Федеральное агентство по образованию

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ГОУВПО “АмГУ”

УТВЕРЖДАЮ

Зав.кафедрой ИУС

_____А. В. Бушманов

“_____”_____2007г.

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО ДИСЦИПЛИНЕ

для специальности

230102 – “Автоматизированные системы обработки информации и управления”

Составитель: С.С. Охотников

Благовещенск

2007 г.

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

С.С. Охотников
Учебно-методический комплекс по дисциплине "Организация ЭВМ и систем" для студентов очной формы обучения специальности 230102 “Автоматизированные системы обработки информации и управления”. – Благовещенск: Амурский гос. ун-т, 2007. – 50 с.

Учебно-методический комплекс предназначен для оказания помощи преподавателям и студентам очной формы обучения по дисциплине "Организация ЭВМ и систем" для студентов очной формы обучения специальности 230102 “Автоматизированные системы обработки информации и управления” и может использоваться для подготовки и проведения занятий, а также для самостоятельного изучения дисциплины.

СОДЕРЖАНИЕ

Федеральное агентство по образованию РФ

Амурский государственный университет

УТВЕРЖДАЮ

Проректор по УНР

_____ Е. С. Астапова

“_____”_____ 200___ г.

РАБОЧАЯ ПРОГРАММА

По дисциплине “Организация ЭВМ и систем”

Для специальности:

230102 – “Автоматизированные системы обработки информации и управления”

Курс	– 2
семестр	– 4
Лекции	– 36 час.
Практические занятия	– 18 час.
Лабораторные занятия	– 18 час.
Экзамен	– 4 семестр
Самостоятельная работа	– 60 час.
Всего часов	– 132 час.
Составитель:	Охотников С. С., ст. преподаватель
Факультет:	Математики и информатики
Кафедра	Информационных и управляющих систем

2006 г.

Рабочая программа дисциплины разработана на основании Государственного образовательного стандарта высшего профессионального образования для специальности 230102 – “Автоматизированные системы обработки информации и управления”.

Госстандарт:

системы счисления; арифметические операции; принцип микропрограммного управления; базовая архитектура ЭВМ; процессор; система команд; способы адресации; язык ассемблера; реализация основных механизмов (механизм подпрограмм, рекурсия, сопрограммы); система прерываний; организация ввода-вывода; прямой доступ в память; иерархия памяти, организация запоминающих устройств; тенденции и принципы развития современных ЭВМ; архитектура микропроцессорного набора; системная магистраль; программируемые интерфейсы; полупроводниковая память; арифметические процессоры; мультипроцессорные конфигурации; технология разработки микропроцессорных систем; инструментальные средства программирования и отладки; поколения вычислительных систем; архитектура вычислительной системы; микропроцессорные системы; архитектурные решения, проблемно-ориентированная архитектура; вопросы анализа и оценки эффективности вычислительной системы.

1. Цели и задачи дисциплины.

Целью дисциплины является ознакомление студентов с основными принципами организации аппаратного обеспечения ЭВМ и систем, принципами работы периферийных устройств и их взаимодействия в составе системы.

Задачи дисциплины состоят в определении места изучаемых механизмов функционирования ЭВМ среди других технических систем, учета особенностей изучаемых систем в проектировании ИС.

2. Требования к уровню освоения содержания дисциплины.

Знать: архитектуру основных типов современных ЭВМ,

терминологию в данной предметной области;

используемые в системах способы обмена информацией;

принципы построения основных периферийных устройств и их взаимодействие в составе системы;

Уметь:

с помощью программных средств организовывать управление ресурсами ЭВМ;

Иметь навыки: выбора архитектур и средств комплексирования современных ЭВМ и систем, проектирования устройств вычислительной техники, анализа работы узлов и блоков ЭВМ.

Иметь опыт: практической работы с ПЭВМ;

Иметь представление: об основных характеристиках современных ЭВМ и существующих в этой области проблемах.

3. Объем дисциплины и виды учебной работы.

Вид учебной работы	Всего часов	Семестры			
Общая трудоемкость дисциплины	132		4	5	6
Аудиторные занятия	90		90		
Лекции	36		36		
Практические занятия (ПЗ)	18		18		
Семинары (С)					
Лабораторные работы (ЛР)	18		18		
Самостоятельная работа	60		60		
Курсовой проект (работа)					
Расчетно-графические работы					
Реферат					
Вид итогового контроля			Экзамен		

4. Содержание дисциплины

4.1. Разделы дисциплины и виды занятий

№ п/п	Раздел дисциплины	Лекции	ПЗ (или С)	ЛР
1	Введение	1	1	1

2	Базовая архитектура ЭВМ	5	3	3
3	Механизмы функционирования ЭВМ	4	3	3
4	Организация памяти и ЗУ	4	1	1
5	Организация систем ЭВМ	3	1	1
6	Заключение	1		

4.2. Содержание разделов дисциплины

1. Введение

системы счисления; арифметические операции;

методы передачи данных;

2. Базовая архитектура ЭВМ

общая шина и базовая архитектура ЭВМ;

процессор; принцип микропрограммного управления;

функциональная и структурная организация процессора.

система команд. основные этапы выполнения команды.

способы адресации;

объектный код и язык ассемблера;

архитектура микропроцессорного набора;

системная магистраль;

программируемые интерфейсы;

тенденции и принципы развития современных ЭВМ;

3. Механизмы функционирования ЭВМ

механизм стека;

механизм подпрограмм, рекурсия, сопрограммы;

механизм прерываний;

механизм ввода-вывода;

механизм прямого доступа к памяти;

механизм сегментного преобразования;

механизм страничного преобразования;

механизм кэширования данных и команд;
механизмы защиты;

4.Организация памяти и ЗУ
полупроводниковая память;
СОЗУ; ДЗУПВ; контроллер ДЗУПВ;
иерархия памяти;
организация запоминающих устройств;
Внешние запоминающие устройства на магнитных носителях.
Оптические дисковые накопители.
Физическая и логическая структура дисков.
Программные средства для работы с дисками.

5. Организация систем ЭВМ
арифметические процессоры;
мультипроцессорные конфигурации;
технология разработки микропроцессорных систем;
инструментальные средства программирования и отладки;
поколения вычислительных систем;
архитектура вычислительной системы;
микропроцессорные системы;
архитектурные решения, проблемно-ориентированная архитектура;
Вычислительные системы и сети ЭВМ. Сопроцессоры. Мультипроцессорные вычислительные системы. Матричные и конвейерные вычислительные системы. Связные устройства. Модемы. Протоколы обмена.

6. Заключение.
Вопросы анализа и оценки эффективности вычислительной системы.

5. Лабораторный практикум

№ п/п	Тема	Кол-во часов
1.	Приемы работы CLI DOS	2
2.	Отладчик Debug.exe	2
3.	Программирование ввода-вывода на примере VGA	4
4.	Программирование ввода-вывода на примере UART	4
5.	Исследование функционирования стека i386	3
6.	Функции BIOS	3

6. Учебно-методическое обеспечение дисциплины.

6.1 Рекомендуемая литература.

а) основная литература:

1. Скляров В. А. Применение ПЭВМ. Кн. 1, Организация и управление ресурсами ПЭВМ. М, Высшая школа, 1992.

2. Григорьев В.Л. Микропроцессор i486. Архитектура и программирование. (в 4-х книгах). Книга 1. Программная архитектура с.346, ил.87. Книга 2. Аппаратная архитектура. Книга 3. Устройство с плавающей точкой. Книга 4. Справочник по системе команд. - М., ГРАНАЛ. 1993.- с.382, ил.54

3. Каган Б.М. Электронные вычислительные машины и системы. Уч. пособие для вузов-3-е изд., перераб. и доп. - М.: Энергоатомиздат, 1991,-592 с.

4. Фрир Дж. Построение вычислительных систем на базе перспективных микропроцессоров.: Пер. с англ. - М.: Мир, 1990 - 413 с.

б) дополнительная литература:

1. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT, AT. – М.: Финансы и статистика, 1991.
2. Мячев А.А. и др. Интерфейсы систем обработки данных: Справочник А.А. Мячев, В.Н. Степанов, В.К. Щербо; Под ред. А.А. Мячева. - М.: Р и С, 1989-416 с., ил.
3. Сопряжение датчиков и устройств ввода данных с компьютерами IBM PC: Пер. с англ./Под ред. У. Томпкинса, Дж. Уэбстера. - М,: Мир, 1992. - 592., ил.

6.2. Средства обеспечения освоения дисциплины.

Системная документация ПК.

7. Материально-техническое обеспечение дисциплины.

Компьютерный класс, оборудованный ПК на базе i386 или выше, подключенный к ЛВС университета. ОС - MS DOS 6.22

Учебно-методическая (технологическая) карта дисциплины

“Организация ЭВМ и систем”.

Номер Недели	Но-мер Темы	Вопросы, изучаемые на лекции	Занятия (номера)		Использу-емые нагляд. И метод. Пособия	Самостоятельная работа студентов		Формы Контроля
			практич. (семин.)	лаборат.		содерж.	часы	
1	2	3	4	5	6	7	8	9
1.	1	1	1	1	3.3.1			
2.	2	2		2	3.3.1	2	2	
3.	3	3	2	3	3.3.1	3	4	
4	4	4		4	3.3.2	4	4	Консультац.
4.	5	5	3	5	3.3.2	5	4	Аттестация
5.	6	6		6	3.3.2	6	4	
6.	7	7	4	7	3.3.1	7	4	
7.	8	8		8	3.3.2	8	4	
8.	9	9	5	9	3.3.2	9	4	Консультац.
9.	10	10		10	3.3.2	10	4	Аттестация
10.	11	11	6	11	3.3.2	11	4	
11.	12	12		12	3.3.2	12	4	
12.	13	13	7	13	3.3.1	13	4	
13.	14	14		14	3.3.1	14	4	Консультац.
14.	15	15	8	15	3.3.2	15	4	Аттестация
15.	16	16		16	3.3.3	16	4	
16.	17	17	9	17	3.3.3	17	2	
17.	18	18		18	3.3.2			Консультац.

СОДЕРЖАНИЕ ЛЕКЦИОННОГО КУРСА

Часть 1. Введение в дисциплину

Основные характеристики ЭВМ. Иерархия аппаратных средств ЭВМ. Общие принципы построения ЭВМ. Функции программного обеспечения. Персональные ЭВМ. Информационно-логические основы ЭВМ.

Системы счисления. Перевод целых чисел. Перевод дробных чисел. Представление информации в ЭВМ. Представление числовой информации. Представление других видов информации. Арифметические основы ЭВМ. Машинные коды. Арифметические операции над числами с фиксированной точкой. Арифметические операции над двоичными числами с плавающей точкой. Арифметические операции над двоично-десятичными кодами чисел.

Логические основы ЭВМ. Основные сведения из алгебры логики. Законы алгебры логики. Понятие о минимизации логических функций. Карты Карно. Техническая интерпретация логических функций.

Элементная база ЭВМ. Классификация элементов и узлов ЭВМ. Комбинационные схемы. Последовательностные схемы.

Последовательная передача данных. Кодирование и модуляция. Помехоустойчивые коды. Параллельная передача данных. Шина. Синхронизация.

Часть 2. Базовая архитектура ЭВМ

Общие принципы функциональной и структурной организации ЭВМ. Организация функционирования ЭВМ с магистральной архитектурой. Организация работы ЭВМ при выполнении задания пользователя. Особенности управления основной памятью ЭВМ. Отображение адресного пространства программы на основную память.

Адресная структура команд процессора и планирование ресурсов. Виртуальная память.

Система прерываний ЭВМ. Центральные устройства ЭВМ. Основная

память. Состав, устройство и принцип действия основной памяти. Размещение информации в основной памяти x86. Расширение основной памяти x86.

Центральный процессор. Структура базового микропроцессора. Система команд микропроцессора. Взаимодействие элементов при работе микропроцессора. Принцип микропрограммного управления. Система команд.

Основные этапы выполнения команды. Циклы шины. Способы адресации. Объектный код и язык ассемблера.

Классификация архитектуры микропроцессоров (МП). Методы ускорения переключения контекста. Аппаратная поддержка сохранения регистров. Использование регистров обработки быстрых прерываний. Стандартизация архитектур МП. Архитектурно-независимая спецификация программ. Технология Java.

Часть 3. Механизмы функционирования ЭВМ

Управление внешними устройствами. Принципы управления. Прямой доступ к памяти. Интерфейс системной шины. Интерфейсы внешних запоминающих устройств. Способы организации совместной работы периферийных и центральных устройств. Последовательно-параллельные интерфейсы ввода-вывода.

