МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Тема 2. BIOS, UEFI и загрузка ОС

Учебно-методическое пособие

для студентов уровня основной образовательной программы: **бакалавриат**
направление подготовки: **09.03.01 - Информатика и вычислительная техника**
направление подготовки: **09.03.03 - Прикладная информатика**

Разработчик
доцент кафедры АСУ

В.Г. Резник

2022

Резник В.Г.

Операционные системы. Тема 2. BIOS, UEFI и загрузка ОС. Учебно-методическое пособие. – Томск, ТУСУР, 2022. – 29 с.

Учебно-методическое пособие предназначено для изучения теоретической части и выполнения лабораторной работы №2 по теме «BIOS, UEFI и загрузка ОС» учебной дисциплины «Операционные системы» для студентов кафедры АСУ ТУСУР уровня основной образовательной программы бакалавриат направлений подготовки: «09.03.01 - Информатика и вычислительная техника» и «09.03.03 - Прикладная информатика».

Оглавление

Введение

Данное пособие содержит учебно-методический материал по второй теме дисциплины «***Операционные системы***».

В определённом смысле этот материал, озаглавленный «***BIOS, UEFI и загрузка ОС***», является продолжением первой темы, но отражает не сами функции ОС, а технические и программные средства обеспечивающие ее загрузку. Знание этого материала и умение им пользоваться входит в базовый набор набор умений, которыми студент должен владеть в процессе выполнения лабораторных работ с использованием ОС УПК АСУ. С этой целью значительная часть его, касающаяся проведения лабораторной работы №2, вынесена в общее методическое пособие [4, раздел 2].

Весь учебный материал данного пособия, как и других пособий данной дисциплины разбит на две части: теоретическую и лабораторную работу. Теоретическая часть содержит основные понятия и определения по данной теме, необходимые для проведения лабораторной работы №2, и формирующие конспективные представления студента по изучаемым вопросам. Для более подробного изучения рассматриваемых вопросов следует воспользоваться учебниками [2-3].

Объем теоретического материала, изложенного в первой части пособия, охватывает архитектуру процессоров ***семейства x86***, которые традиционно используются в качестве аппаратной основы рабочих станций и большинства переносных компьютеров нашей страны. Изучение этой архитектуры стало классикой преподавания данной дисциплины, хотя все современные ОС способны работать на разных аппаратных платформах.

Далее даются общие представления о ***BIOS*** и его функциях, а также рассматриваются основы применения ***UEFI*** и проводится его сравнение с традиционными средствами загрузки ОС. Естественно, что данный материал не может быть усвоен без представлений о блочных и символьных устройствах компьютера, а также без описания структуры блочных устройств, более подробное изучение которых будет проведено в теме 4.

Завершается теоретическая часть описанием универсального загрузчика ***GRUB***, тем самым, обеспечивая необходимыми знаниями процесс проведения лабораторной работы.

Практическая часть обучения по данной теме опирается на выполнение лабораторной работы №2. Здесь предполагается, что студент получил необходимые навыки при выполнении работы №1, достаточные для самостоятельной загрузки ОС УПК АСУ и подключения личной рабочей области. Выполняя данную работу, студент совершенствует свои навыки до уровня самостоятельного создания на ***flashUSB*** аварийного варианта дистрибутива ОС УПК АСУ.

1 Тема 2. BIOS, UEFI и загрузка ОС

В предыдущей теме было рассмотрено множество моделей касающихся как ОС, так и компьютера в целом. В частности, было утверждение, что *ОС является некоторой виртуальной машиной*, которая установлена на аппаратных средствах некоторой ЭВМ.

Большинство современных ОС могут работать на аппаратных средствах ЭВМ различных архитектур, при этом, пользователь может и не знать об этих различиях, поскольку внешние проявления в поведении таких ОС могут быть неразличимы. Достигается это *специальной архитектурой ядра ОС*, в которой:

- *стандартизируется интерфейс* (API) между режимами ядра и пользователя;
- *выделяется программная прослойка* между ядром ОС и аппаратным обеспечением ЭВМ, реализуемая с помощью *модулей (драйверов)*, которые или статически компилируются с ядром или загружаются по мере необходимости.

С другой стороны, прежде чем ОС начнёт функционировать, её ядро должно быть загружено в память ЭВМ. Для этого служит специальное ПО, которое входит в состав самого компьютера и является его неотъемлемой частью, независимо от установленной ОС.

В данной теме, мы рассмотрим такое ПО, которое появилось с развитием средств микропроцессорной техники, позволившей заменить специальные аппаратные средства загрузки ОС на программные.

По традиции, изложение учебного материала будет дано *для архитектуры процессора x86*, модификации которого установлены в компьютерных классах кафедры АСУ и широко используются в переносных ЭВМ.

1.1 Архитектура процессоров x86

Чтобы выйти на должный уровень понимания, следует рассмотреть ряд вопросов, касающихся аппаратной части компьютеров. Традиционно, в качестве основной модели компьютера, рассматриваются изделия созданные на базе процессора ***x86***.

x86 (***Intel 80x86***) — архитектура процессора с одноимённым набором команд, которая впервые была реализованна в процессорах компании *Intel* и относится к серии её процессоров ранних моделей — ***8086, 80186, 80286, 80386 (i386), 80486 (i486)***.

1.1.1 Базовый набор регистров процессора 8086

На рисунке 1.1, показаны *16-битные регистры* процессора ***8086***, обеспечивающие работу компьютера на *20-битной шине адреса*:

- *для выполнения арифметических и логических операций* служат регистры общего назначения: AX, BX, CX и DX;

- *индексные регистры* служат для формирования массивов; для этих же целей служат *указательные регистры*;
- *регистр состояния* содержит биты, которые изменяются в процессе выполнения различных операций;
- *сегментные регистры* являются указателями на начало областей оперативной памяти (сегменты);
- *указатель команды* — смещение относительно начала сегмента команд, определяемого регистром CS.

Регистры общего назначения

AH	AL	**AX** (primary accumulator)
BH	BL	**BX** (base, accumulator)
CH	CL	**CX** (counter, accumulator)
DH	DL	**DX** (accumulator, other functions)

Индексные регистры

SI	**S**ource **I**ndex
DI	**D**estination **I**ndex

Указательные регистры

BP	**B**ase **P**ointer
SP	**S**tack **P**ointer

Регистр состояния

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	(bit position)
-	-	-	-	O	D	I	T	S	Z	-	A	-	P	-	C	Флаги

Сегментные регистры

CS	**C**ode **S**egment
DS	**D**ata **S**egment
ES	**E**xtra**S**egment
SS	**S**tack **S**egment

Указатель команды

IP	**I**nstruction **P**ointer

Рисунок 1.1 - Регистры процессора 8086

В общем случае, для работы с регистрами процессора необходимо знать набор его команд, которые на практике пишутся на языке ассемблера.

Поскольку изучение ассемблера не входит в программу нашего обучения, то мы отметим основные качественные характеристики этого процессора:

- объем адресуемой памяти *1 Мбайт*;
- работа в реальном режиме: *защищённый режим работы отсутствует*.

1.1.2 Наборы регистров 64-битных процессоров

Преемником компании ***Intel***, для дешёвых процессоров x86, стала компания ***AMD***. Со временем, появились ***32-битные*** и ***64-битные*** процессора.

Начиная с процессора 80386, появился полноценный защищённый режим работы, который стал обеспечивать ядрам ОС *привилегированный режим работы*.

Для 64-битных процессоров и соответствующего ПО, совместимых с набором команд ***x86***, стало применяться обозначение ***x86-64***. На рисунке 1.2, показаны регистры процессора ***AMD64***.

