информации называется:
    a. Регистром
    b. Счетчиком
    c. Коллектором
    d. Аккумулятором
11. Количество триггеров и вспомогательных схем, обеспечивающих выполнение некоторых элементарных операций в регистре, определяет его:
    a. Быстродействие,
    b. Синхронизацию,
    c. Разрядность
    d. Работоспособность
12. Электронная схема, предназначенная для подсчета числа сигналов, поступающих на его счетный вход называется:
    a. Регистром
    b. Счетчиком
    c. Коллектором
    d. Аккумулятором
13. Сущность фон-неймановской концепции вычислительной машины можно свести к четырем принципам:
    a. Двоичного кодирования;
    b. Разделения памяти.
    c. Программного управления;
    d. Однородности памяти;

e. Адресности,

14. Достоинством микропроцессорной системы, построенной по принципу «жесткой логики» является:
    a. Отсутствие аппаратной избыточности
    b. Максимальная производительность
    c. Не возможность программирования
    d. Длительный срок службы
15. Недостатком микропроцессорной системы, построенной по принципу «мягкой логики» является:
    a. Узкая специализация
    b. Возможность программного управления
    c. Аппаратная избыточность
    d. Максимальная производительность
16. Реакцию микропроцессорной системы на внешнее воздействие можно организовать с помощью следующих методов:
    a. Программного
    b. Опроса флага
    c. Прерывания
    d. Прямого доступа к памяти
17. Условно в классификации микропроцессорных систем выделяют следующие виды:
    a. Микроконтроллеры, контроллеры, компьютеры, микрокомпьютеры
    b. Компьютеры, ЭВМ, супер ЭВМ
    c. Карманные, портативные, настольные, большие
    d. SIMD, SISD, MISD, MIMD
18. Первый в мире 4-разрядный микропроцессор 4004, предназначенный для использования в микрокалькуляторах был выпущен фирмой
    a. AMD
    b. INTEL
    c. ASUS
    d. SIS
19. Основными характеристиками центрального процессора являются:
    a. Тактовая частота
    b. Разрядность РОН
    c. Наличие КЭШа
    d. Количество ядер
    e. Технологический процесс
    f. Система команд
20. Какой регистр ЦП используется для хранения адреса команды:
    a. SI,
    b. IP,
    c. CX,
    d. DX
21. Какой регистр ЦП используется для хранения указателя стека:
    a. SI,
    b. IP,
    c. CX,
    d. SP.
22. Какие из флагов относятся к статусным:
    a. Флаг нуля
    b. Флаг направления (уменьшение/ увеличение)
    c. Флаг знака
    d. Флаг прерывания (возможно/ невозможно)

e. Флаг четности
f. Флаг переполнения
g. Флаг трассировки
h. Флаг переноса
i. Флаг арифметического переноса

23. Центральная комплексная печатная плата, предоставляющая электронную и логическую связь между всеми устройствами, входящими в состав персонального компьютера называется:
    a. Материнской
    b. Системной
    c. Главной
    d. Основной
24. Основным компонентом материнской платы, объединяющим между собой все ключевые подсистемы является:
    a. Процессор
    b. Память
    c. Чипсет
    d. Системная шина
25. Основными функциями центрального процессора являются:
    a. Формирование управляющих команд
    b. Выполнение арифметических и логических операций
    c. Обеспечение интерфейса между всеми устройствами
    d. Управление системной шиной
26. Набор проводящих дорожек материнской платы, работающих по определенному протоколу, называется:_____ шиной
27. Адресация, использованная в команде MOV AX, [100], называется ____ прямой
28. Адресация, использованная в команде MOV AX, 100, называется ____ непосредственной
29. Адресация, использованная в команде MOV AX, BX, называется ____ регистровой
30. Авторским коллективом концепции вычислительной машины с хранимой в памяти программой являются:
    a. Джон фон Нейман
    b. Говорд Эйкен
    c. Джон Моучли
    d. Преспер Эккерт

3.3. Вопросы к зачету

1. Эволюция средств вычислительной техники: нулевое поколение ЭВМ.
2. Эволюция средств вычислительной техники: первое и второе поколение ЭВМ.
3. Эволюция средств вычислительной техники: третье и четвёртое поколение ЭВМ.
4. Эволюция средств вычислительной техники: пятое и шестое поколения ЭВМ.
5. Концепция машины с хранимой в памяти программой. Принцип Джона фон Неймана.
6. Основные компоненты ЭВМ: элемент, узел, устройство.
7. Материнская плата.
8. Центральный процессор: организация.
9. Центральный процессор: архитектура (методы адресации, форматы данных).
10. Центральный процессор: архитектура (система команд пересылки данных).
11. Центральный процессор: архитектура (система арифметических команд).
12. Центральный процессор: архитектура (система команд передачи управления).
13. Технологии увеличения производительности процессора.
14. Общие вопросы организации памяти ЭВМ.
15. Оперативная память.
16. Постоянная память.

17. Внешние запоминающие устройства на магнитных дисках.
18. Накопители на оптических дисках.
19. Флэш-накопители.
20. Обслуживание жестких дисков
21. Мониторы: виды, принцип работы, правила эксплуатации.
22. Принтеры: виды, принцип работы, правила эксплуатации.
23. Сканеры: виды, принцип работы, правила эксплуатации.
24. Клавиатуры: виды, принцип работы, правила эксплуатации.
25. Мыши: виды, принцип работы, правила эксплуатации.
26. Дополнительное периферийное оборудование: виды, принцип работы, правила эксплуатации.
27. Вычислительные системы.
28. Определение производительности компьютера и вычислительных систем
29. Сетевое оборудование для организации ЛВС.
30. Оборудование для организации глобальной сети и подключения к ней.
31. Перспективные технологии развития вычислительной техники.