Аппаратная поддержка мультипрограммирования на примере x86. Регистры процессора. Привилегированные команды. Средства поддержки сегментации памяти. Виртуальное адресное пространство. Преобразование адреса. Защита данных при сегментной организации памяти.

Средства вызова процедур и задач. Вызов процедур. Вызов задачи. Жизненный цикл задачи. Механизм прерываний. Стек. Кэширование в процессорах x86. Буфер ассоциативной трансляции. Кэш первого уровня. Совместная работа кэшей разного уровня.

Ввод-вывод. Организация параллельной работы устройств ввода-

вывода и процессора. Прямой доступ к памяти. Отображение ввода-вывода на память. Согласование скоростей обмена. Буферизация и кэширование. Разделение устройств и данных между процессами.

Часть 4. Организация памяти и ЗУ

Полупроводниковая память. Статическая и динамическая память. Контроллер ДЗУПВ. Иерархия памяти. Организация запоминающих устройств.

Внешние запоминающие устройства (ВЗУ). ВЗУ на гибких магнитных дисках. Накопитель на жестком магнитном диске. Стриммер. Оптические ЗУ. Флеш-память.

Часть 5. Организация систем ЭВМ

Мультипроцессорные конфигурации. Суперскалярные и мультискалярные процессоры. Архитектура суперскалярных процессоров. Предварительная выборка команд и предсказание переходов. Декодирование команд, переименование ресурсов и диспетчеризация. Исполнение команд. Работа с памятью. Завершение выполнения команды. Направления развития суперскалярной архитектуры.

Мультискалярные процессоры. Мультискалярная модель выполнения программы. Мультискалярные программы. Мультискалярные аппаратные средства. Направления развития мультискалярной архитектуры.

Классификация вычислительных систем. Симметричные мультипроцессорные структуры. Архитектура мультипроцессорной системы. Комплексирование в вычислительных системах. Типовые структуры вычислительных систем.

Транспьютеры. Основы архитектуры. Диспетчеризация процессов. Передача данных по линку. Ожидание сигнала от блока событий. Ожидание сигнала от таймера. Группировка команд. Конвейер процессора.

Кластеры. Типовая структура. Универсальные и специализированные

кластеры. Супер-ЭВМ. Программные средства поддержки кластерных вычислений. Распределенные хранение и обработка данных.

Классификация сетей ЭВМ. Управление взаимодействием прикладных процессов. Протоколы передачи данных. Управление доступом к передающей среде. Коммутация в сетях.

Маршрутизация пакетов. Типы и характеристики ОВС. Режимы асинхронной передачи. Управление локальными сетями.

Глобальные вычислительные сети. Протоколы обмена данными в сетях. Системы сетевых коммуникаций. Модель взаимодействия открытых систем. Перспективы развития ЭВМ и телекоммуникационных вычислительных сетей.

Часть 6. Заключение

Вопросы анализа и оценки эффективности вычислительной системы. Пиковая (техническая) производительность МП. Реальная производительность вычислительной системы. Способы измерения. Пакет тестовых программ SPEC-95.

РЕКОМЕНДАЦИИ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа 1. Приемы работы CLI DOS (2 часа)

Эмуляция режима DOS реализована в ОС Win2k в режиме командной строки интерпретатором cmd.exe. Запустив на выполнение с cmd с ключем /? , получаем помощь по использованию (подробное разъяснение и примеры в утилите help и GUI-системе помощи Win2k):

```
CMD [/A | /U] [/Q] [/D] [/E:ON | /E:OFF] [/F:ON | /F:OFF] [/V:ON | /V:OFF]
    [[/S] [/C | /K] строка]

/C      Выполнение указанной команды (строки) с последующим завершением.
/K      Выполнение указанной команды (строки) без последующего завершения.
/S      Изменение поведения после /C или /K (см. ниже)
/Q      Отключение режима вывода команд на экран (ECHO).
/D      Отключение выполнения команд AutoRun из реестра (см. ниже)
/A      Вывод результатов выполнения команд в формате ANSI.
/U      Вывод результатов выполнения команд в формате UNICODE.
/T:цв   Выбор цвета текста/фона (более подробно см. COLOR /?)
/E:ON   Разрешение расширений команд (см. ниже)
/E:OFF  Запрет расширений команд (см. ниже)
/F:ON   Разрешение символов завершения имен файлов и папок (см. ниже)
/F:OFF  Запрет символов завершения имен файлов и папок (см. ниже)
/V:ON   Разрешение отложенного расширения переменных среды с применением
        символа '!' в качестве разделителя. Например, /V:ON разрешает
        использовать !var! в качестве расширения переменной var во время
        выполнения. Синтаксис var служит для расширения переменных
        при вводе, что приводит к совсем другим результатам внутри
        цикла FOR.
/V:OFF  Запрет отложенного расширения переменных среды.

Чтобы указать в одной строке несколько команд, следует разделить их
символами '&&' и заключить в кавычки. Кроме того, из соображений
совместимости, /X означает то же, что и /E:ON, /Y то же, что и /E:OFF
и /R то же, что и /C.  Все прочие ключи командной строки игнорируются.

Если указаны ключи /C или /K, то остальная часть командной строки после
такого ключа обрабатывается как командная строка, а обработка символов
кавычек (") ведется по следующим правилам:
```

1. Если выполняются все перечисленные ниже условия, то символы кавычек в командной строке сохраняются:

   - ключ /S отсутствует
   - есть ровно два символа кавычек
   - между ними нет других специальных символов, как то: &<>()@^|
   - между ними имеются один или несколько пробелов
   - строка, заключенная в кавычки, является именем исполнимого файла.

2. В противном случае, проверяется первый символ, и если он является символом кавычек, то он удаляется, также удаляется последний символ кавычек в командной строке, а весь текст после этого последнего символа кавычек сохраняется.

Задание 1. Проверить действие указанных ключей (кроме /D) на командах `dir *.*` и `echo "Str"`. Прокомментировать действие первых трех ключей.

Если ключ /D НЕ УКАЗАН в командной строке, то при запуске CMD.EXE выполняется проверка значений переменных REG_SZ или REG_EXPAND_SZ для следующих разделов системного реестра:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\AutoRun

и/или

HKEY_CURRENT_USER\Software\Microsoft\Command Processor\AutoRun

и если одна из них или обе они присутствуют, то сначала выполняются они.

По умолчанию расширенная обработка команд включена. Чтобы запретить расширенную обработку для конкретного вызова, используется ключ /E:OFF Можно включить или отключить расширенную обработку команд для всех вызовов CMD.EXE на данном компьютере или для данного пользователя, с помощью REGEDT32.EXE задав значения REG_DWORD в системном реестре для следующих разделов:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\EnableExtensions

и/или

HKEY_CURRENT_USER\Software\Microsoft\Command Processor\EnableExtensions

равными 0x1 или 0x0. Параметры пользователя перекрывают параметры компьютера.

```
Ключи командной строки, в свою очередь, перекрывают параметры реестра.

При расширенной обработке команд изменения и/или добавления затрагивают
следующие команды:

    DEL или ERASE
    COLOR
    CD или CHDIR
    MD или MKDIR
    PROMPT
    PUSHD
    POPD
    SET
    SETLOCAL
    ENDLOCAL
    IF
    FOR
    CALL
    SHIFT
    GOTO
    START (изменен также вызов внешних команд)
    ASSOC
    FTYPE

Для получения более подробных сведений введите "имяКоманды /?".
```

Задание 2. Рассмотреть расширенный режим использования команд COLOR, PROMPT, IF, FOR. Прокомментировать отличие функциональности данных команд от стандартного режима.

```
Отложенное расширение переменных среды НЕ ВЛЮЧЕНО по умолчанию. Можно
включить или отключить отложенное расширение переменных среды для
конкретного вызова CMD.EXE с помощью ключей /V:ON или /V:OFF.
Можно включить или отключить отложенное расширение переменных среды для
всех вызовов CMD.EXE на данном компьютере или для данного пользователя,
с помощью REGEDT32.EXE задав значения REG_DWORD в системном реестре
для следующих разделов:

    HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\DelayedExpansion

        и/или

    HKEY_CURRENT_USER\Software\Microsoft\Command Processor\DelayedExpansion

равными 0x1 или 0x0. Параметры пользователя перекрывают параметры компьютера.
```

Ключи командной строки, в свою очередь, перекрывают параметры реестра.

Если отложенное расширение переменных среды включено, то символ '!' (восклицательный знак) может использоваться для замены текущего значения переменной среды во время выполнения.

Завершение имен файлов и папок НЕ ВЛЮЧЕНО по умолчанию. Можно включить или отключить завершение имен файлов и папок для конкретного вызова CMD.EXE с помощью ключей /F:ON или /F:OFF. Можно включить или отключить отложенное расширение переменных среды для всех вызовов CMD.EXE на данном компьютере или для данного пользователя, с помощью REGEDT32.EXE задав значения REG_DWORD в системном реестре для следующих разделов:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\CompletionChar
HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\PathCompletionChar

и/или

HKEY_CURRENT_USER\Software\Microsoft\Command Processor\CompletionChar
HKEY_CURRENT_USER\Software\Microsoft\Command Processor\PathCompletionChar

установив их шестнадцатеричные значения равными коду управляющего символа, используемого для конкретной функции (например, 0x4 для Ctrl-D или 0x6 для Ctrl-F). Параметры пользователя перекрывают параметры компьютера. Ключи командной строки, в свою очередь, перекрывают параметры реестра.

Если завершение включено с помощью ключа командной строки /F:ON, то используются два управляющих символа: Ctrl-D для имен папок и Ctrl-F для имен файлов. Чтобы отключить конкретный символ завершения в системном реестре, в качестве кода специального символа используется шестнадцатеричное значение символа пробела (0x20).

Завершение вызывается при вводе одного из этих двух специальных символов. Функция завершения берет строку пути слева от точки ввода, дописывает к ней символ шаблона, если таковой отсутствует, а затем строит список путей, которые соответствуют полученному определению. После этого выводится первый элемент этого списка соответствующих путей. Если же список пуст, то подается звуковой сигнал и ничего не выводится. После этого повторный ввод того же самого специального символа приводит к циклическому перебору всех соответствующих путей. Нажатие клавиши <Shift> при вводе управляющего символа позволяет просматривать список путей в обратном порядке. Если изменить выведенный путь, а затем снова ввести управляющий символ, сохраненный список очищается и строится новый список путей. То же самое происходит, если переключиться с одного символа завершения на другой. Единственное отличие при использовании символа завершения файла состоит в том, что при этом для построения списка

соответствия берется и путь, и имя файла, а при использовании символа завершения папки берется только путь. Если символ завершения используется в одной из встроенных команд манипулирования папками (CD, MD или RD), то всегда подразумевается символ завершения папок.

Символ завершения правильно работает и с именами файлов, содержащими пробелы или иные специальные символы, если при этом строка соответствия заключена в кавычки. Кроме того, если сместить точку ввода влево, а затем использовать символ завершения внутри строки, оставшийся справа от точки ввода текст будет отброшен.

Домашнее задание. Изучить использование ключа /D, установив необходимые значения параметров реестра. Привести пример использования. Изучить команду "SET" , создать персональное цветовое окружение и приглашение. Сохранить в файле .cmd для использования в следующей лабораторной работе.