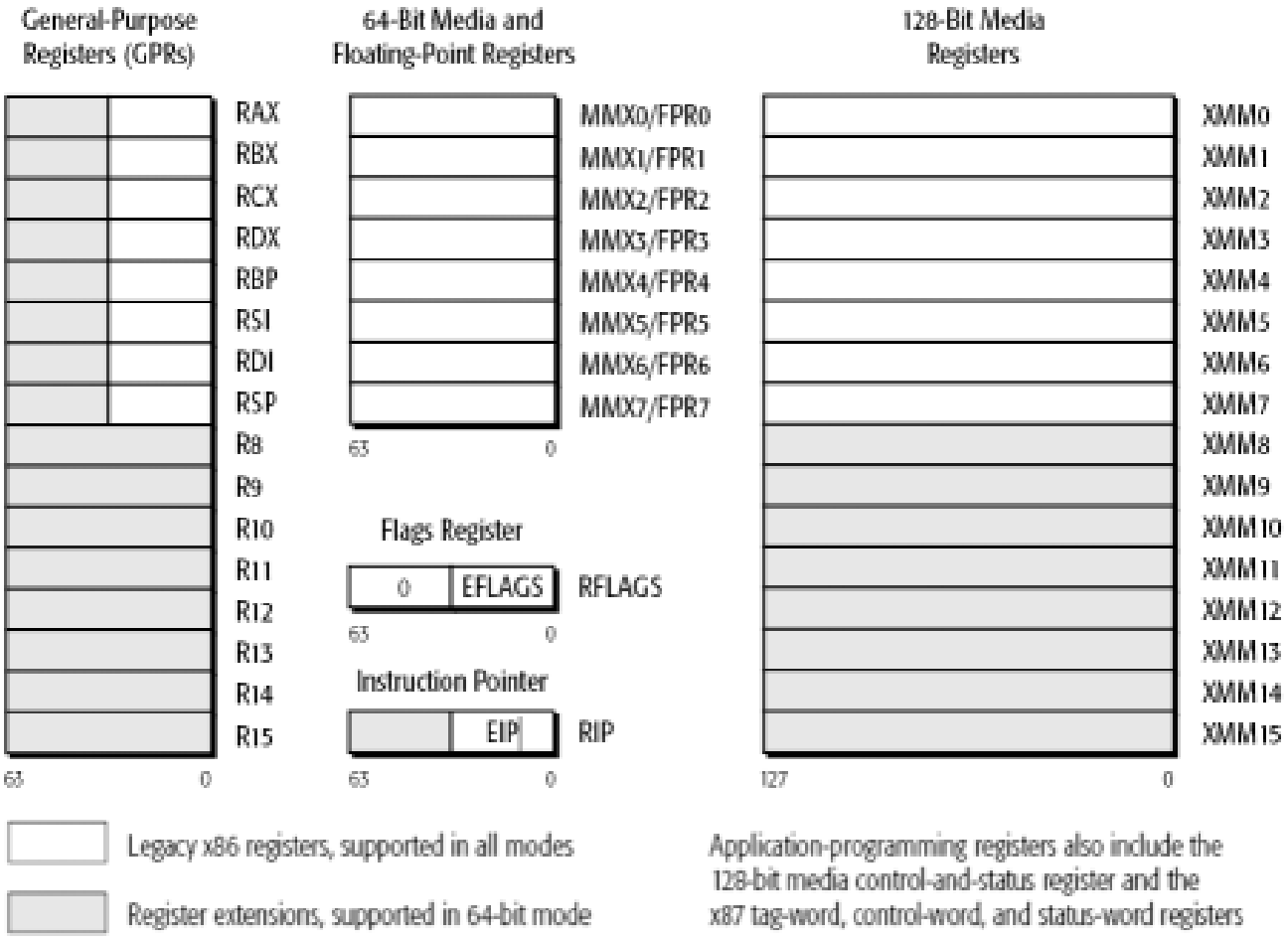

Рисунок 1.2 - Набор регистров процессоров x86-64

Замечание

Хотя 32-битное ПО может работать на процессорах x86-64, *64-битное ПО не может работать на процессорах x86*. Несмотря на расширение количества регистров и увеличение их размеров, имеется подмножество команд, которые выполняются на всех процессорах. Такой набор команд соответствует *процессору 80386*, а набор команд часто в дистрибутивах ОС обозначается как *i386*.

На рисунке 1.2 (б) показан полный набор регистров процессоров x86-64.

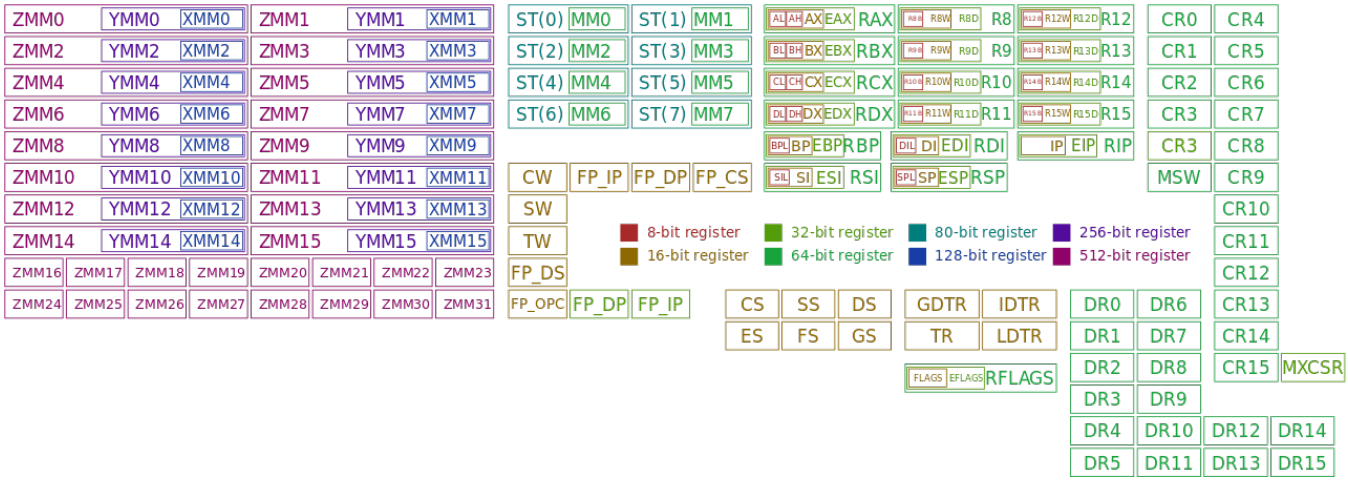

Рисунок 1.2 (б) — Полный набор регистров процессоров x86-64 (Википедия)

1.1.3 Вызов функций и прерывания

Среди всего множества команд, которые типично работают на всех процессорах и используют сегмент стека, *выделяются две*:

- *вызов функции* или команда: **CALL адрес**; в стек заносятся сегмент и смещение следующей за вызовом функции команды, а затем управление передаётся по адресу указанному в команде CALL; когда функция заканчивает работу, то из стека извлекаются сохранённые сегмент и смещение, которые помещаются в регистры CS и IP, обеспечивая продолжение работы программы;
- *прерывание* или команда: **INTERRUPT номер**; кроме сегмента и смещения команд, в стек заносятся регистр флагов, тем самым сохраняя полное состояние процессора; **номер** соответствует *номеру слова* в начале памяти ЭВМ, в которые предварительно записаны значения сегмента и смещения, соответствующие адресу обработчика прерывания; таким образом обычно обрабатываются аппаратные прерывания процессора.

1.1.4 Три способа ввода-вывода

Другим важным аспектом работы аппаратного обеспечения ЭВМ является *способ выполнения операций ввода-вывода*. Существует три таких способа:

- ***Программируемый ввод-вывод***, когда процессор посылает команды, связанные с ним контроллеру, а затем *периодически проверяет состояние модуля ввода-вывода с целью проверки завершения операции*.

- ***Ввод-вывод, управляемый прерываниями***, когда процессор посылает необходимые команды контроллеру ввода-вывода и продолжает выполнять текущий процесс, если нет необходимости в ожидании выполнения операции ввода-вывода. В ином случае, текущий процесс приостанавливается до получения

сигнала прерывания о завершении ввода-вывода, а процессор переключается на выполнение другого процесса. Наличие прерываний процессор проверяет в конце каждого цикла выполняемых команд.

- ***Прямой доступ к памяти (direct memory access – DMA)***. В этом случае, имеется специальный *аппаратный модуль прямого доступа к памяти*, который управляет обменом данных между основной памятью и контроллером ввода-вывода. При этом, процессор посылает запрос на передачу блока данных модулю DMA, а само прерывание происходит только после передачи всего блока данных.

В современных компьютерах используется прямой способ доступа к памяти, схема которого показана на рисунке 1.3.

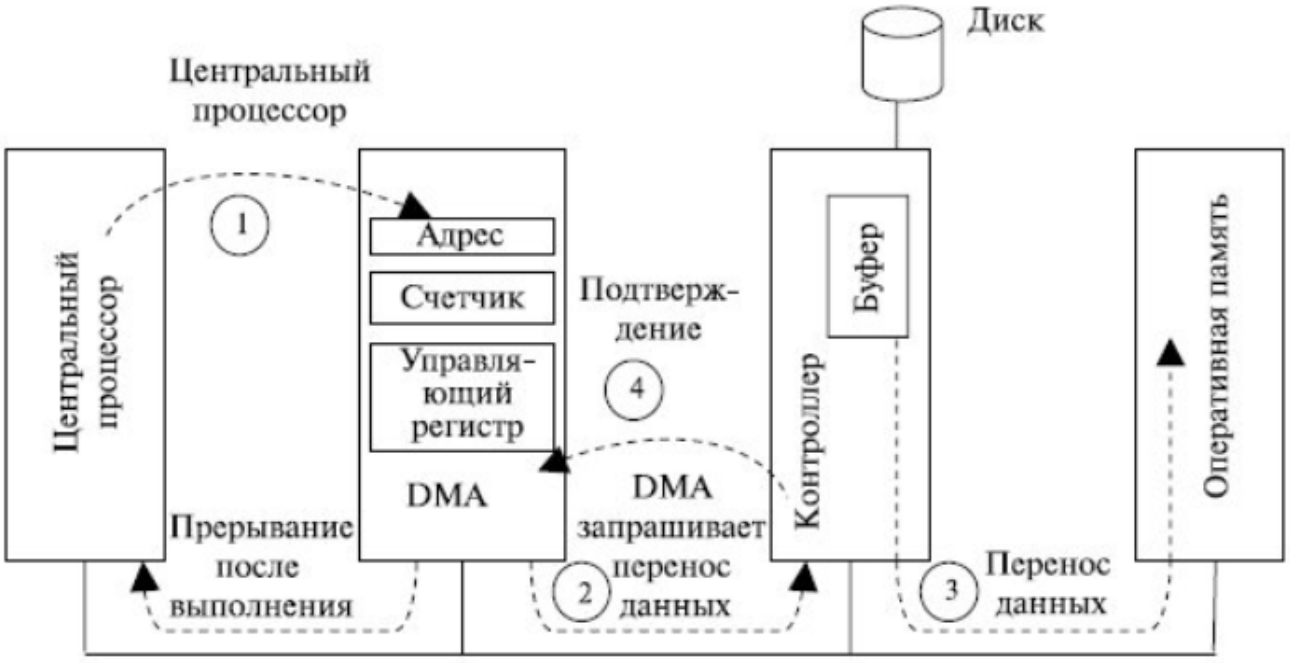

Рисунок 1.3 - Схема прямого доступа к памяти с помощью DMA

Подсистема ввода-вывода должна также учитывать *режим работы шины*, который может быть *пословным* или *поблочном*.

В пословном режиме, контроллер DMA выставляет запрос на перенос одного слова и получает его. Если процессору также нужна эта шина, ему приходится подождать.

Такой механизм называется ***захватом цикла***, потому, что контроллер устройства периодически забирает случайный цикл шины у центрального процессора, слегка тормозя его.

На рисунке 1.4, показана позиция цикла команд, в которых работа процессора может быть приостановлена. В любом случае, приостановка процессора происходит только при необходимости использования шины. После этого, устройство DMA

выполняет передачу слова и возвращает управление процессору.

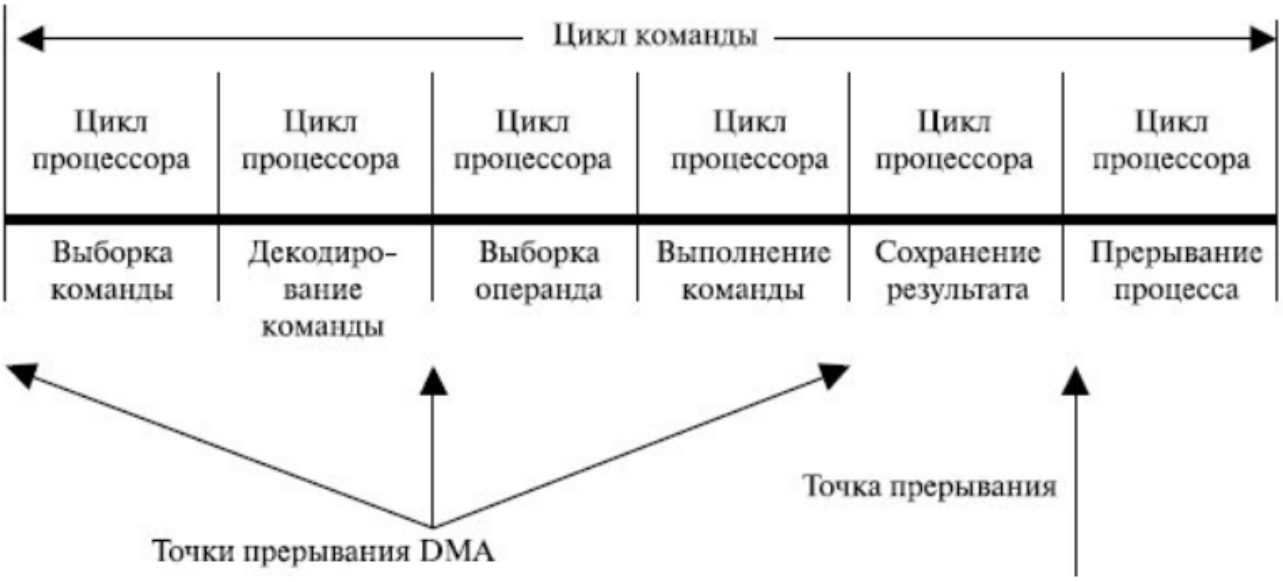

Рисунок 1.4 - Точки прерывания процессора устройством DMA

В блочном режиме, работы контроллер DMA занимает шину на серию пересылок (***пакет***). Такой режим более эффективен, однако при переносе большого блока данных центральный процессор и другие устройства могут быть заблокированы на существенный промежуток времени.

При большом количестве устройств, от подсистемы ввода-вывода *требуется спланировать свою работу в реальном масштабе времени*, в котором работают все внешние устройства, запуск и приостановку большего количества разных драйверов, обеспечив, при этом, время реакции каждого драйвера на независимые события контроллеров внешних устройств.

С другой стороны, необходимо минимизировать загрузку процессора задачами ввода-вывода.

Решение этих задач достигается на основе *многоуровневой приоритетной схемы обслуживания прерываний*: для обеспечения приемлемого уровня реакции, все драйверы распределяются по нескольким приоритетным уровням, в соответствии с требованиями по времени реакции и временем использования процессора.

Для реализации приоритетной схемы управления используется ***общий диспетчер прерываний ОС***.

Замечание

Более подробное рассмотрение алгоритмов взаимодействия подсистемы ввода-вывода с процессором, шиной и контроллерами внешних устройств требует знания и использования *диаграмм сигналов*, а также знания и практику работы с языком ассемблера. Все эти вопросы выходят за рамки нашего курса.

1.2 BIOS и его функции

BIOS (*Basic Input/Output System* **-** базовая система ввода-вывода) - часть системного программного обеспечения ЭВМ, реализованная в виде микропрограмм, и обеспечивающая для ОС доступ к материнской плате компьютера.

В персональных *IBM PC-совместимых компьютерах*, использующих процессоры *x86*, BIOS записана в микросхему *EEPROM (ПЗУ)* и обеспечивает:

- *начальное тестирование* компьютера;
- *последующую загрузку* ОС.

Имеется два типа перезаписываемых микросхем, хранящих BIOS:

- *микросхемы EPROM (Erasable Programmable Read Only Memory)*: содержимое этих микросхем может быть стёрто при помощи ультрафиолетового излучения специальным прибором (*старый вариант*);
- *микросхемы EEPROM (Electrically EPROM)*: содержимое этих микросхем может быть стёрто при помощи электрического сигнала, при этом микросхему не обязательно вынимать из компьютера.