Лабораторная работа 2. Отладчик Debug.exe (2 часа)

Отладчик реального режима процессора x86 запускается из командной строки:

```
C:\>debug
```

Приглашение отладчика - дефис. Справочник по использованию debug вызывается командой "?" :

```
-?
assemble     A [адрес]
compare      C диапазон адрес
dump         D [диапазон]
enter        E адрес [список]
fill         F диапазон список
go           G [=адрес] [адреса]
hex          H значение1 значение2
input        I порт
load         L [адрес] [диск] [превый_сектор] [число]
move         M диапазон адрес
name         N [путь] [список_аргументов]
output       O порт байт
proceed      P [=адрес] [число]
quit         Q
register     R [регистр]
search       S диапазон список
trace        T [=адрес] [значение]
unassemble   U [диапазон]
write        W [адрес] [диск] [первый_сектор] [число]
выделение памяти EMS            XA [#число_страниц]
освобождение памяти EMS         XD [дескриптор]
сопоставление страниц EMS       XM [Lстраница] [Pстраница] [дескриптор]
вывод состояния памяти EMS      XS
```

Задание 1. Изучить синтаксис записи аргументов `[адрес]`, `[диапазон]`, `[список]`. Получить дамп области данных BIOS. Проверить на соответствие с выделенными ниже курсивом переменными BIOS IBM PC (Драйвер Help фирмы Flambeaux Software Версия 1.2. Адаптировал С.М.Абель):

Адрес	Наименование и описание
0000:0000	Таблица векторов прерываний: 256 4-байтовых адресов

0040:0000 | Область данных ROM-BIOS

———————┼——————————— См. Область данных ROM-BIOS

0050:0000 | Область данных DOS

———————┼———

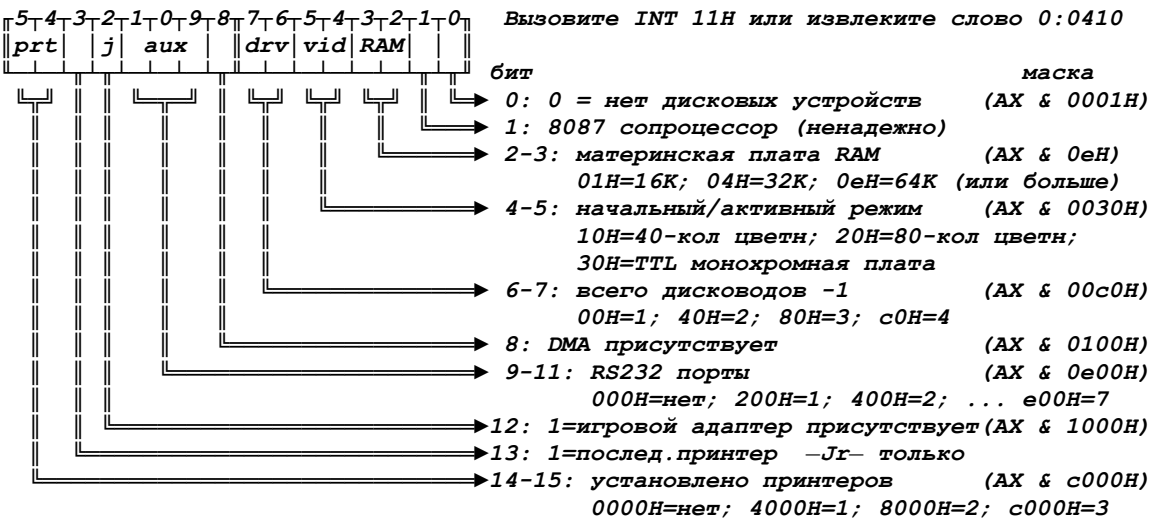
Этот раздел описывает наиболее важные переменные и области данных ROM-BIOS. Все адреса документированы и должны сохраниться в будущих версиях BIOS.

Схема приводится по возрастанию адресов, а пустые строки отмечают приблизительную функциональную группировку. Не показанные адреса зарезервированы.

> Хотя листинг BIOS показывает адреса как смещения в сегменте 40H, многие программисты предпочитают использовать смещения в абсолютном сегменте 00H. Наша таблица так и построена. Например, переменную по адресу 0:0412 можно также адресовать как 0040:0012.

Адрес	Длина	Содержимое
0:0400	***2***	***Базовый адрес порта первого адаптера RS-232 (COM1)***
0:0402	2	Порт для COM2
0:0404	2	Порт для COM3
0:0406	2	Порт для COM4
0:0408	2	Базовый адрес порта для 1-го адаптера параллельного принтера (LPT1)
0:040A	2	Порт для LPT2
0:040C	2	Порт для LPT3
0:040E	2	Порт для LPT4
0:0410	***2***	***Установленное оборудование. См. список оборудования***

Этот список описывает установленное или активное обоудование. Он возвращается прерыванием INT 11H (в AX) и хранится среди данных BIOS по адресу 0:0410.



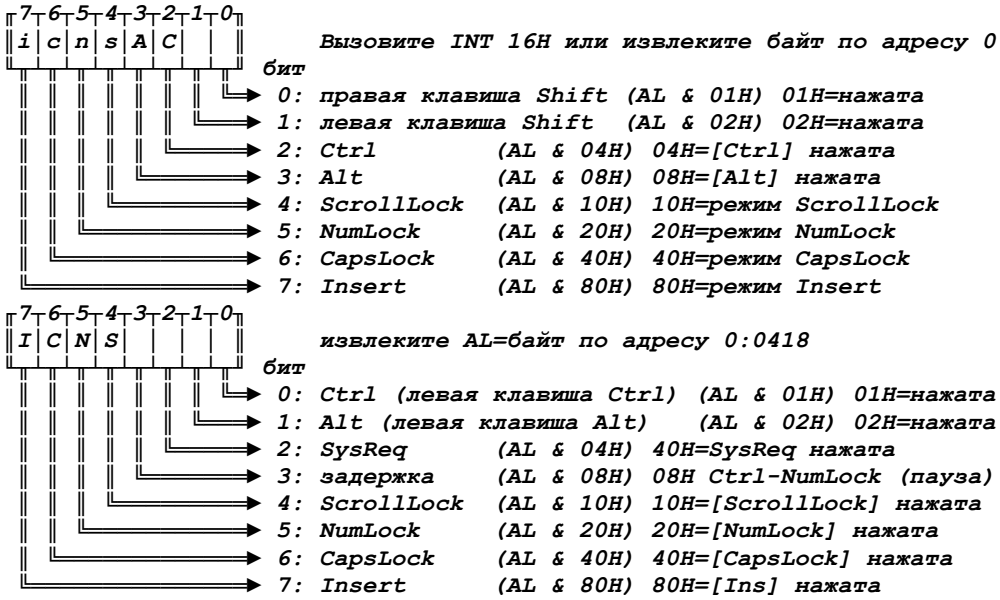
Замеч: Часто используется для проверки режима видео. Если (AX & 30H) = 30H, то адаптер монохромный——видео сегмент RAM по адресу 0b000H, иначе 0b800H.

0:0412	1	ошибки в инфракрасной связи клавиатуры PCjr
0:0413	2	общая память в К-байтах (то же, что в AX после INT 12H)

0:0415 2 рабочее поле для тестов изготовителя

0:0417 2 биты состояния клавиатуры. См. флаги клавиатуры

Два байта с адресами 0:0417 и 0:0418 содержат состояние служебных клавиш и другую полезную информацию. Прерывание INT 16H возвращает первый байт в AL.



Замеч: Биты 0-2 байта 0:0418 определены только для 101-клавишной клавиатуры. Бит 2 используется в PCjr для индикации ситуации "key click".

Некоторые старые программы изменяют биты NumLock и CapsLock (по адресу 0:0417), чтобы установить нужный статус. Это не следует делать, потому что современные клавиатуры снабжены световыми индикаторами, которые тем самым перестанут отражать текущий статус.

0:0419 1 текущее (накопленное) значение ввода Alt+цифровая клавиатура. Обычно 0. Когда [Alt] отпускается, значение пересылается в буфер клавиатуры.

0:041a 2 адрес "головы" буфера клавиатуры (символ по этому адресу следующий)

0:041c 2 адрес "хвоста" буфера клавиатуры

0:041e 20H буфер клавиатуры. Здесь BIOS хранит ввод клавиш (голова и хвост указывают на адреса от 041eH до 043dH включительно).

0:043e 1 требуется рекалибрация флоппи-дисковода (бит 0=A, бит 1=B, etc.)

0:043f 1 мотор дискеты включен (бит 0=drive A, бит 1=B, etc.)

0:0440 1 время до выкл. мотора. INT 08H выключает мотор, когда здесь 0.

0:0441 1 код ошибки дискеты. То же, что статус, возвращаемый по INT 13H

0:0442 7 область информации состояния контроллера дискет

0:0449 1 текущий видео режим. См. видео режимы и INT 10H

0:044a 2 ширина экрана в текстовых колонках

0:044c 2 длина (в байтах) видео области (regen size)

0:044e 2 смещение в видео сегменте активной страницы видео памяти

0:0450 10H положение курсора (8 2-байтовых; младший=колонка, старший=строка)

0:0460 2 размер (форма) курсора. Младший=посл.строка; старший=нач.строка.

0:0462 1 номер текущей активной видео страницы

0:0463 2 адрес порта для чипа 6845 видеоконтроллера См. Порты в/в CGA

0:0465 1 текущее значение CRT_MODE контроллера 6845 (регистр порта 3x8H)

0:0466 1 текущее значение CRT_PALETTE контроллера 6845 (регистр порта 3x9H)

```
0:0467    5  область данных кассеты или область данных POST
```
0:046c   4 счетчик тиков таймера (55мсек-единиц с момента сброса)

Задание 2. Проверить количество тиков таймера в секунду, используя команды "D" и "M" с 10-секундным интервалом по системным часам Win2k. Сравнить с приведенным выше значением.

```
0:0470    1  флаг переполнения таймера
0:0471    1  флаг Ctrl-Break. Бит 7=1 при нажатии. Никогда не сбрасывается,
             если вы не сделаете это сами.
0:0472    2  1234H означает, что работает перезагрузка Ctrl-Alt-Del. BIOS
             проверяет это, чтобы избежать "холодной загрузки" и программы POST
0:0474    4 управляющая область дискеты —Jr— или твердого диска ▌AT▐
0:0478    4  значения таймаутов принтеров
0:047c    4  значения таймаутов RS-232
0:0480    2 начальный адрес смещения буфера клавиатуры ▌AT▐ (обычно 001eH)
0:0482    2 конечный адрес (обычно 003eH)
0:0484    1  ◄EGA► число символьных строк -1  (максим. действительный # строки)
0:0485    2  ◄EGA► байт на символ (скан-строк/символ в активном режиме)
0:0487    1  ◄EGA► разные флаги (EGA активен, исп. монохромный дисплей и т.п.)
0:0488    1  ◄EGA► разные флаги (средства, переключатели)  См. ◄EGA► области
0:0490    1  ▌AT▐ биты состояния диска 0 (исп. для 1.2M дисков)
0:0491    1                  для диска 1
0:0492    1  ▌AT▐ флаг запуска операции для устройства 0
0:0493    1                             для устройства 1
0:0494    1  ▌AT▐ текущий номер цилиндра для устройства 0
0:0495    1                              для устройства 1
0:0497    1  ▌AT▐ additional keyboard flag for LED 'key lock' display
0:0498    4  ▌AT▐ адрес 8-битового флага ожидания пользователя. См. INT 15H
0:049c    4  ▌AT▐ число микросекунд ожидания пользователя
0:04a0    1  ▌AT▐ флаг активности ожидания. 1=busy; 80H=posted; 0=acknowledged
0:04a1    7  ▌AT▐ зарезервировано для сетевых адаптеров
0:04a8    4  ◄EGA► SAVE_PTR адрес таблицы указателей.      См. ◄EGA► области
0:04f0   10H Область связи между приложениями. Программы могут использовать эту
             область для записи статуса и т.п.
0:0500    1  статус печати экрана.
             00H=ok; 01H=печать активна; 0ffH=ошибка при печати экрана
0:0504    1  статус фантомных флоппи-дисков. 01H=диск A действует как диск B.
0:0510   11H используется интерпретатором BASIC
0:0530    3  используется командой MODE
```
f000:fff0 5 инструкция FAR JMP на начало POST
***   (после холодного старта 8088/86/286/386 передает управление сюда)***
f000:fff5 8 дата издания ROM-BIOS в коде ASCII ("04/24/81" в первых PC)

```
f000:fffc 2  (не используется)
```

```
f000:fffe 1  код типа компьютера IBM (не совсем надежно).
             0ffH = оригинальный PC        <=== может быть также 0feH
             0feH = XT или Portable PC
             0fdH = PCjr
             0fcH = AT
             0f9H = Convertible PC
```

Домашнее задание. Изучить флаги клавиатуры. Используя перенаправление ввода-вывода, записать командный файл для управления индикаторами клавиатуры. Создать три дампа буфера клавиатуры, демонстрирующих заполнение циклического буфера. Дать пояснения.