Когда появились первые персональные компьютеры, необходимость в BIOS стала критической. Производители компьютеров стали использовать продукты BIOS трёх производителей: ***AMI***, ***AWARD*** *и* ***Phoenix***. Пользователю предоставляется «*Меню*», которое позволяет выполнить некоторые специальные настройки компьютера. Для обеспечения выполнения настроек все указанные фирмы используют текстовый режим монитора, который именуется *псевдографикой*. Поскольку основные настройки BIOS выполняются самими производителями компьютера, то обычному конечному пользователю следует использовать только две возможности:

- *установка приоритетов* загрузочных устройств;
- *установка адресов* дополнительных плат расширения компьютера.

Замечание

Как правило, другие настройки изменять *НЕ РЕКОМЕДУЕТСЯ*!

Для сохранения *настроек BIOS* используется микросхема CMOS-памяти.

CMOS - ***complementary metal-oxide-semiconductor*** - технология электронных схем или КМОП - *комплементарный металлооксидный полупроводник*.

Кроме настроек BIOS в CMOS хранятся *параметры конфигурации компьютера*. Суммарный объем памяти CMOS составляет ***256 байт*** и потребляет очень мало энергии. Стандартная батарейка, расположенная на материнской плате питает CMOS в течение ***5-6 лет***, после чего необходимо производить ее замену.

Замечание

Если срок батарейки, питающей CMOS, подошёл к концу, то при включении ЭВМ на экран будет выведено сообщение, например, "*CMOS-checksum error*". Для возобновления работы компьютера необходимо будет установить новую батарейку взамен вышедшей из строя.

В зависимости от *версии BIOS* и *модели материнской платы*, функции настройки BIOS могут меняться.

В разных версиях, одни и те же функции могут иметь разные названия.

Справочную информацию по настройке можно найти в инструкции к материнской плате или в сети Интернет.

Замечание

Программа настройки ***BIOS*** (*BIOS Setup*) может быть вызвана после перезагрузки компьютера нажатием определённой клавиши или группы клавиш.

Наиболее распространенные — **Del**, **F2** или **Esc**.

Существуют также определённые комбинации клавиш, позволяющие:

- *запустить микропрограмму восстановления* (перезаписи) BIOS в микросхеме в случае повреждения её, аппаратно либо вирусом;
- *восстановить заводские настройки*, позволяющие запустить компьютер после неверных настроек.

Замечание

Неверные настройки BIOS могут нарушить работу компьютера.

1.3 Этапы и режимы POST

Основную часть BIOS материнской платы составляют микропрограммы инициализации контроллеров на материнской плате. Подключённые к материнской плате устройства, в свою очередь, могут иметь *управляющие контроллеры с собственными BIOS*.

Сразу после включения питания компьютера, во время начальной загрузки компьютера, при помощи программ записанных в BIOS, происходит самопроверка аппаратного обеспечения компьютера — POST.

POST (*Power-On Self-Test*) — самостоятельное тестирование устройств компьютера после включения его электропитания.

Может использоваться *полный* или *сокращённый* тест.

Сокращённый тест, включает четыре этапа:

1. Проверку целостности программ BIOS в ПЗУ, используя контрольную сумму.
2. Обнаружение и инициализацию основных контроллеров, системных шин и подключённых устройств: графического адаптера, контроллеров дисководов и другие.
3. Выполнение программ BIOS, обеспечивающих самостоятельную инициализацию внешних устройств.
4. Определение размера оперативной памяти и тестирования первого ее сегмента: *64 Кбайт*.

Полный регламент работы POST:

1. Проверка регистров процессора;
2. Проверка контрольной суммы ПЗУ;
3. Проверка системного таймера и порта звуковой сигнализации;
4. Тест контроллера прямого доступа к памяти;
5. Тест регенератора генератора оперативной памяти;
6. Тест нижней области ОЗУ для проецирования резидентных программ в BIOS;
7. Загрузка резидентных программ;
8. Тест стандартного графического адаптера (VGA);
9. Тест оперативной памяти;
10. Тест основных устройств ввода (НЕ манипуляторов);
11. Тест CMOS - *Complementary Metal-Oxide-Semiconductor*;
12. Тест основных портов LPT/COM;
13. Тест накопителей на гибких магнитных дисках (НГМД);
14. Тест накопителей на жёстких магнитных дисках (НЖМД);
15. Самодиагностика функциональных подсистем BIOS;
16. Передача управления загрузчику ОС.

Замечание

Выбор между прохождением *полного* или *сокращённого* набора тестов, при включении компьютера, можно задать в программе настройки базовой системы ввода-вывода (*Setup BIOS*).

1.4 UEFI и его стандартизация

Для новых современных платформ ЭВМ, компания *Intel* предлагает **EFI** — *Extensible Firmware Interface*.

Первоначально, ***с середины 1990 годов***, EFI разрабатывалась для первых систем *Intel-HP Itanium*.

Позже, этот интерфейс был переименован в ***UEFI***, разработку которого продолжил *Unified EFI Forum*.

На данный момент, *последняя версия UEFI 2.9*, принятая ***в марте 2021 года***.

Обычно, UEFI имеет новый графический интерфейс, предполагающий *улучшить «реликтовый BIOS»*. Это стало возможным благодаря совершенствованию технологии изготовления микросхем EEPROM: увеличению их объёма и быстродействия, а также снижению себестоимости. Тем не менее, между функционированием BIOS и UEFI имеются существенные различия, которые необходимо хорошо знать. Рассмотрим это подробнее.

1.4.1 Программное обеспечение 16-битного BIOS

Процессор ***x86***, после включения питания ЭВМ, проводит самотестирование и начинает свою работу *в реальном режиме*, который обеспечивает ему доступ ко всем ресурсам компьютера. Обнулив все регистры, он выставляет значения ***CS*** и ***IP*** специальным образом:

- *для моделей до 80386-DX*: CS=0xFFFF, IP=0x0000 — что указывает на последние 16 байт в конце 1-го Мбайта оперативной памяти ЭВМ;
- *начиная с 80386-DX*: CS=0x0000, EIP=0xFFFFFFF0 — что указывает на последние 16 байт в конце 4-х Гбайт оперативной памяти ЭВМ;

После установки начальных значений регистров и захвата шины компьютера, процессор начинает выполнять команды извлекаемые из памяти ЭВМ и эта работа не прекращается до полной остановки самого процессора. Обычно, указанные 16 байт, содержат команду ***GOTO*** по адресу ПО BIOS, что поддерживается специальной микросхемой памяти, определённой аппаратным конструктивом компьютера. Таким образом, начинается работа любой современной ЭВМ.

ПО BIOS, начиная свою работу, делает небольшой тайм-аут выводит на экран подсказку, чтобы пользователь мог войти в режим настройки (***BIOS Setup***). Выполнив все программы POST, BIOS ищет загрузочное устройство ЭВМ, среди списка доступных, после чего запускает загрузочный код, расположенный в специальном секторе блочного устройства: MBR.

MBR — *Master Boot Record* — специальная структура загрузочного устройства, подробно рассмотренная далее.

Возможности ПО BIOS достаточно широки и не ограничиваются только перечисленными выше функциями POST, поиском загрузочного устройства и запуском программного кода MBR. Чтобы это показать, рассмотрим структуру ОЗУ ЭВМ.

1.4.2 Оперативная память 16-битного IBM PC компьютера

Типичная схема оперативной памяти (ОЗУ) IBM PC-совместимого компьютера показана на рисунке 1.5. В начале ее расположена *область векторов прерываний*, занимающая 1024 байта: по 4 байта на один вектор (всего 256 векторов).

Вектор прерывания — адрес программы в памяти ОЗУ (*обработчика прерывания*), которая будет исполняться процессором, когда такое прерывание произойдёт.

BIOS в начале работы, выставляет адреса этих векторов на своё собственное ПО, обеспечивая возможности полнофункционального управления компьютером.