Лабораторная работа 3. Программирование ввода-вывода на примере VGA (4 часа)

Регистры видеоадаптера VGA, совместимые с EGA, доступны для чтения/записи через команды "I" и "O" отладчика. Часть работы может быть выполнена с помощью встроенного ассемблера debug (-a 100; исполнение -g=100):

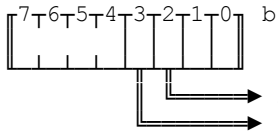
```
EGA совместим вверх с CGA при доступе через BIOS. ◄EGA► намного сложнее аппарат-
но, но он эмулирует много регистров/операций CGA. См. Порты в/в CGA .   EGA не
допускает "снега", т. е. не нужна проверка на обратный ход при записи на экран.
      • EGA #1: порты 3c0H-3dfH;       • EGA #2: порты 2c0H-2dfH
The EGA can generate interrupt level 2 (IRQ 2, INT 0aH) during the overscan at
the start of the vertical retrace. The ▌AT▐ uses IRQ 2 as a cascade request for
a whole class of IRQs (IRQ 8-15).  Techniques for handling multiple hardware
interrupts through the same IRQ are covered in the AT Tech Ref 6183355 (1986).
Note:  BIOS stores mirrors of the current CRTC port values and other EGA infor-
  mation in ◄EGA► Data Areas   To change values, you should read the appropriate
  variable, set the bit, perform OUT, and store the new value in the variable.
Port  Description
════  ═══════════════════════════════════════════════════════════════════════
3c0H  Write: Attribute Controller (ATC) address and data port
             Perform  IN al,3daH      ;(or IN 3baH) to force address mode
                      OUT 3c0H,reg_no ;select ATC register (toggle to data mode)
                      OUT 3c0H,value  ;store a value in an ATC register
             Note: bits 0-4: select an ATC register
                   bit 5:     1=enable display; 0=set register After selecting a
register, perform another OUT 3c0H,data as below:
           ATC Reg Data description
           ═══════ ═══════════════════════════════════════════════════════════
           00-0fH  palette registers: select colors for this attribute
                   ┌7┬6┬5┬4┬3┬2┬1┬0┐ bit
                   │ │ │r│g│b│R│G│B│  0: Red     Note: R,G,B are 2/3-intensity
                   └─┴─┴─┴─┴─┴─┴─┴─┘  1: Green          r,g,b are 1/3-intensity
                                      2: Blue
                                      3: blue' (also MDA-emulation underline)
                                      4: green' (also CGA-emulation intensity)
                                      5: red'
            10H    mode control
                   ┌7┬6┬5┬4┬3┬2┬1┬0┐ bit
                   │ │ │ │ │ │ │ │ │  0: 1=graphics modes, 0=text
                   └─┴─┴─┴─┴─┴─┴─┴─┘  1: 1=MDA attributes, 0=color attributes
                                      2: 1=9th dot is 8th dot for line/box chars
                                         0=use bkgd color as 9th dot of char
                                      3: 1=enable blink, 0=allow 4-bit bkgd
              11H    select overscan (border) color.  Same rgbRGB as palette regs 12H
enable color planes
                   ┌7┬6┬5┬4┬3┬2┬1┬0┐ bit
                   │ │ │ │ │ │ │ │ │  0-3: set bits to enable color planes 0-3
                   └─┴─┴─┴─┴─┴─┴─┴─┘  4-5: video status to multiplex
                                           00=R&B, 01=b&G, 10=r&g, 11=unused
            13H   horizontal pixel panning
                  Number of dots to shift data left.  Bits 0-3 valid (0-0fH)
3c2H  Write: miscellaneous control functions
      ┌7┬6┬5┬4┬3┬2┬1┬0┐ bit
      │ │ │ │ │ │ │ │ │  0: 1=select 3BxH (emulate MDA); 0=select 3DxH (CGA)
      └─┴─┴─┴─┴─┴─┴─┴─┘  1: 1=enable RAM; 0=disable RAM
                         2-3: Dot clock: 00=14mHz; 01=16mHz; 10=external; 11=res
                         4: 1=select feature output; 0=select internal
                         5: Page bit for odd/even mode                          s
                         6: Horizontal retrace polarity: 1=negative, 0=positive
                         7: Vertical retrace polarity: 1=negative, 0=positive
```

3c4H Write: sequencer address register
3c5H Write: sequencer data register

Perform OUT 3c4H,reg_no; then OUT 3c5H,data Seq Reg Data description

00 sequencer reset.
Bits 0-1 indicate asynchronous/synchronous reset.

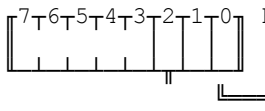
01 clocking mode



02 map mask: bits 0-3 enable writes to bit planes 0-3

03 character map select:
bits 0-1 select Map B (for when attribute bit 3=0)
bits 2-3 select Map A (for when attribute bit 3=1)

04 memory mode



3caH Write: graphics 2 position (must be 0 for EGA)
3ccH Write: graphics 1 position (must be 1 for EGA)
3ceH Write: graphics 1 and 2 address register
3cfH Write: graphics controller data register
Perform OUT 3ceH,reg_no; then OUT 3cfH,data

GDC Reg Data description

00 set/reset. bits 0-3 select planes for write mode 00

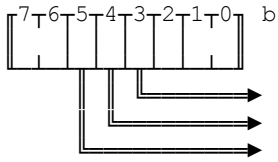
01 enable set/reset

02 color compare. Bits 0-3 select color for read mode 01

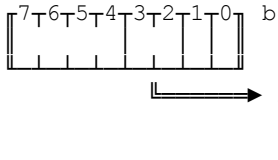
03 data rotate and function select for write mode 00
bits 0-2: set rotate count for write mode 00
3-4: fn select for write modes 00 and 02
00=no change; 01=AND; 10=OR; 11=XOR

04 read map select. Bits 0-2 select map number for read mode 00

05 mode register



06 miscellaneous graphics control



07 color masking disable
bits 0-3 disable planes from compare logic in read mode 01

08 bit mask. Bits 0-7 select bits to be masked in all planes

3b4H or 3d4H CRT controller address
3b5H or 3d5H CRT controller internal registers
Note: Port 3c0H bit 0 selects which port address-pair to use.
(3bxH is normal for MDA; 3dxH is normal for CGA).
Perform OUT 3x4H,reg_no; then OUT 3x5H,data

CRTC Rg Data description

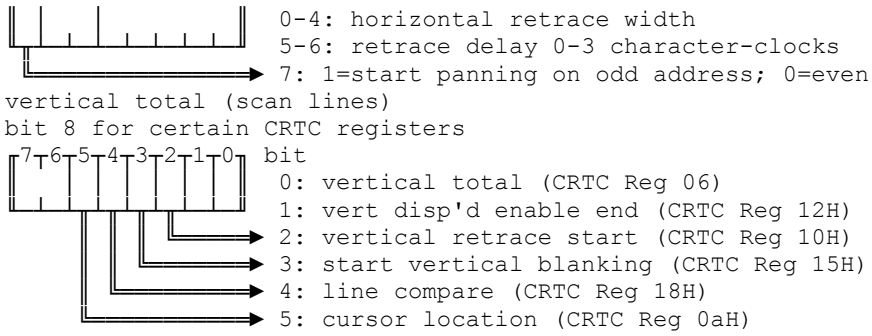
00-11H See Video Parameter Table and CGA I/O Ports Exceptions:

02 start horizontal blanking (character count)

03 end horizontal blanking
bits 0-4 are blank width; bits 5-6 are skew enable (0 to 3)

04 start horizontal retrace (character position)

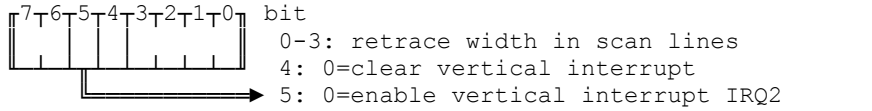
05 end horizontal retrace
7 6 5 4 3 2 1 0 bit



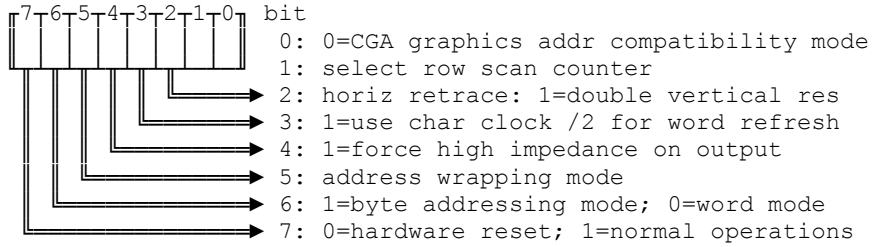

```
            06      vertical total (scan lines)
            07      bit 8 for certain CRTC registers
```



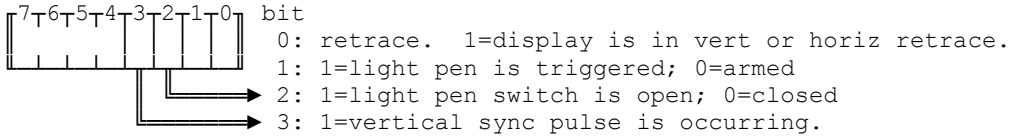
```
            08      preset row scan first scan line after vertical retrace
            10H     Write: vertical retrace start
            11H     Write: vertical retrace end
```




```
            12H     vertical displayed end (scan line)
            13H     vertical displayed adjustment
            14H     underline location.  Bits 0-5 are scan line number.
            15H     start vertical blanking (scan line)
            16H     end vertical blanking
            17H     mode control
```




```
            18H     line compare (scan line)  Used for split screen operations.
3baH or 3daH  Write: bits 0-1 control Feature Control outputs FC0 and FC1
               Note: Port 3c0H bit 0 selects which port address to use.
      Read:  status bits are same as MDA (port 3baH) or CGA (port 3daH)
```




```
3bdH or 3dbH  Write: light pen latch reset.  Any OUT clears the latch.
3bcH or 3dcH  Write: light pen latch set.  Any OUT arms the latch.
```

Задание 1. Осуществить плавную прокрутку экрана текстового режима - регистр PRSR index 8. Управление осуществляется битами 0-4. Записать командный файл, создающий эффект "прыгающего окна". Обьяснить назначение битов 5-6.

Задание 2. Осуществить горизонтальное параномирование экрана текстового режима - регистр HPR index 13. Управление осуществляется битами 0-3. Написать программу, осуществляющую параномирование. Обьяснить отличия работы программы в различных текстовых режимах.

Далее детализированы функции INT 10H стандартного видеосервиса ROM-BIOS.

```
См.  ◄EGA► BIOS сервис о дополнительных функциях, предоставляемых EGA BIOS.
См.  Порты в/в EGA  и  Порты в/в CGA  об управляющей информации аппаратуры.

АН  Сервис
```

00H уст. видео режим. Очистить экран, установить поля BIOS, установить режим.
Вход: AL=режим

AL	Тип	Формат	Цвета	Адаптер	Адрес	Монитор
0	текст	40x25	16/8 полутона	CGA,EGA	b800	Composite
1	текст	40x25	16/8	CGA,EGA	b800	Comp,RGB,Enhanced
2	текст	80x25	16/8 полутона	CGA,EGA	b800	Composite
3	текст	80x25	16/8	CGA,EGA	b800	Comp,RGB,Enhanced
4	графика	320x200	4	CGA,EGA	b800	Comp,RGB,Enhanced
5	графика	320x200	4 полутона	CGA,EGA	b800	Composite
6	графика	640x200	2	CGA,EGA	b800	Comp,RGB,Enhanced
7	текст	80x25	3 (b/w/bold)	MA,EGA	b000	TTL Monochrome
0dH	графика	320x200	16	EGA	A000	RGB,Enhanced
0eH	графика	640x200	16	EGA	A000	RGB,Enhanced
0fH	графика	640x350	3 (b/w/bold)	EGA	A000	Enhanced,TTL Mono
10H	графика	640x350	4 или 16	EGA	A000	Enhanced

8,9,0aH режимы PCjr
0bH,0cH (резервируется для EGA BIOS)

Задание 3. Исследовать формат хранения копии экрана в режимах 03h (сегмент B800h) и 13h (сегмент A000h). Объяснить назначение полей байта цветовых атрибутов.
Создать пиктограмму 10x10 для двух указанных режимов.

Замечание: для ◄EGA► и ─Jr─ можно добавить 80H к AL, чтобы инициализировать видео режим без очистки экрана.