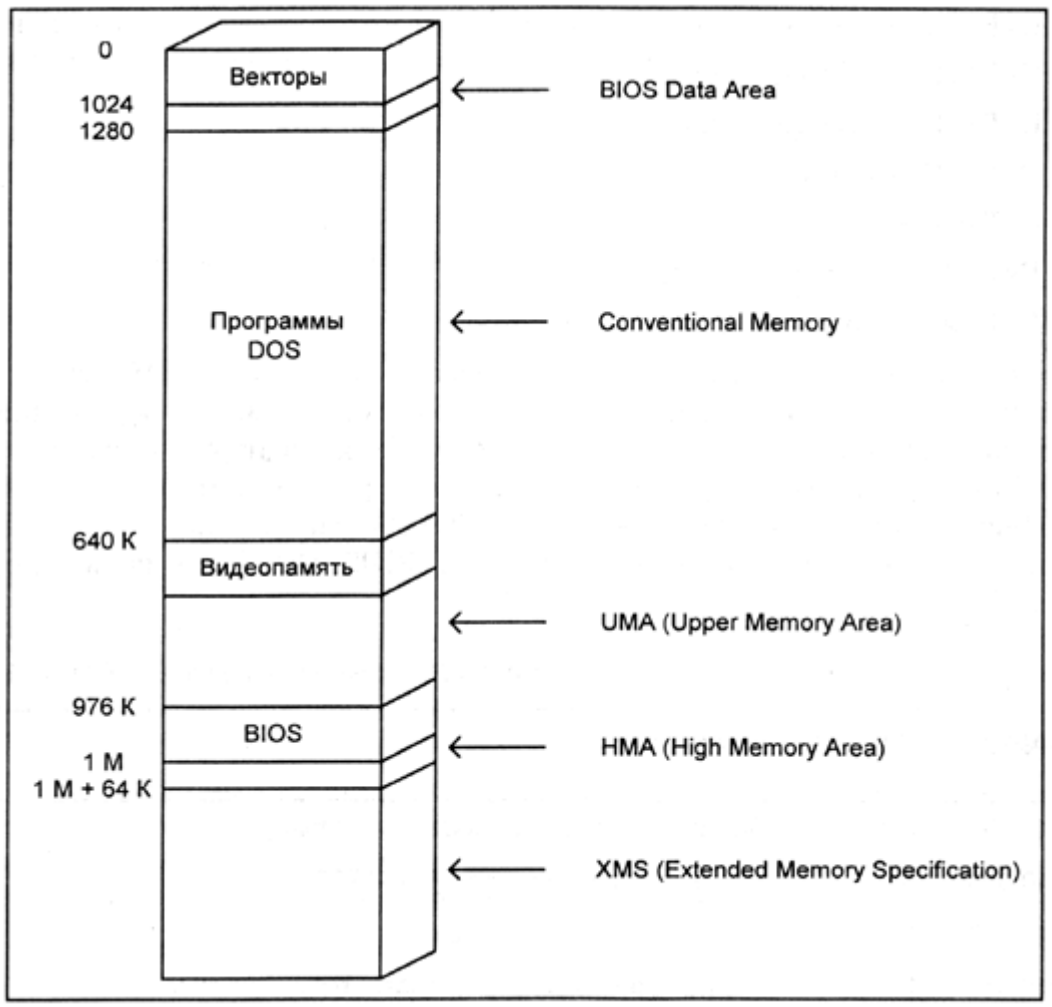

Рисунок 1.5 — Типичная схема ОЗУ памяти ЭВМ

Назначение ряда векторов прерываний IBM PC AT — следующее:

- **INT 00h** — деление на 0;
- **INT 01h**— пошаговый режим;
- **INT 02h**— немаскируемое прерывание;
- **TXTT ATI INT O3h** — точка останова;
- **INT 04h** — переполнение;
- **INT 08h** — таймер;
- **INT 09h** — клавиатура;

- **INT 10h** — видео сервис;
- **INT 33h** — поддержка мыши;
- **INT 4Ah** — будильник пользователя.

Первоначально, такие ОС, как MS DOS, использовали эти прерывания в своей работе. MSDOS имеет даже свой собственный вектор ***21h*** — функции DOS.

1.4.3 Отличительные особенности ПО UEFI

ПО UEFI интенсивно использует новейшие технологические возможности современных компьютеров. Хотя также выполняются функции аналогичные POST, UEFI сразу переводит процессор в защищённый режим работы, тем самым, обеспечивая повышенную надёжность работы ПО. Такой подход позволяет использовать сертифицированные подходы к использованию ЭВМ, буквально на этапе его включения. Кроме того, ПО UEFI способно работать с файловыми системами и более современной структурой блочных устройств — GPT.

GPT — *GUID Partition Table* — новая структура блочных устройств, позволяющая разбивать их на *128 основных разделов*, в отличие от структуры MBR, которая допускает наличие *только 4-х основных разделов*.

Можно выделить следующие основные особенности ПО UEFI:

- *работа в защищённом режиме* процессора;
- *возможность защищённой сертификатами* загрузки ОС;
- *возможность поддержки CSM* - *Compatibility Support Module* — модули, обеспечивающие загрузку ОС, совместимую с возможностями BIOS;
- *модульная организация ПО* поддержки аппаратных средств компьютера, *firmware*, которое можно собирать и устанавливать от производителей оборудования;
- *специальная структура ПО*, соответствующая запускаемым *exe-файлам* ОС MS Windows;
- *возможность загрузки множества* разных ОС;
- *непосредственный вызов* загрузчика ОС;
- *возможность работы* со структурой GPT блочных устройств;
- *требование наличия собственного специального раздела* для блочных устройств;
- *требование формата FAT12/16/32* для собственного раздела.

Замечание

1. Стали появляться ноутбуки с UEFI и предустановленной ОС MS Windows 8. При покупке такого компьютера, убедитесь, что основное окно UEFI имеет переключатель «*Legacy BIOS*» и отключение режима «*Security*», иначе вам не удастся загрузить другую ОС.
2. Для компьютеров на базе иных платформ, чем ***x86***, для обозначения встроенного ПО могут использоваться другие термины. Например, для архитектур *SPARC* или *UltraSPARC*, *firmware* может называться ***PROM*** или ***Boot***.

1.5 Блочные и символьные устройства компьютера

В предыдущей теме, перечисляя типы специальных файлов, мы отметили *файлы устройств*.

Все файлы устройств разделяются на ***символьные*** и ***блочные***:

- *блочными* называются внешние устройства компьютера, обмен данными с которыми можно производить только блоками: *целостной упорядоченной последовательностью байт*; к блочным устройствам относятся «жёсткие» и ***floppy***-диски, магнитные ленты, диски CDROM и другие; на блочных устройствах возможно создание *файловых систем*;
- *все другие устройства*, не являющиеся блочными, называются *символьными*; обмен данными с символьными устройствами осуществляется по *одному байту*; например, клавиатура, мышь, консоль экрана, COM-порты, сетевые устройства и другие — *символьные*.

Замечание

Магнитные ленты могут иметь *физические блоки переменной длины*.
«Жёсткие диски (винчестера)» имеют *физические блоки фиксированной длины*.
Текущий стандарт физического блока винчестера: *1 сектор — 512 байт*.
Символьное устройство *не обозначает текстового содержимого*.
Для символьных устройств, во многих случаях, понятие объёма хранения данных не применимо.

ОС MS Windows обозначает разделы блочных устройств, имеющих форматы FAT и NTFS буквами с двоеточием: *A:, B:, C:, ..., Z:*. Прописные и заглавные буквы — неразличимы. Символьные устройства обычно скрыты за графическим интерфейсом и, в явном виде, не используются.

ОС UNIX и Linux имеют общие правила обозначения устройств:

- имена устройств находятся в специальной директории ***/dev***; в нее смонтирована специальная область ядра ***dev*** с файловой системой типа *devtmpfs*;
- имена устройств имеют имя драйвера, которое управляет этим устройством; прописные и заглавные буквы различаются;
- имена устройств, объединённых одним драйвером, разделяются цифрой, добавляемой к имени драйвера, начиная с нуля.

Замечание

При наличии соответствующих драйверов, в ОС UNIX и Linux, можно с блочным устройством работать как с символьным, поэтому понятия блочный и символьный применимы и к драйверам, управляющим устройствами.

1.6 Винчестер и загрузочные устройства

Традиционно, загрузочным устройством ЭВМ является *винчестер* или *«жёсткий диск»*.