01H уст. размер/форму курсора (текст). Курсор, если он видим, всегда мерцает.
Вход: CH = начальная строка (0-1fH; 20H=подавить курсор)
CL = конечная строка (0-1fH)

02H уст. позицию курсора. Установка на строку 25 делает курсор невидимым.
Вход: BH = видео страница
DH,DL = строка, колонка (считая от 0)

03H читать позицию и размер курсора
Вход: BH = видео страница
Выход: DH,DL = текущие строка,колонка курсора
CH,CL = текущие начальная,конечная строки курсора (см. функцию 01H)

04H читать световое перо
Вход: нет
Выход: AH = триггер (0=нет значений; 1=возвращены значения светового пера)
DH,DL = строка,колонка символа (текст)
BX = колонка точки (графика)
CH = строка точки (для графики EGA возвращается в CX)

05H выбрать активную страницу дисплея
Вход: AL = номер страницы (большинство программ использует страницу 0)

06H листать окно вверх (или очистить). Листать на 1 или более строк вверх.
Вход: CH,CL = строка,колонка верхнего левого угла окна (считая от 0)
DH,DL = строка,колонка нижнего правого угла окна (считая от 0)
AL = число пустых строк, вдвигаемых снизу (0=очистить все окно) BH = видео атрибут, используемый для пустых строк

07H листать окно вниз (вдвинуть пустые строки в верхнюю часть окна)
Вход: (аналогично функции 06H)

08H читать символ/атрибут в текущей позиции курсора
Вход: BH = номер видео страницы
Выход: AL = прочитанный символ
AH = прочитанный видео атрибут (только для текстовых режимов)

09H писать символ/атрибут в текущей позиции курсора

```
    Вход:  BH = номер видео страницы
           AL = записываемый символ
           CX = счетчик (сколько экземпляров символа записать)
           BL = видео атрибут (текст) или цвет (графика)
                (графические режимы: +80H означает XOR с символом на экране)
```

```
0aH писать символ в текущей позиции курсора
    Вход:  BH = номер видео страницы AL = записываемый символ
           CX = счетчик (сколько экземпляров символа записать)
```

```
0bH выбрать цвет палитры/бордюра (CGA-совместимые режимы)
    Вход:  BH = 0: (текст) выбрать цвет бордюра
                   BL = цвет бордюра (0-1fH; от 10H до 1fH - интенсивные)
           BH = 1: (графика) выбрать палитру
                   BL = 0: палитра green/red/brown
                   BL = 1: палитра cyan/magenta/white
```

```
0cH писать графическую точку (слишком медленно для большинства приложений!)
    Вход:  BH = номер видео страницы
           DX,CX = строка,колонка
           AL = значение цвета (+80H означает XOR с точкой на экране)
```

```
0dH читать графическую точку (слишком медленно для большинства приложений!)
    Вход:  BH = номер видео страницы
           DX,CX = строка,колонка
               Выход:          AL     =     прочитанное     значение     цвета
```

```
0eH писать символ на активную видео страницу (эмуляция телетайпа)
    Вход:  AL = записываемый символ (использует существующий атрибут)
           BL = цвет переднего плана (для графических режимов)
```

```
0fH читать текущий видео режим
    Вход:  нет
   Выход:  AL = текущий режим (см. функцию 00H)
           AH = число текстовых колонок на экране
           BH = текущий номер активной страницы дисплея
```

```
10H-12H  См.  ◄EGA► BIOS сервис
```

```
13H писать строку ▌AT▐  ◄EGA►  Выдает строку в позиции курсора. Символы 0dH
    (CarRet), 0aH (LineFeed), 08H (backspace) и 07H (Beep) трактуются как
    команды управления и не высвечиваются.
    Вход:   ES:BP => выводимая строка (спецформат для AL=2 и AL=3)
            CX = длина строки (подсчитываются только символы)
            DH,DL = строка,колонка начала вывода BH = номер страницы
            AL = код подфункции:
                 0 = использовать атрибут в BL; не трогать курсор
                 1 = использовать атрибут в BL; курсор - в конец строки
                 2 = формат строки: char,attr, char,attr...; не трогать курсор
                 3 = формат строки: char,attr, char,attr...; передвинуть курсор
```

Домашнее задание. Написать программу, использующую функции BIOS для отрисовки пиктограмм задания 3 в режимах 3,13,14.

Лабораторная работа 4. Программирование ввода-вывода на примере UART (4 часа)

Команда mode позволяет задать режим работы последовательного порта IBM PC:

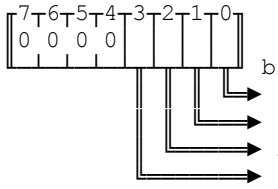
```
C:\>mode /?
Настройка системных устройств.

Последовательный порт:   MODE COMm[:] [BAUD=b] [PARITY=p] [DATA=d] [STOP=s]
                                      [to=on|off] [xon=on|off] [odsr=on|off]
                                      [octs=on|off] [dtr=on|off|hs]
                                      [rts=on|off|hs|tg] [idsr=on|off]

Состояние устройства:    MODE [устройство] [/STATUS]

Переадресация печати:    MODE LPTn[:]=COMm[:]

Выбор кодовой страницы:  MODE CON[:] CP SELECT=yyy

Вывод кодовой страницы:  MODE CON[:] CP [/STATUS]

Режим работы экрана:     MODE CON[:] [COLS=c] [LINES=n]

Ввод с клавиатуры:       MODE CON[:] [RATE=r DELAY=d]
```

Регистры микросхемы UART доступны по следующим адресам:

```
The BIOS Data area contains a list of up to four COM port base addresses.
During POST  the BIOS tests for and initializes COM1 and COM2.

      • The COM1 adapter decodes ports 3f8H through 3ffH
      • The COM2 adapter decodes ports 2f8H through 2ffH

The ROM BIOS INT 14H will work with any of the four ports, as long as you store
the port's base address into the COMM port table starting at 0:0400.  It is
critical that no two adapters share the same addresses or neither will work.

BIOS supports a simple polling-style serial I/O.  The adapter is able to force a
hardware interrupt on a variety of conditions, depending upon the values in the
Interrupt Enable Register (3f9H or 2f9H).

      COM1 forces interrupt level 4 (IRQ 4 is handled by the INT 0cH vector)
      COM2 forces interrupt level 3 (IRQ 3 is handled by the INT 0bH vector)
```

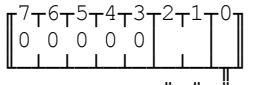
Port	Description	Baud	Divisor	Baud	Divisor
3f8H	Write: transmitter holding register. 8 bits of character to be sent.				
	Read: receiver buffer register. 8 bits of character received.				
	Write: (when DLAB=1) divisor latch low byte				
	After an OUT 3fbH,80H this port holds				
	the low byte of the clock divisor	110	1040	1200	96
	which, together with the high byte	150	768	2400	48
	(port 3f9h) constitute a 16-bit value	300	384	4800	24
	that sets the baud rate as shown:	600	192	9600	12

3f9H Write: divisor latch high byte (when DLAB=1; ie, after OUT 3fbH,80H)
      Write: interrupt enable register



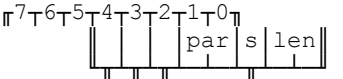
0: 1=enable an interrupt when rec'd data is available
1: 1=enable interrupt when transmitter buffer is empty
2: 1=enable int on rec'r line status (error or break)
3: 1=enable interrupt on modem status (CTS,DSR,RI,RLSD)

3faH Read: interrupt identification register. When an interrupt occurs, read
      this register to find what caused it.



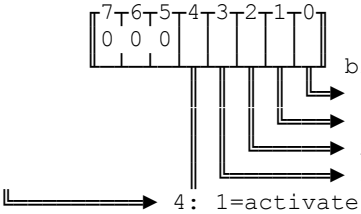
0: 1=no interrupt pending (can be used for polling)
1: 00=receiver line status interrupt. Occurs upon:
      overrun, parity, or framing error, or break
      Reset by reading line status (port 3fdH)
   01=received data available
      Reset by reading receiver buffer (port 3f8H)
   10=transmitter buffer empty
      Reset by writing transmitter buffer (port 3f8H)
   11=modem status. Occurs upon: Clear To Send, Data
       Set Ready, Ring Ind, or Rec'd Line Signal Detect
      Reset by reading modem status (port 3feH)

3fbH Read/Write: line control register



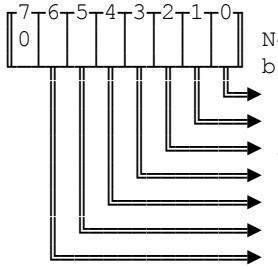
0-1: word length: 00=5, 01=6, 10=7, 11=8
2: stop bits: 0=1, 1=2 (some un-noteworthy exceptions)
3-4 parity: x0=None, 01=Odd, 11=Even
5: stuck parity (not used by BIOS)
6: enable break control. 1=start sending 0s (spaces)
7: DLAB (Divisor Latch Access Bit) Determines the mode
   of ports 3f8H and 3f9H. 1=set baud rate, 0=normal

3fcH Write: modem control register



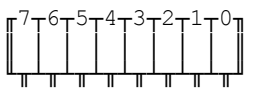
0: 1=activate -DTR (-data terminal ready), 0=deactivate
1: 1=activate -RTS (-request to send), 0=deactivate
2: 1=activate -OUT1 (spare, user-designated output)
3: 1=activate -OUT2
4: 1=activate loopback for diagnostic testing

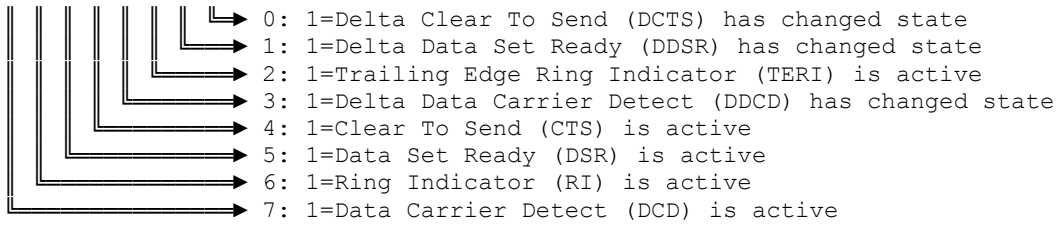
3fdH Read: line status register



0: 1=data ready (DR). Reset by reading recv'r buffer
1: 1=overrun error (OE). Previous character is lost
2: 1=parity error (PE). Reset by reading line status
3: 1=framing error (FE). Bad stop bit in character
4: 1=break indicated (BI). Sustained space received
5: 1=transmitter holding register empty. OK to send
6: 1=transmitter empty. No data being processed.

3feH Read: modem status register





Задание 1. Используя нуль-модемный кабель, осуществить обмен данными мужду ПК в окне debug. Изменить командой mode скорость передачи данных. Написать и установить собственный обработчик прерываний UART.

INT 14H: Ввод-вывод через последовательный порт еще ↓

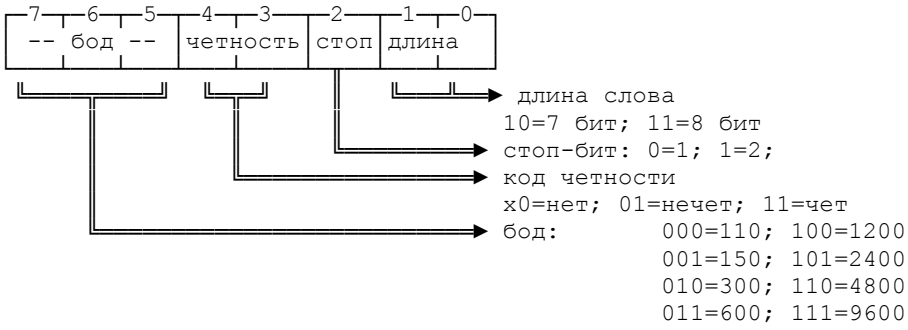
Эти функции BIOS предоставляют доступ к двум портам RS-232.

Начиная с 0:0400, хранятся базовые адреса до 4-х последовательных портов, однако POST проверяет и инициализирует лишь два первых порта.

См. Порты асинхронного адаптера - описание аппаратного интерфейса.

AH Сервис

00H инициализировать коммуникационный порт
Вход: DX = номер порта (0-1)
AL = битовые флаги параметра инициализации:



Выход: AH = статус коммуникаций (см. ниже)

01H послать символ через выбранный порт RS-232
Вход: DX = номер порта (0-1)
AL = посылаемый символ
Выход: AL сохранен.
Если установлен бит 7 в AH, то произошла ошибка, и
AH (биты 6-0) = статус линии связи (см. AH ниже)

02H получить символ через выбранный порт RS-232
Вход: DX = номер порта (0-1)
Выход: AL = полученный символ
AH ненулевой, если произошла ошибка

03H дать статус порта связи
Вход: DX = номер порта (0-1)
Выход: AX = статус порта связи

AH = статус линии	AL = статус модема
bit 7: timeout	bit 7: received line detect signal
bit 6: trans shift reg empty	bit 6: ring indicator

```
bit 5: trans holding reg empty bit 5: data set ready
bit 4: break detect            bit 4: clear to send
bit 3: framing error           bit 3: delta recv line signal detect
bit 2: parity error            bit 2: trailing edge ring detector
bit 1: overrun error           bit 1: delta data set ready
bit 0: data ready status       bit 0: delta clear to send
```

Домашнее задание. Написать программу пересылки файла между ПК, используя обработчик задания 1. Отчет должен содержать листинг кода с подробными комментариями.

Лабораторная работа 5. Исследование функционирования стека x86 (3 часа)

Таблица инструкций процессора x86 приведена ниже:

* означает только ▌286▌ (недоступно для 8088/8086).

	x0	x1	x2	x3	x4	x5	x6	x7
0x	ADD r/m,r8	ADD r/m,r16	ADD r8,r/m	ADD r16,r/m	ADD AL,im8	ADD AX,im16	PUSH ES	POP ES
1x	ADC r/m,r8	ADC r/m,r16	ADC r8,r/m	ADC r16,r/m	ADC AL,im8	ADC AX,im16	PUSH SS	POP SS
2x	AND r/m,r8	AND r/m,r16	AND r8,r/m	AND r16,r/m	AND AL,im8	AND AX,im16	SEG ES	DAA
3x	XOR r/m,r8	XOR r/m,r16	XOR r8,r/m	XOR r16,r/m	XOR AL,im8	XOR AX,im16	SEG SS	AAA
4x	INC AX	INC CX	INC DX	INC BX	INC SP	INC BP	INC SI	INC DI
5x	PUSH AX	PUSH CX	PUSH DX	PUSH BX	PUSH SP	PUSH BP	PUSH SI	PUSH DI
6x	* PUSHA	* POPA	* BOUND					
7x	JO	JNO	JB/ JNAE	JNB/ JAE	JE/ JZ	JNE/ JNZ	JBE/ JNA	JNBE/ JA
8x	ArOp1 r/m,im8	ArOp1 r/m,im16	ArOp2 r/m8,im8	ArOp2 rm16,im8	TEST r/m,r8	TEST r/m,r16	XCHG r8,r/m	XCHG r16,r/m
9x	NOP	XCHG AX,CX	XCHG AX,DX	XCHG AX,BX	XCHG AX,SP	XCHG AX,BP	XCHG AX,SI	XCHG AX,DI
Ax	MOV AL,mem8	MOV AX,mem16	MOV mem8,AL	MOV mem16,AX	MOVSB	MOVSW	CMPSB	CMPSW
Bx	MOV AL,im8	MOV CL,im8	MOV DL,im8	MOV BL,im8	MOV AH,im8	MOV CH,im8	MOV DH,im8	MOV BH,im8
Cx	* ShfOp r/m8,imm	* ShfOp r/m16,im	RET near ёim16	RET near	LES r16,mem	LDS r16,mem	MOV mem,im8	MOV mem,im16
Dx	ShftOp r/m8,1	ShftOp r/m16,1	ShftOp r/m8,CL	ShftOp r/m16,CL	AAM	AAD		XLAT
Ex	LOOPNE/ LOOPNZ	LOOPE/ LOOPZ	LOOP	JCXZ	IN AL,port8	IN AX,port8	OUT AL,port8	OUT AX,port8
Fx	LOCK		REP/ REPNE	REPZ/ REPE	HALT	CMC	Grp1 r/m8	Grp1 r/m16
	x0	x1	x2	x3	x4	x5	x6	x7

	x8	x9	xA	xB	xC	xD	xE	xF
0x	OR r/m,r8	OR r/m,r16	OR r8,r/m	OR r16,r/m	OR AL,im8	OR AX,im16	PUSH CS	*Prtectd Mode Op
1x	SBB r/m,r8	SBB r/m,r16	SBB r8,r/m	SBB r16,r/m	SBB AL,im8	SBB AX,im16	PUSH DS	POP DS

2x	SUB r/m,r8	SUB r/m,r16	SUB r8,r/m	SUB r16,r/m	SUB AL,im8	SUB AX,im16	SEG CS	DAS
3x	CMP r/m,r8	CMP r/m,r16	CMP r8,r/m	CMP r16,r/m	CMP AL,im8	CMP AX,im16	SEG DS	AAS
4x	DEC AX	DEC CX	DEC DX	DEC BX	DEC SP	DEC BP	DEC SI	DEC DI
5x	POP AX	POP CX	POP DX	POP BX	POP SP	POP BP	POP SI	POP DI
6x	* PUSH imm16	* IMUL r/m,im16	*PUSH imm8	* IMUL r/m,im8	* INSB	*INSW	* OUTSB	* OUTSW
7x	JS	JNS	JP/ JPE	JNP/ JPO	JL/ JNG	JNL/ JGE	JLE/ JNG	JNLE/ JG
8x	MOV r/m,r8	MOV r/m,r16	MOV r8,r/m	MOV r16,r/m	MOV r/m,seg	LEA r16,mem	MOV seg,r/m	POP r/m
9x	CBW	CWD	CALL far	WAIT	PUSHF	POPF	SAHF	LAHF
Ax	TEST AL,mem8	TEST AX,mem16	STOSB	STOSW	LODSB	LODSW	SCASB	SCASW
Bx	MOV AX,im16	MOV CX,im16	MOV DX,im16	MOV BX,im16	MOV SP,im16	MOV BP,im16	MOV SI,im16	MOV DI,im16
Cx	* ENTER im16,im8	* LEAVE	RET far ёim16	RET far	INT 3	INT im8	INTO	IRET
Dx	ESC 0	ESC 1	ESC 2	ESC 3	ESC 4	ESC 5	ESC 6	ESC 7
Ex	CALL near	JMP near	JMP far	JMP short	IN AL,DX	IN AX,DX	OUT AL,DX	OUT AX,DX
Fx	CLC	STC	CLI	STI	CLD	STD	Grp2 r/m8	Grp3 r/m16
	x8	x9	xA	xB	xC	xD	xE	xF

Операции, непосредственно модифицирующие стек:

```
PUSH src            переслать reg16 или r/m16 в стек
                     SP-=2; SS:[SP]←src

PUSH immed          переслать immed16 (или расширение со знаком immed8) в стек.
                    SP-=2; SS:[SP]←immed16    ▌286▌ только.

PUSHA               PUSH ALL: скопировать общие регистры в стек.  ▌286▌ только.
                     SP-=10H; AX,BX,CX,DX,SI,DI,BP,SP сохраняются в стеке

PUSHF               PUSH Flags: переслать регистр флагов в стек
                     SP-=2; SS:[SP]←флаги

POP  dest           переслать из стека в reg16 или r/m16 или segreg
                     dest←SS:[SP]; SP+=2

POPA                POP All: скопировать общие регистры из стека. ▌286▌ только.
                     SP+=10H; AX,BX,CX,DX,SI,DI,BP восстанавливаются из стека

POPF                POP Flags: переслать из стека в регистр флагов
                     flags←SS:[SP]; SP+=2
```

```
JMP     target       безусловная передача управления на метку
                         short: IP←(IP+(смещение цели, расширенное со знаком)
                          near: IP←(IP+(смещение цели)
                      indirect: IP←(регистр или значение в памяти)
                           far: SS←целевой_сегмент; IP←целевое_смещение

JCXZ    short_label  переход если CX==0

LOOP    short_label  CX←(CX-1) переход если CX!=0

LOOPE/  short_label  CX←(CX-1) переход если CX!=0 && ZF==ZR==1
 LOOPZ

LOOPNE/ short_label  CX←(CX-1) переход если CX!=0 && ZF==NZ==0
 LOOPNZ

Jcond   short_label  переход при удовлетворении условия
                     IP←(IP+(8-битовое смещение, расширенное со знаком до 16))

JA/JNBE short_label  переход если выше ((CF & ZF)==0 после беззнак ариф)
JAE/JNB short_label  переход если выше/равно (CF==NC==0 после беззнак ариф)
JB/JC   short_label  переход если ниже/переход если Carry (CF==CY==1)
JE/JZ   short_label  переход если равно (ZF==ZR==1)
JG/JNGE short_label  переход если больше (SF==(OF & ZF) после знак ариф)
JGE/JNL short_label  переход если больше/равно (SF==OF после знак ариф)
JL/JNGE short_label  переход если меньше (ZF != OF после знак ариф)
JLE/JNG short_label  переход если меньше/равно (SF!=OF || ZF==0 после знак ариф)
JNC     short_label  переход если не Carry (CF==NC==0) (то же, что JAE/JNB)
JNE/JNZ short_label  переход если не равно (ZF==NZ==0)
JNO     short_label  переход если не переполнение (OF==NO==0)
JNP/JPO short_label  переход если нечет (PF==PO==0: число 1-битов НЕЧЕТНО)
JNS     short_label  переход если не знак (SF==PL==0: -- старший бит dest)
JO      short_label  переход если переполнение (OF==OV==1)
JP/JPE  short_label  переход если чет(PF==PE==1 число 1-битов ЧЕТНО)
JS      short_label  переход если знак (SF==NG==1: -- старший бит dest)

INT type             выполнить программное прерывание (вызов системной функции)
                      PUSHF; IF←0; TF←0;
                      PUSH CS; PUSH IP
                      IP← 0000:[type * 4]; CS ← 0000:[(type * 4) + 2]

INTO type            если OF==OV==1, то выполнить INT type

IRET                 возврат из прерывания. Действие эквивалентно следующему:
                      POP IP; POP CS; POPF
CALL
RET
CALLF
RETF
```

Лабораторная работа 6. Функции BIOS (3 часа)

Общйи список функций BIOS приведен ниже:

Прерывания	Сервис	Указатели
INT 00H Деление на 0	INT 10H Видео сервис	INT 1dH Видео параметры
INT 01H Пошаговое	INT 11H Список оборудования	INT 1eH Параметры дискет
INT 02H Немаскируемое	INT 12H Размер исп.памяти	INT 1fH Символы графики
INT 03H Точка прерыв.	INT 13H Дисковый в/в	
INT 04H Переполнение	INT 14H В/в через последовательный порт	
INT 05H Печать экрана	INT 15H Расшир.сервис AT	INT 41H hard disk 0 parms
INT 06H (резерв)	INT 16H В/в клавиатуры	INT 46H hard disk 1 parms
INT 07H (резерв)	INT 17H В/в принтера	
INT 08H Таймер	INT 18H ROM-BASIC	INT 44H EGA graphic chars
INT 09H Клавиатура	INT 19H Загрузка	
INT 0aH-0dH (hdwr ints)	INT 1aH В/в таймера	INT 4aH user alarm addr
INT 0eH Дискета	INT 1bH Прерывание клавиатуры	INT 50H CMOS timer int
INT 0fH (hdwr int)	INT 1cH Пользовательское прерывание по таймеру	
	INT 20H-2fH Прерывания DOS	

Задание 1. Используя функции, перечисленные ниже, а также изученные ранее, написать коммуникационную программу для передачи данных клавиатурного ввода с одного ПК на другой. Необходимо установить собственные обработчики прерываний таймера, клавиатуры и последовательного порта.

INT 08H: Прерывание от таймера еще ↓

Это аппаратно генерируемое прерывание (IRQ 0) вызывается по каждому тику часов реального времени PC. Часы тикают каждые 55мс, или около 18.2 раз в секунду.

Код ROM-BIOS для этого прерывания обновляет значение часов на 0:046 сек. Этот код также выключает двигатели гибких дисков по истечении примерно двух секунд без операций ввода/вывода. См. также данные BIOS .

Если вы хотите использовать это прерывание для обработки событий, зависящих от времени, вы должны предусмотреть код, сообщающий системе, когда прерывание закончено. Магическая последовательность такова:

```
mov     al,20H     ;послать сигнал Конец-Прерывания
out     20H,al     ; контроллеру прерываний 8259
```

Большинство программ используют для этого ROM-BIOS. Например, вначале программа пользователя сохраняет вектор ROM-BIOS в переменной с именем cs:[saved_int8]. Затем по каждому прерыванию INT 08H выполняется следующий код:

```
        pushf                                ;имитировать INT
call  dword ptr cs:[saved_int8]    ;выполнить нормальную обработку
        cmp   cs:my_var,test_value           ;теперь сделать свои черные дела
    ... и т.д. ...
        iret    ;возврат в прерванную прикладную программу "переднего плана"
```

INT 09H: Прерывание от клавиатуры еще ↓

Это аппаратно генерируемое прерывание (IRQ 1) выполняется при каждом нажатии и отпускании клавиши. Код ROM-BIOS интерпретирует это, сохраняя значения в буфере клавиатуры по адресу 0:041e. Он также обрабатывает специальные случаи клавиш PrtSc и SysReq, и отслеживает состояние клавиш Shift и различных Lock.