1.6.1 Цилиндр, головка, сектор (CHS) - блочная адресация жёсткого диска

Конструктивно, винчестер состоит из набора круглых пластин, которые центральной частью, на некотором расстоянии, надеты на шпиндель, вращающийся посредством электродвигателя:

- *каждая сторона круглой пластины* покрыта магнитным составом, способным фиксировать информацию посредством *магнитных головок*, которые «плавают» над каждой стороной диска;
- *отдельная окружность* на отдельной стороне диска образует *трек (track)*;
- *совокупность треков одного диаметра*, образуют *цилиндр (cylinder)*; цилиндры пронумерованы от внешнего края диска, *начиная с 0*;
- *все магнитные головки винчестера (head)* одновременно находятся над треками одного цилиндра и пронумерованы от 0 (*обычно от 0 до 15*);
- *каждый трек разделен на 63 части* — *сегменты (физический блок)*, пронумерованные, *начиная с 1*.

Таким образом, физические блоки винчестера (*сегменты*) пронумерованы в системе координат ***CHS***, начиная с сегмента (*0, 0, 1*).

Замечание

Следует отметить, что *адресация CHS* (*Цилиндр, Головка, Сектор*), заложенная в конструкцию первых персональных компьютеров и ПО BIOS, не позволяет адресовать *более 7.8 Гбайт* данных. Поэтому современные ЭВМ, имеющие винчестера ёмкостью более 7.8 Гбайт, используют *адресацию LBA*.

1.6.2 Блочная адресация LBA

LBA (*Logical block addressing*) — механизм адресации и доступа к блоку данных на на «жёстком диске», при котором *системному контроллеру* нет необходимости учитывать геометрию самого жёсткого диска: количество цилиндров, сторон и секторов на цилиндре. Контроллеры современных IDE дисков в качестве основного режима трансляции адреса используют LBA.

Суть LBA состоит в том, что *каждый блок, адресуемый на жёстком диске, имеет свой номер* - целое число, *начиная с нуля* и далее:

LBA 0 = Цилиндр 0/Головка 0/Сектор 1

Преимущество метода адресации LBA — ограничение размера диска обусловлено лишь разрядностью LBA. В настоящее время, для задания номера блока используется *48 бит*, что даёт возможность адресовать (2^{48}) 281 474 976 710 656 блоков.

Технический комитет X3T10 установил правила получения адреса блока в режиме LBA, при условии, что размер блока равен размеру сектора:

$$LBA(c,h,s)=(c*H+h)*S+s-1$$
$$s=(LBA\ mod\ S)+1$$
$$h=\frac{(LBA+1-s)\ mod\ (H*S)}{S}$$
$$c=\frac{LBA+1-s-h*S}{H*S}$$

где:
c — номер текущего цилиндра;
h — номер текущей головки;
s — номер текущего сектора;
H — число головок;
S — число секторов на дорожке;
mod — операция взятия остатка от деления.

После того как BIOS закончит начальный тест POST, BIOS начнёт просматривать блочные устройства ЭВМ с целью поиска *загрузочного устройства* ОС:

- Блочные устройства просматриваются в том порядке, который указан в настройках BIOS. Чтобы определить является ли устройство загрузочным, BIOS читает первый сектор блочного устройства и помещает его в ОЗУ ЭВМ. В компьютерах архитектуры IBM PC, этот адрес обычно ***0000:7c00***.
- Если сектор соответствует MBR — *Master Boot Record*, то BOIS передаёт управление его загрузочному коду: обычно командой *long jump*.
- Если структура прочитанного сектора не соответствует MBR, то проверяется следующее устройство.
- Если все просмотренные устройства не являются загрузочными, то BIOS: или перезапускает ЭВМ или загружает встроенный в BIOS интерпретатор языка BASIC (если он, конечно, — есть).

Замечание

BIOS рассматривает flashUSB как загрузочное устройство, если:

- его ПО поддерживает такие устройства;
- имеется раздел MBR, который отмечен как загрузочный.

UEFI рассматривает flashUSB как загрузочное устройство, если оно имеет раздел, форматированный как FAT12/16/32, и в корне раздела имеется директория *EFI*.

1.7 Загрузочные сектора MBR и GPT

***MBR* (*Master Boot Record)* —** это *Главная загрузочная запись* блочного устройства, — это программный код и данные, расположенные в первом секторе блочного устройства и используемые для загрузки некоторой ОС.

В общем случае, под *загрузчик MBR* выделено ***32 Кбайт*** винчестера или другого внешнего блочного накопителя. Если под загрузчик ОС используются все 32 Кбайт, то под MBR понимают весь этот загрузочный код. В этом случае, ***первые 512 байт*** называют *MBS — Master Boot Sector* или *главным загрузочным сектором*.

Для операционных систем *MS Windows*, понятия MBR и MBS совпадают, так как вся MBR содержится в MBS и они рассматриваются как синонимы.

Замечание

MBR может не содержать загрузочного кода, если блочное устройство не является загрузочным. Более того, сам термин появился в те времена, когда:

- с одного устройства загружалась только одна ОС;
- структура блочного устройства была уникальна для каждой ОС.

Последующая унификация структур блочных устройств и самих загрузочных записей привели к тому, что ***MBR*** — это ещё не загрузка ОС, а всего лишь выбор: *«с какого раздела жёсткого диска следует загружать ОС»:*

- *На стадии MBR* происходит только выбор раздела диска и ничего более.
- *Загрузка самой ОС* происходит на более поздних этапах.

1.7.1 Общая структура MBR

Структура MBR содержит три основные части (см. таблицу 1.1):

- *небольшой фрагмент* исполняемого кода, - 446 байт;
- *таблицу* разделов (*Partition table*), - 64 байт;
- *специальную сигнатуру*, - 2 байта.

Таблица 1.1 - Структура Главной загрузочной записи (MBR)

Адрес	***Содержимое***
0x0000	Код загрузчика (446 байт), включая чеыре байта следующей строки
0x01B8	4-х байтная сигнатура диска (только для MS Windows 2000 и XP)
0x01BE	Четыре 16-байтных записи схемы таблицы основных разделов MBR **(Partition table)**
0x01FE	2-х байтная сигнатура MBR (0x55AA)

Поскольку утверждённого стандарта на структуру MBR не существует, то используется «*стандарт де-факто*», распространённый Microsoft, и которого придерживаются большинство дистрибьюторов ОС.

Согласно «*традиции MBR*», винчестер может быть *разбит на четыре основных раздела*. Допускается один из разделов использовать как *расширенный раздел* и делить его дополнительно. Традиционно также, *MBR создаётся или редактируется*

в момент инсталляции ОС на внешний носитель.

Когда BIOS прочитает первый сектор блочного устройства и запишет его по адресу ***0000:7C00***, она проверяет наличие сигнатуры ***55AAh***:

- Если сигнатура ***есть***, управление передаётся коду загрузчика MBR;
- Если сигнатуры **нет**, то проверяется следующее блочное устройство.

Код загрузчика MBR:

- *копирует себя* с адреса ***0000:7C00*** по адресу ***0000:6000***, освобождая место для непосредственного загрузчика ОС;
- *работает с таблицей разделов* (Partition table), структура отдельной строки которой показана в таблице 1.2: *если загрузочный раздел найден*, то первый сектор загрузчика ОС записывается по адресу ***0000:7C00*** и ему передаётся управление; *если загрузочный раздел не найден* или обнаружена ошибка записи *partition table*, то делается прерывание **INT 18h** и управление передаётся назад в BIOS.

1.7.2 Структура отдельной записи Partition table

Первый байт Partition table содержит ***признак активности раздела***: признак, обозначающий возможность загрузки операционной системы с данного раздела. Для стандартных загрузчиков может принимать следующие значения:

- **80h** — раздел является активным;
- **00h** — раздел является неактивным;
- ***Другие*** значения являются ошибочными и игнорируются.

Следующие три байта задают начало раздела в системе координат (***C,H,S***).

Пятый байт обозначает *код файловой системы:* Partition Ids, некоторые значения которого приведены в таблице 1.3.

Следующие три байта задают окончание раздела в координатах (***C,H,S***).

Завершают строку *Partition table* два четырёхбайтовых числа, задающие начало раздела и его длину в секторах.