См.: INT 16H ..... сервис BIOS для доступа к ключам, запомненным в буфере, и опроса состояния служебных клавиш.
Сканкоды ..... перечень значений всех возможных ключей, как они принимаются прерыванием INT 09H.
Таблица ASCII

Расширение ASCII . сводка значений, сохраняемых BIOS в буфере
клавиатуры после трансляции сканкодов.
Флаги клавиатуры . как получить, проверить и модифицировать
битовые значения флагов Shift и Lock.

Резидентные popup-программы, которые обычно предусматривают "кнопку вызова", перехватывают INT 09H и проверяют на определенный ключ примерно так:

```
         push    ax
in      al,60H             ;читать ключ
         cmp     al,POP_KEY         ;это кнопка вызова?
         je      do_pop             ; да, активизировать popup
                                    ;  нет, уйти на исходный обработчик
         pop     ax
         jmp     cs:[int9_vect]     ;переход на первоначальный обработчик

do_pop:  ;------ следующий код необходим для отработки аппаратного прерывания
         in      al,61H             ;взять значение порта управления клавиатурой
         mov     ah,al              ; сохранить его
         or      al,80h             ;установить бит разрешения для клавиатуры
         out     61H,al             ; и вывести его в управляющий порт
         xchg    ah,al              ;извлечь исходное значение порта
         out     61H,al             ; и записать его обратно

         mov     al,20H             ;послать сигнал "конец прерывания"
         out     20H,al             ; контроллеру прерываний 8259
         ;------ дальше - прочие проверки, и наконец - активизация popup
```

INT 16H: Сервис клавиатуры еще ↓

Это - интерфейс прикладного уровня с клавиатурой. Нажатия клавиш на самом деле обрабатываются асинхронно на заднем плане. Когда клавиша получена от клавиатуры, она обрабатывается прерыванием INT 09H и помещается в циклическую очередь.

См. Клавиатуру AT о способах ускорения клавиатуры и других сведениях.

AH Сервис

00H читать (ожидать) следующую нажатую клавишу
Выход: AL = ASCII символ (если AL=0, AH содержит Расширенный код ASCII)
AH = Сканкод или Расширенный код ASCII

01H Проверить готовность символа (и показать его, если так)
Выход: ZF = 1 если символ не готов.
ZF = 0 если символ готов.
AX = как для подфункции 00H (но символ здесь не
удаляется из очереди).

02H Читать состояние shift-клавиш. Определить, какие shift-клавиши нажаты в данный момент, находится ли клавиатура в состоянии NumLock, и т.п.
Выход: AL = статус клавиатуры -- см. Флаги клавиатуры

INT 1cH: Пользовательское прерывание по таймеру

Этот вектор (0:0070) BIOS берет по каждому тику аппаратных часов (каждые 55 миллисекунд; приблизительно 18.2 раз в секунду). Первоначально он указывает на IRET, но может быть изменен пользовательской программой, чтобы адресовать фоновую программу пользователя, базирующуюся на таймере.

Поскольку программа INT 1cH выполняется во время низкоуровневого аппаратного прерывания, вы должны помнить, что система еще не сбросила контроллер прерываний и потому другие аппаратные прерывания, в том числе прерывание от клавиатуры, не будут происходить при работе INT 1cH (т.е. вы не получите ввода пользователя).

Большинство popup-программ предпочитают перехватывать вектор INT 08H , вызывать первоначальный вектор, а затем уже выполнять операции, зависящие от времени, после того как BIOS закончит свою обслугу.

См. INT 08H насчет предложений и рекомендаций.

```
╔═══════════════════════════════════════════════════════════════════════════╗
║                          INT 21H: Сервис DOS                              ║
╚═══════════════════════════════════════════════════════════════════════════╝
```

Это прерывание служит главным входом большинства функций DOS.

Программа, запрашивающая сервис DOS, должна подготовить всю необходимую информацию в регистрах и управляющих блоках, указать в регистре AH номер желаемой функции DOS и затем вызвать прерывание INT 21H.

См.также: Функции DOS
Функции DOS по группам
О функциях DOS

Следующие функции или векторы DOS НЕ вызываются через прерывание INT 21H:

INT 20H Завершить программу

```
╔═══════════════════════════════════════════════════════════════════════════╗
║                    DOS Fn 09H: Выдать строку на дисплей                   ║
╚═══════════════════════════════════════════════════════════════════════════╝
```

Вход	AH	09H
	DS:DX	адрес строки, заканчивающейся символом '$' (ASCII 24H)
Выход	нет	

Описание: Строка, исключая завершающий ее символ '$', посылается на стандартный вывод.

Символы Backspace обрабатываются как в функции 02H Display Char.

Обычно, чтобы перейти на новую строку, включают в текст пару CR/LF (ASCII 13H и ASCII 0aH).

Строки, содержащие '$', можно выдать через 40H Write Handle (BX=0).

Домашнее задание. Отладить программу на ПК, используя два последовательных порта и два коммандных окна. Предоставить листинг программы с подробными комментариями.

ТЕМЫ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

1. История ЭВМ. Рассматривается история развития ЭВМ, представлены поколения ЭВМ, параметры ЭВМ разных поколений, стоимостные оценки ЭВМ. Представлены 3 этапа информационных технологий, а также основные принципы работы ЭВМ. Логические основы. Дается понятие булевой алгебры, описаны задачи анализа и синтеза. Дается описание элементарных функций одной и двух переменных. Представлены основные эквивалентности.

2. Совершенные дизъюнктивные и конъюнктивные нормальные формы ФАЛ. Дано определение совершенной дизъюнктивной и конъюнктивной нормальных форм. Представлены правила записи функции по нулям и единицам. Дано понятие функциональной полноты, поставлена задача минимизации функции. Сформулирована теорема Квайна. Метод проб. Представлены способы минимизации на основе метода проб, метода Квайна-Мак-Класки, на основе минимизирующих диаграмм для функции 2-х, 3-х, 4-х переменных (диаграммы Вейча).

3. Минимизация неполностью определенных функций. Синтез функций в базисах штрих Шеффера и стрелка Пирса, даны подходы к минимизации конъюнктивных форм.

4. Арифметические основы. Рассматриваются системы счисления, представлена методика выбора системы счисления, даны правила перевода чисел из одной системы счисления в другую. Способы представления чисел в ЭВМ. Представлены способы представления чисел в ЭВМ: фиксированная и плавающая запятая. Описаны прямой, дополнительный и обратный коды. Дано сложение чисел в дополнительном и обратном кодах.

5. Модифицированные коды. Рассматриваются модифицированный дополнительный и обратный коды. Даны методы умножения чисел с фиксированной запятой в прямом и дополнительном кодах, а также алгоритмы сдвига. Методы деления чисел с фиксированной

запятой в прямых кодах и дополнительных (обратных) кодах. Описаны операции над числами с плавающей запятой – умножение, деление, сложение, вычитание.

6. Структура однопрограммной ЭВМ. Рассматриваются классические основы построения ЭВМ (машина Тьюринга, элемент и автомат Неймана), принципы Неймана построения ЭВМ, структура классической ЭВМ.

7. Система кодирования команд. Способы адресации . Рассматриваются различные системы кодирования команд, взаимосвязь основных параметров ЭВМ с форматом команды, основные способы адресации и их влияние на время выборки операнда, длину поля адреса, особенности их использования при составлении программ для обработки различных структур данных.

8. Цикл выполнения команды Рассматривается взаимодействие узлов и устройств классической трехадресной ЭВМ на различных этапах автоматического выполнения программ. Основы схемотехнической реализации ЭВМ . Рассматриваются основные элементы, составляющие систему логических элементов, их схемотехническая реализация, статические и динамические параметры, порядок проектирования комбинационных схем на примере одноразрядного сумматора.

9. Архитектура персонального компьютера. Рассматривается обобщенная структура персональной ЭВМ, реализованной по магистральному принципу, структура и основные блоки 16-разрядного микропроцессора I8086, представление цифровой и символьной информации в ЭВМ, организация памяти и формирование физического адреса в сегментированном адресном пространстве.

ПРИМЕРЫ ТЕСТОВ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Тест по контролю знаний **Дисциплина: "Организация ЭВМ и систем"**

Факультет МиИ Специальность: 2202 Курс II

Утверждено на заседании кафедры _____ 2005г. **"Утверждаю":_____ зав. кафедрой ИУС**

Группа: _____ ФИО Студента:_____

Сложение по модулю два определяется как (подчеркивание обозначает Инверсию):

1	1	2	3	4	5
	$XY+\underline{XY}$	$(\underline{X}+Y)(X+\underline{Y})$	$\underline{X}Y+X\underline{Y}$	$X\underline{X}+Y\underline{Y}$	$(X+\underline{Y})(\underline{X}+Y)$

Канонический вид булевой функции определяется как:

	1	2	3	4	5
	Пересечение объединений	Объединение пересечений	Дополнение пересечений	Пересечение дополнений	Объединение дополнений

Выберите последовательностное устройство:

3	1	2	3	4	5
	Мультиплексор	ПЗУ	Регистр	Дешифратор	АЛУ

Процессор i8086 является представителем одной из архитектур ВС:

4	1	2	3	4	5
	Гарвардской	Мультискалярной	Векторной	Бостонской	Пристонской

Совместное адресное пространство памяти и ввода/вывода ВС уменьшает:

5	1	2	3	4	5
	Время доступа к устройству	Количество команд процессора	Время доступа к оперативной памяти	Количество холостых тактов шины	Производитель-ность ВС в целом

За приоритетное обслуживание запросов внешних устройств в ВС отвечает:

6	1	2	3	4	5
	Контроллер ПДП	Сопроцессор	Микропроцессор	Контроллер прерываний	Контроллер ввода/вывода

Организация ДЗУПВ в банки позволяет:

7	1	2	3	4	5
	Уменьшить длину машинного слова	Увеличить емкость ОЗУ	Уменьшить эффективное время доступа к ОЗУ	Уменьшить длину машинной команды	Увеличить разрядность шины адреса

Роль арбитра шины, помимо процессора, выполняет:

8	1	2	3	4	5
	Сопроцессор	Контроллер ПДП	Контроллер прерываний	Видео-контроллер	Системный таймер

За трансляцию виртуальных адресов в физические отвечает:

9	1	2	3	4	5
	Буфер TLB	Кэш-память данных/команд	Регистр CR2	Дескриптор сегмента	Контроллер НЖМД

Дескриптор сегмента НЕ содержит поля:

10	1	2	3	4	5
	Base	Type	Length	DPL	Limit

TSS хранит:

11	1	2	3	4	5
	Содержимое сегмента стека	Дескриптор сегмента стека	Шлюз вызова	Селектор сегмента стека	Контекст задачи

За текущий уровень привилегии задачи отвечает:

12	1	2	3	4	5
	Поле RPL селектора сегмента	Поле CPL	Поле DPL	Регистр CR0	Регистр GDTR

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Дисциплина: "Организация ЭВМ и систем"

Тест по контролю знаний Факультет МиИ Специальность: 2202 Курс II

Утверждено на заседании кафедры _____ 2006г. **"Утверждаю":_____ зав. кафедрой ИУС**

Группа: _____ ФИО Студента:_____

1	Виртуальные адреса заменяются на физические во время:				
	загрузки программы в ОП	обращения к виртуальному адресу	компиляции программы	обращения к физическому адресу	выделения сегмента кода для программы

2	Где хранятся каталоги и таблицы станиц:				
	в регистрах процессора	в стеке ядра ОС	в RAM	в ROM	в области своппинга

3	Выделите термины, являющиеся синонимами, применительно к сетевым ОС:				
	оболочка	сервер	клиент	сервис	услуга

4	Драйвер устройства выполняет функции:				
	управления файловой системой	обработки прерывания от устройства	организации прямого доступа к памяти	управления выводом информации	низкого уровня по управлению устройством

5	Выделите термины, являющиеся синонимами, применительно к многозадачной ОС:				
	программа	процесс	задача	поток	нить

6	За приоритетное обслуживание запросов устройств ввода-вывода к процессору отвечает:				
	Контроллер ПДП	Сопроцессор	Микропроцессор	Контроллер прерываний	Контроллер ввода/вывода

7	Выделите термины, являющиеся синонимами, применительно к режимам работы задачи:				
	реальный	супервизора	защищенный	ядра	пользовательский

8	Роль арбитра шины, помимо процессора, выполняет:				
	Сопроцессор	Контроллер ПДП	Контроллер прерываний	Видео-контроллер	Системный таймер

9	За трансляцию виртуальных адресов в физические отвечает:				
	Буфер TLB	Кэш-память данных/команд	Регистр CR2	Дескриптор сегмента	Контроллер НЖМД