Таблица 1.2 - Структура описания раздела

Смещение	***Длина***	***Описание***
00h	**1**	***Признак активности*** раздела
01h	1	Начало раздела — головка
02h	1	Начало раздела — сектор (биты 0-5), дорожка (биты 6,7)
03h	1	Начало раздела — дорожка (старшие биты 8,9 хранятся в байте номера сектора)
04h	**1**	***Код типа раздела*** – код файловой системы
05h	1	Конец раздела — головка
06h	1	Конец раздела — сектор (биты 0-5), дорожка (биты 6,7)
07h	1	Конец раздела — дорожка (старшие биты 8,9 хранятся в байте номера сектора)
08h	**4**	**Смещение** первого сектора раздела блочного устройства
0Ch	**4**	**Количество** секторов раздела

Таблица 1.3 - Ранее распространённые коды типов файловых систем

ID (hex)	***Описание***
01	Primary DOS12 (12-bit FAT)
04	Primary DOS16 (16-bit FAT)
05	Extended DOS
06	Primary big DOS (> 32MB)
0A	OS/2®
83	Linux (EXT2FS)
A5	FreeBSD, NetBSD, 386BSD (UFS)

Замечание

Допускается, чтобы один из разделов блочного устройства имел код типа файловой системы равный ***05h***, который соответствует структуре раздела ***EBR — Extended Boot Record***, начинающегося со структуры, приведённой в таблице 1.4.

Таблица 1.4 - Структура EBR

Смещение	***Длина***	***Описание***
1BEh	16	Указатель на раздел
1CEh	16	Указатель на следующий EBR
1FEh	2	Сигнатура (55AAh)

Замечание

Формат указателей в таблице 1.4 аналогичен формату строки *Partition Table* в MBR.

Раздел EBR не может содержать в себе других разделов EBR.

Традиционная таблица разделов винчестера MBR ориентирована на загрузку только одной ОС.

1.7.3 Общая структура GPT

Дальнейшее развитие структуры блочных устройств связано с созданием новой усовершенствованной таблицы разделов: **GPT**, показанной на рисунке 1.6.

GPT — ***GUID Partition Table***.

GUID — Global Unique IDentifier — глобальный уникальный идентификатор размером ***16 байт***, который в данном контексте используется для именования разделов блочных устройств:

- *само устройство* именуется в момент создания на нем структуры GPT;
- *раздел устройства* именуется в момент создания на нем файловой системы.

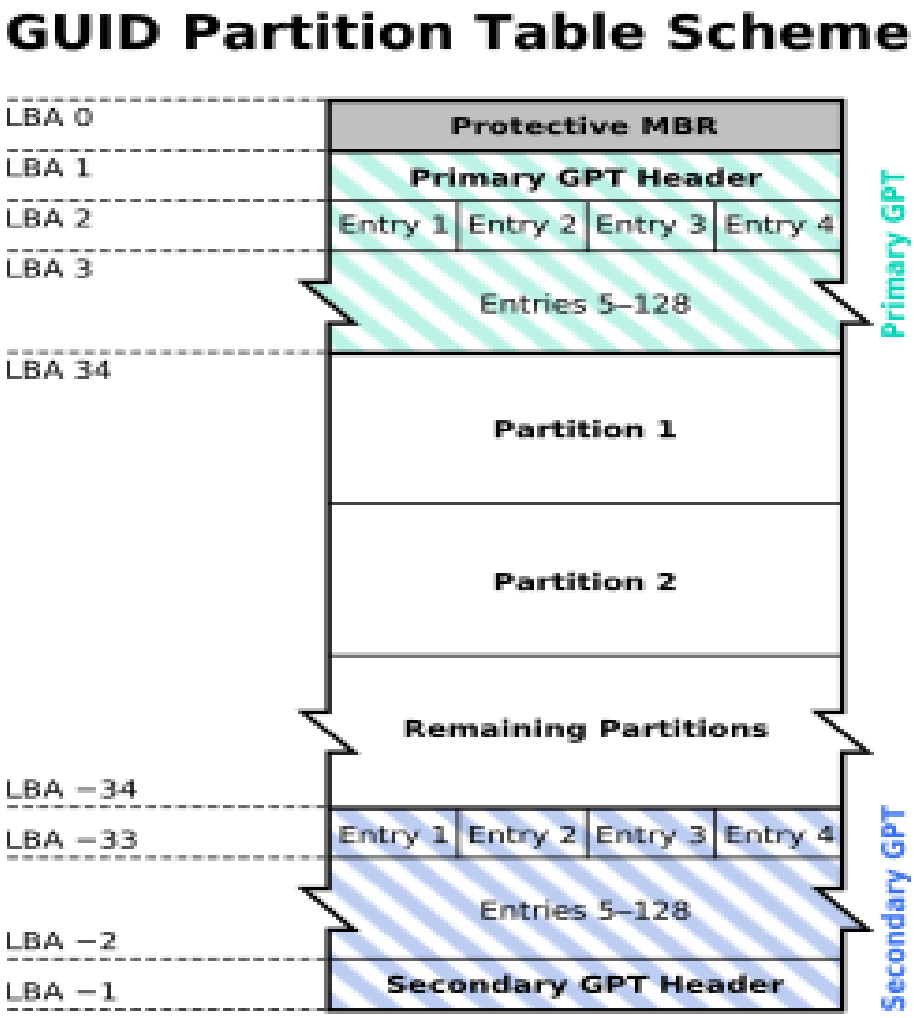

Рисунок 1.5 - Структура таблицы разделов GPT

Структура блочного устройства — GPT, обеспечивает следующие его свойства:

а) Каждое блочное устройство может быть разбито на ***128 разделов***, записи о которых дублируются в конце устройства **(***отрицательные номера LBA***)**.
б) Все разделы GPT являются основными.
в) Размер ***LBA=512 байт*** и ***LBA0*** — для совместимости соответствует *MBR*.
г) Отдельный LBA содержит *записи о 4-х разделах*: по 128 байт на раздел.
д) Для идентификации раздела используются GUID.

Замечание

MBR, входящий в структуру GPT, должен указывать наличие на блочном устройстве только одного раздела с типом ***eeh*** и размером:

- *всего блочного устройства*, для устройств < 2 ТБайт;
- *2 ТБайт*, для устройств > 2 ТБайт.

Не все BIOS могут работать со структурой GPT.

1.8 GRUB как универсальный загрузчик ОС

Проблема загрузки разных ОС с одного блочного устройства всегда будоражила умы пользователей ЭВМ, тем более, что ёмкость винчестеров постоянно увеличивалась. Хотя появились UEFI и структура блочных устройств GPT, имеется ряд проблем, в том числе и организационных, которые делают необходимым применение различных универсальных загрузчиков.

1.8.1 Примеры загрузчиков ОС

Наиболее распространёнными загрузчиками ОС являются:

- NTLDR— загрузчик ядра MS Windows NT;
- Windows Boot Manager (bootmgr.exe, winload.exe) — загрузчик ядра MS Windows Vista; *bootmgfw.efi — для MS Windows 7, 8, 8.1 и других*;
- LILO (LInux LOader) — старый загрузчик ядра Linux;
- *GRUB* (Grand Unified Bootloader) — новый загрузчик ядра Linux и Hurd;
- RedBoot — загрузчик для встраиваемых систем;
- SILO (SPARC Improved bootLOader) — загрузчик Linux и Solaris для машин с архитектурой SPARC;
- Loadlin — загружает Linux из под MS DOS или MS Windows;
- Syslinux — загружает Linux из под MS DOS или MS Windows;
- BOOTP — применяется для загрузки по сети.

Среди перечисленного ПО, наиболее интересным является ***GRUB*** — официальный загрузчик Linux из проекта GNU. Он может загружать разные ОС, включая MS Windows, с многих разных аппаратных платформ. GRUB (точнее ***GRUB2***) входит и в дистрибутив ***Arch Linux***. Он устанавливается на ЭВМ в процессе инсталляции ОС. Это позволяет свободно использовать и Linux и MS Windows на одном компьютере.

Более подробно, вопросам работы с ПО GRUB посвящена лабораторная работа №2, которая входит в программу обучения по данной теме. В качестве дополнительного учебного материала, которым необходимо пользоваться как справочной информацией, является пособие [4, раздел 2].

1.8.2 Меню и функции GRUB

ПО GRUB можно рассматривать как маленькую однопользовательскую ОС специального назначения: *интерактивная загрузка различных ОС*.

Многие идеи этого ПО и, в частности модульная организация, использовались разработчиками ***UEFI***.

Среди основных функций GRUB следует выделить:

- *поддержку интерпретатора сценариев*, близкого по функциональным возможностям к языку ***shell***, который собственно говоря и был его прототипом;
- *умение работать со структурами* ***MBR*** и ***GPT*** блочных устройств;
- *поддержка работы* со многими современными устройствами ЭВМ;

- *распознавание и умение работать* со многими современными файловыми системами;
- *поддержка национальных языков* и других мультимедийных средств ЭВМ;
- *обнаружение на блочных устройствах ЭВМ* наличия различных ОС и автоматическое формирование для них сценария меню загрузки.

Хотя не все функции GRUB работают одинаково эффективно, потенциал этого ПО является очевидным.

Основная часть ПО GRUB располагается в разделе блочного устройства и устанавливается в процессе инсталляции на него ОС Linux. По умолчанию, оно помещается в директорию */boot/grub*, туда же помещается автоматически созданный файл конфигурации *grub.cfg*.

Дополнительно, в процессе инсталляции ОС анализируется наличие ***UEFI*** и выполняется необходимая работа с ним.

Важно помнить, что дистрибутивы ***GRUB*** различаются для разных архитектур процессора ЭВМ, хотя допускается их смешанная установка на один компьютер.

Имеются варианты дистрибутивов для работы с ЭВМ, имеющими ***UEFI***.

Замечание

При повторной инсталляции разных дистрибутивов ОС будет установлена и соответствующая версия ПО GRUB и частично могут быть изменены уже имеющиеся его настройки. В первую очередь, это касается файла *grub.cfg*, поэтому следует заранее позаботиться о его сохранности.

Если на ЭВМ установлено несколько дистрибутивов ПО GRUB, то в процессе загрузки будет работать только один из них, *хотя файл grub.cfg* ***может быть общий!***

Одним из вариантов использования ПО GRUB является установка его на личное устройство ***flashUSB*** студента, что и применяется в процессе обучения по нашей дисциплине.

Такой подход позволяет избежать многих проблем, связанным с недостаточной квалификацией исполнителей.

2 Лабораторная работа №2

Учебная цель данной лабораторной работы - закрепление теоретических знаний и получение практических навыков по теме 2 «*BIOS, UEFI и загрузка ОС*».

Основным учебным пособием для данной работы, является [*4, раздел 2*].

Данный раздел учебного пособия содержит материал касающийся основных организационных мероприятий и обеспечивающий успешное выполнение работы.

Ограничения, применительно к которым изложен данный методический материал:

- студент выполняет работу в учебном классе кафедры АСУ на компьютерах с установленным ПО ОС УПК АСУ, имеющих базовое ПО BIOS и структуру загрузочных устройств MBR;
- студент имеет flashUSB с установленным ПО GRUB2 и архивом рабочей области пользователя ***upk*** по данной дисциплине;
- студент успешно выполнил лабораторную работу №1 и способен запустить ОС УПК АСУ, подключить к ней архив, войти в сеанс пользователя ***upk***, найти местоположение учебного материала и запустить на чтение данное пособие, находящееся в файле *Тема2_os.pdf*.

Исходные организационные мероприятия:

- *студенты разбиваются парами* (можно больше), каждая из которых имеет устройство flashUSB с минимальным объёмом ПО, подготовленным преподавателем на лабораторной работе №1;
- *каждой паре доступна* ЭВМ с запущенным ПО ОС УПК АСУ и учебный материал данного руководства;
- *вопросы вариантов* выполнения данной работы разрешаются преподавателем индивидуально.

2.1 Установка ПО GRUB на устройство flashUSB

Изучается три подраздела методического пособия [4, подразделы 2.1-2.3].

Изучается подраздел 2.4 методического пособия [4], учитывая возможные особенности исполнения работ в среде пользователя ***upk***.

Выполняется установка ПО GRUB2 на flashUSB.

Пары разделяются по отдельным компьютерам, запускают ОС УПК АСУ и заполняют личные отчёты по выполненной части работ. Отчёт заполняется в плане всех вопросов, изложенных в пособии [4, подразделы 2.1-2.4].

Проводится индивидуальный перезапуск ОС УПК АСУ с целью проверки качества исполнения проведённой части работы.

Замечание

По результатам выполнения этой части работы, все студенты должны иметь личные flashUSB с установленным на них рабочим ПО GRUB2.

2.2 Создание аварийного варианта ОС УПК АСУ

Задание

- *Изучается* подраздел методического пособия [4, подраздел 2.5].
- *Выполняется установка* ПО аварийного варианта ОС УПК АСУ на ***flashUSB*** студента.
- *Заполняется личный отчёт* по выполненной части работы, в плане всех вопросов, изложенных в пособии [4, подраздел 2.5].

Технология выполнения задания пользователем ***upk***:

- *провести*, средствами ОС УПК АСУ, извлечение личного ***flashUSB*** из компьютера;
- *вставить* личный ***flashUSB*** в разъем компьютера, а когда запустится файловый менеджер ***Thunar***, - закрыть окно менеджера;
- *запустить* виртуальный терминал, в котором командой ***mc*** запустить файловый менеджер «***Midnight Commander***»;
- *в левом окне* менеджера перейти в директорию ***/run/basefs/asu64upk***, а в правом окне - в директорию ***/run/media/upk/<UUID-утройства>/asu64upk***;
- *из левого* окна файлового менеджера *в правое окно* скопировать две директории: ***boot*** и ***upkasu*** (вместе с содержимым этих директорий);
- *закрыть* файловый менеджер, *отмонтировать и извлечь* ***flashUSB***, *завершить* работу с ОС УПК АСУ, выключив компьютер;
- *проверить* установку аварийного варианта ОС, используя для загрузки второй пункт меню ***GRUB***.

Замечание

Аварийный вариант загрузки ОС УПК АСУ осуществляется в тестовом режиме, поэтому, когда на экране компьютера появится подсказка ***login:***, следует ввести имя и пароль пользователя **asu**.

Дополнительно, из командного режима можно попытаться запустить графическую оболочку командой:

```
startxfce4
```

Помните, что аварийный вариант загрузки используется в случае проблем с загрузкой основного варианта, например, в случае отсутствия нужного драйвера графической подсистемы ОС.

2.3 Практика настройки файла конфигурации grub.cfg

Задание

- *Изучается* подраздел методического пособия [4, подраздел 2.6].
- *Выполняются эксперименты* с настройками файла ***grub.cfg***.
- *Студенты*, установившие аварийный вариант ПО ОС УПК АСУ на свою ЭВМ, выполняют модификацию первого пункта меню в файле ***grub.cfg***, применительно к запуску ОС со своего комьютера, и проводится экспериментальная проверка выполненной работы.
- *Заполняется личный отчет* по выполненной части работы, в плане всех вопросов, изложенных в пособии [4, подраздел 2.6].
- *Выполняется* создание архива и его перенос на личный ***flashUSB***, а также все мероприятия по завершению лабораторной работы №2.

Замечание

Студенты, выполняющие работу на личных ЭВМ, имеющих UEFI, выполняют эту часть работы с преподавателем.

Список использованных источников

1 Резник В.Г. Операционные системы. Самостоятельная и индивидуальная работа студента. Учебно-методическое пособие. – Томск, ТУСУР, 2016. – 13 с.

2 Гордеев А.В. Операционные системы: учебное пособие для вузов. ─ СПб.: Питер, 2004. ─ 415с.

3 Таненбаум Э., Бос Х. Современные операционные системы. - СПб.: Питер, 2015. - 1120с.

4 Резник В.Г. Учебный программный комплекс кафедры АСУ на базе ОС ArchLinux. Учебно-методическое пособие. – Томск, ТУСУР, 2017. – 38 с.

5 Поль Коббо. Фундаментальные основы Linux, 2014/Перевод А. Панина. - [Электронный ресурс]. - Режим доступа: - Фундаментальные основы Linux__by Paul Cobbaut .pdf.