10	Дескриптор сегмента x86 НЕ содержит поля:				
	Base	Type	Length	RPL	Limit

11	Процесс, при прочих равных условиях, будет выполняться быстрее:				
	в мультизадачной ОС	в ОС реального времени	в невытесняющей ОС	в однозадачной ОС	в ОС с вытеснением

12	За текущий уровень привилегии задачи отвечает:				
	Поле RPL селектора сегмента	Поле CPL	Поле DPL	Регистр CR0	Регистр GDTR

13	ОС выделяет файлам пространство на диске				
	секторами	дорожками	кластерами	цилиндрами	байтами
14	TSS хранит:				
	Содержимое сегмента стека	Дескриптор сегмента стека	Шлюз вызова	Селектор сегмента стека	Контекст задачи

15	Организация ДЗУПВ в банки позволяет:				
	Уменьшить длину машинного слова	Увеличить емкость ОЗУ	Уменьшить эффективное время доступа к ОЗУ	Уменьшить длину машинной команды	Увеличить разрядность шины адреса

16	Процессор i8086 является представителем одной из архитектур ВС:				
	Гарвардской	Мультискалярной	Векторной	Бостонской	Пристонской

17	Совместное адресное пространство памяти и ввода/вывода ВС уменьшает:				
	Время доступа к устройству	Количество команд процессора	Время доступа к оперативной памяти	Количество холостых тактов шины	Производитель-ность ВС в целом

18	Сложение по модулю два определяется как (подчеркивание обозначает Инверсию):				
	XY+$\underline{XY}$	($\underline{X}$+Y)(X+$\underline{Y}$)	$\underline{X}$Y+X$\underline{Y}$	X$\underline{X}$+Y$\underline{Y}$	(X+$\underline{Y}$)($\underline{X}$+Y)

19	Канонический вид булевой функции определяется как:				
	Пересечение объединений	Объединение пересечений	Дополнение пересечений	Пересечение дополнений	Объединение дополнений

20	Выберите последовательностное устройство:				
	Мультиплексор	ПЗУ	Регистр	Дешифратор	АЛУ

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Дисциплина: "Организация ЭВМ и систем"

Тест по контролю знаний Факультет МиИ Специальность: 2202 Курс II

Утверждено на заседании кафедры _____ 2007г. **"Утверждаю":_____ зав. кафедрой ИУС**

Группа: _____ ФИО Студента:_____

1	Запишите команды debug для поиска строки "IBM" в трех первых сегментах памяти				
2	Кратко определите функции обработчика исключения отсутствующей страницы				
3	Драйвер устройства выполняет функции:				
	управления файловой системой	обработки прерывания от устройства	организации прямого доступа к памяти	управления выводом информации	низкого уровня по управлению устройством
4	Опишите этапы выполнения запроса к памяти в защищенном режиме				
5	За приоритетное обслуживание запросов устройств ввода-вывода к процессору отвечает:				
	Контроллер ПДП	Сопроцессор	Микропроцессор	Контроллер прерываний	Контроллер ввода/вывода
6	Выделите термины, являющиеся синонимами, применительно к режимам работы задачи:				
	реальный	супервизора	защищенный	ядра	пользовательский

7	Роль арбитра шины, помимо процессора, выполняет:				
	Сопроцессор	Контроллер ПДП	Контроллер прерываний	Видео-контроллер	Системный таймер
8	За трансляцию виртуальных адресов в физические отвечает:				
	Буфер TLB	Кэш-память данных/команд	Регистр CR2	Дескриптор сегмента	Контроллер НЖМД
9	Дескриптор сегмента x86 НЕ содержит поля:				
	Base	Type	Length	RPL	Limit
10	Задача, при прочих равных условиях, будет выполняться быстрее:				
	в мультизадачной ОС	в ОС реального времени	в невытесняющей ОС	в однозадачной ОС	в ОС с вытеснением
11	За текущий уровень привилегии задачи отвечает:				
	Поле RPL	Поле CPL	Поле DPL	Регистр CR0	Регистр GDTR
12	TSS хранит:				
	Содержимое сегмента стека	Дескриптор сегмента стека	Шлюз вызова	Селектор сегмента стека	Контекст задачи
13	Организация ДЗУПВ в банки позволяет:				
	Уменьшить длину машинного слова	Увеличить емкость ОЗУ	Уменьшить эффективное время доступа к ОЗУ	Уменьшить длину машинной команды	Увеличить разрядность шины адреса
14	Процессор i8086 является представителем одной из архитектур ВС:				
	Гарвардской	Мультискалярной	Векторной	Бостонской	Пристонской
15	Укажите линии шины, отвечающие за мех.-мы прерывания и ПДП, стрелками укажите направление (к процессору: >, от него:<)				
16	Сложение по модулю два определяется как (подчеркивание обозначает Инверсию):				
	XY+$\underline{XY}$	($\underline{X}$+Y)(X+$\underline{Y}$)	$\underline{X}$Y+X$\underline{Y}$	X$\underline{X}$+Y$\underline{Y}$	(X+$\underline{Y}$)($\underline{X}$+Y)
17	Канонический вид булевой функции определяется как:				
	Пересечение объединений	Объединение пересечений	Дополнение пересечений	Пересечение дополнений	Объединение дополнений
18	Выберите последовательностное устройство:				
	Мультиплексор	ПЗУ	Регистр	Дешифратор	АЛУ
19	Перечислите способы передачи управления кода с CPL=3 к коду с CPL=0				
20	Запишите команды debug для копирования вектора прерывания 56h в A3h				
21	Определите кратко функции контроллера ДЗУПВ				
22	Что такое битовая карта ввода вывода				
23	Виртуальные адреса заменяются на физические во время:				
	загрузки программы в ОП	обращения к виртуальному адресу	компиляции программы	обращения к физическому адресу	выделения сегмента кода для программы
24	Где хранятся каталоги и таблицы станиц:				
	в регистрах процессора	в стеке ядра ОС	в RAM	в ROM	в области своппинга

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Экзаменационный билет №

Дисциплина: "Организация ЭВМ и систем"

Факультет МиИ Курс 2

Утверждено на заседании кафедры _____ 2007г. **"Утверждаю":_____ зав. кафедрой ИУС**

Группа: _____ **ФИО**

Студента:_____

1	Буферы устройств ввода/вывода работают обычно в режиме: 1.LIFO 2.FIFO 3.Случайной выборки 4.Кэширования 5.Хэширования
2	Перечислите плюсы многозадачности
3	Перечислите поля элемента РТ x86

4	Чем подчиненный сегмент кода отличается от сегмента кода
5	Перечислите известные Вам типы интерфейсов жестких дисков:
6	Перечислите поля селектора сегмента x86
7	Дайте определение канальной программы
8	Опишите этапы выполнения запроса к памяти в защищенном режиме
9	Программа в формате .EXE содержит последовательность команд в виде: 1.Объектного кода 2.Ассемблера 3.Байт-кода 4.Кода NRZ 5.Команд языка Си
10	Опишите последовательность выполнения аппаратного прерывания
11	Процессор в архитектуре "Общая шина" выполняет роль: 1.ПЗУ 2.Арбитра 3.Контроллера 4.ОЗУ 5.Коммутатора
12	Механизм страничного преобразования памяти использует: 1.Прерывания 2.Сегментное преобразвание адреса 3.Прямой доступ к памяти 4.Отображение ввода/вывода на память 5.Кэширование команд
13	Запишите команду debug для сравнения первой половины таблицы векторов прерываний со второй
14	Запишите команду debug для помещения строки "IBM" в самый конец третьего сегмента памяти
15	Определите состав простейшего процессора:
16	Расшифруйте понятие уровня привилегии задачи:
17	Назовите известные Вам последовательностные устройства:
18	Перечислите функции драйвера устройства:
19	Канонический вид булевой функции сложение по модулю 2 определяется как:
20	Укажите известные Вам линии шины управления
21	Перечислите минусы многозадачности
22	Перечислите способы передачи управления кода с CPL=3 к коду с CPL=0 :
23	Максимальная емкость ОЗУ = 16Мбайт. Разрядность шины данных=16. Назовите разрядность шины адреса:
24	Перечислите плюсы организации ДЗУПВ в банки:
25	Определите кратко функции контроллера ДЗУПВ
26	Перечислите содержимое TSS:
27	Что такое битовая карта ввода вывода
28	Драйверы блочных устройств активно используют механизм: 1.Страничного преобразования 2.Прямого доступа к памяти 3.Защиты 4.Переключения задач 5.Сегментного преобразования
29	Выберете программу, хранящуюся в BIOS: 1.cmd 2.POST 3.Ping 4.BASH 5.flash
30	Перечислите поля дескриптора сегмента x86

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Экзаменационный билет №

Дисциплина: "Организация ЭВМ и систем"

Факультет МиИ Курс 2

Механизм страничного преобразования памяти использует:
1.Прерывания 2.Сегментное преобразвание адреса 3.Прямой доступ к памяти 4.Отображение ввода/вывода на память
5.Кэширование команд

Перечислите поля справочника кэш-памяти x86
Определите кратко функции TLB
Перечислите поля элемента PT x86
Опишите последовательность выполнения ПДП
Дешифратор команд представляет из себя: 1.ПЗУ 2.Арбитр 3.Контроллер 4.ОЗУ 5.Коммутатор
Что такое IOPL
Опишите требования, предъявляемые к обработчику прерывания
Дайте определение канальной программы
Перечислите типы сегментов x86
Выберете программу, хранящуюся в BIOS: 1.cmd 2.POST 3.Ping 4.BASH 5.flash
Последовательностные устройства могут быть получены из комбинационных с помощью:
Расшифруйте понятие уровня привилегии задачи:
Максимальная емкость ОЗУ = 32Мбайт. Разрядность шины данных=8. Назовите разрядность шины адреса:
Запишите команды debug для поиска строки "IBM" в первом сегменте памяти и замены ее на "MBI"
Перечислите известные Вам интерфейсы символьных устройств:
Запишите команду debug для сравнения таблицы векторов прерываний с содержимым области данных BIOS
Определите дисциплину стека процессора x86: 1.FIFO 2.LIFO 3.Случайная выборка 4.Кэш 5.Кольцо
Перечислите функции каонтроллера прерываний:
Перечислите минусы принципа МПУ
Перечислите способы передачи управления в реальном режиме x86:
Дайте определение шлюза вызова
Определите состав простейшей ЭВМ:
Перечислите регистры реального режима x86
Запишите таблицу истинности булевой функции "исключающее или":
Приведите порядок времени выполнения запроса к статической и динамической RAM:
Программа в формате .EXE содержит последовательность команд в виде: 1.Объектного кода 2.Ассемблера 3.Байт-кода 4.Кода NRZ 5.Команд языка Си
Опишите структуру встроенной кэш-памяти x86 :
Укажите известные Вам линии шины управления x86
Определите механизм, использующийся в реальном режиме: 1.Страничного преобразования 2.Прямого доступа к памяти 3.Защиты 4.Переключения задач 5.Сегментного преобразования

ЭКЗАМЕНАЦИОННЫЕ БИЛЕТЫ

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Утверждено на заседании кафедры _____ 2000г.

Кафедра АСОИУ
Факультет МиИ
Курс II

Заведующий кафедрой "Утверждаю":_____

Дисциплина "Организация ЭВМ и систем"

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ 1

1. Архитектура минимального микропроцессора.
2. ВС, применяемые в сетях передачи данных.

.
.
.
.
.
.

1. Общие механизмы функционирования ВС: механизм прерываний.
2. Программная модель i8086

1. Общие механизмы функционирования ВС: ПДП.
2. Иерархия аппаратного обеспечения ВС.

1. Общие механизмы функционирования ВС: организация адресного пространства
   памяти и ввода-вывода.
2. Иерархия программного обеспечения ВС.

1. Функциональная классификация ВС.
2. Особенности архитектуры суперскалярных процессоров.

1. Обзор аппаратных средств поддержки многозадачности в i486.
2. Особенности архитектуры транспьютеров.

1. Общие механизмы функционирования ВС: защита.
2. Архитектура минимальной ВС.

1. Особенности архитектуры мультискалярных процессоров
2. Система команд i8086

1. Методы повышения производительности ВС.
2. Специализированные МП для обработки сигналов

1. Архитектурная классификация ВС.
2. Реализация страничного преобразования i486

1. Иерархия устройств памяти.
2. Реализация сегментного преобразования i486

1. Компоненты ВС.
2. Реализация кэш-памяти i486

1. Способы организации распределенных ВС.
2. Общие механизмы функционирования ВС: Стек.