

ОПЕРАЦИОННЫЕ СИСТЕМЫ v2015

1. Климентьев Константин Евгеньевич

2. 16 лекций, 4 лабораторных работы (512 ауд. 14 корп.), экзамен

3. Литература

3.1. По принципам организации операционных систем

- Таненбаум «Современные операционные системы»
- Гордеев, Молчанов «Системное программное обеспечение»
- Олифер, Олифер «Сетевые операционные системы»

3.2. По программированию на языке ассемблера

- Юров «Assembler. Учебный курс» и «Assembler. Практикум»
- Аблязов «Программирование на ассемблере на платформе x86-64»
- Магда «Ассемблер для процессоров Intel Pentium»
- Калашников «Ассемблер – это просто»

3.3. По системному программированию в Windows

- Харт «Системное программирование в среде Win32»
- Рихтер «Windows для профессионалов»
- Щупак «Win32 API. Эффективная разработка приложений»

3.4. По системному программированию в UNIX

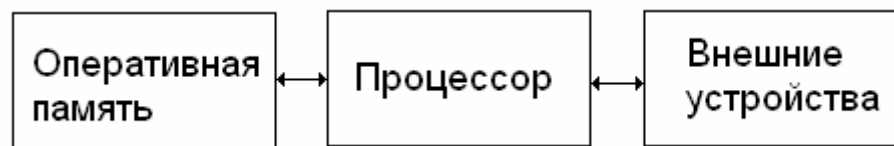
- Робачевский «Операционная система UNIX»
- Митчелл «Программирование для Linux. Профессиональный подход»

«Классическая» ЭВМ со внешними устройствами

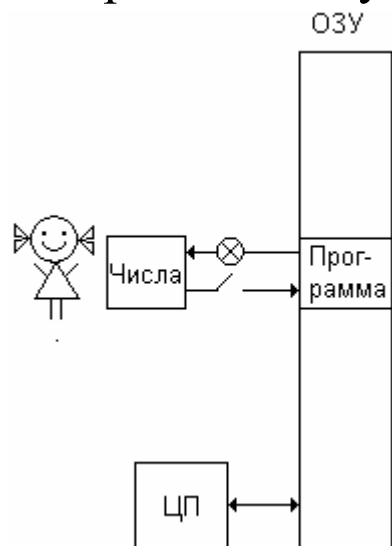


Предпосылки и история операционных систем

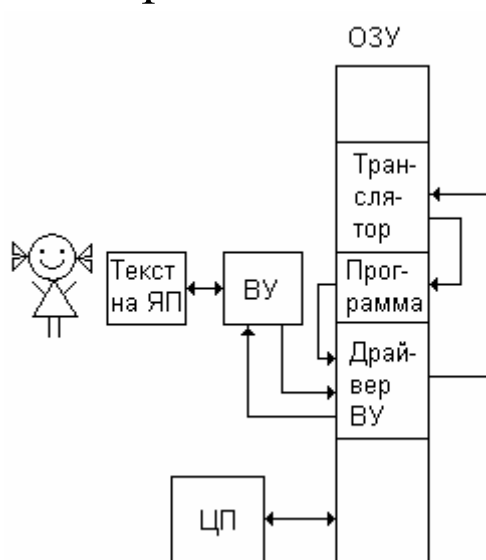
1. Упрощенная функциональная схема ЭВМ



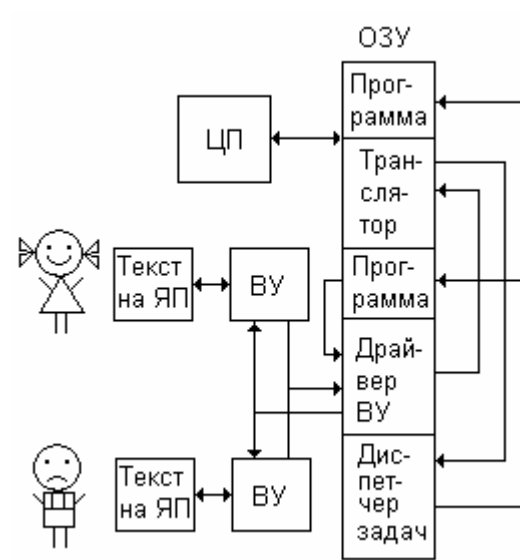
2. Прямой доступ



3. Трансляция



4. Многозадачность



ОПЕРАЦИОННАЯ СИСТЕМА – комплекс управляющих и

обрабатывающих программ, который решает 3 класса задач:

- 1) Обслуживание взаимодействия приложений и оборудования;
- 2) Обслуживание взаимодействия пользователя и приложений;
- 3) Рациональное распределение вычислительных ресурсов (памяти, процессорного времени, доступа ко внешним устройствам, etc.).

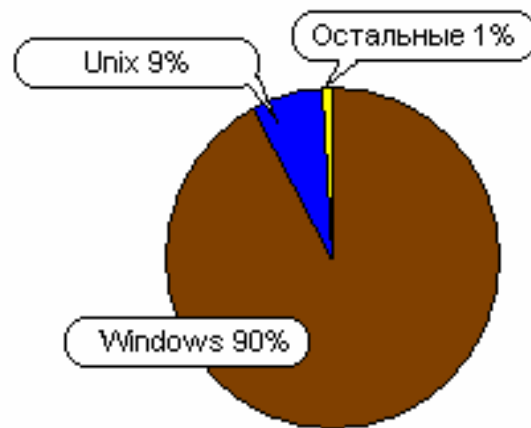
Классификация операционных систем по назначению

1. Офисные ОС (MS Windows, клоны UNIX, OS-2, ReactOS и пр.).
2. ОС реального времени (QNX, OS-9/9000, VxWorks и пр.).
3. Специализированные ОС.
 - 3.1. Для мобильных устройств (Android, Apple iOS, Symbian и пр.).
 - 3.2. Для сетевых устройств (Cisco IOS).
 - 3.3. Для мэйнфреймов (IBM System/360, IBM System/370, z/OS).
 - 3.4. Для мини-ЭВМ (RSX-11, RT-11, VMS и OpenVMS).

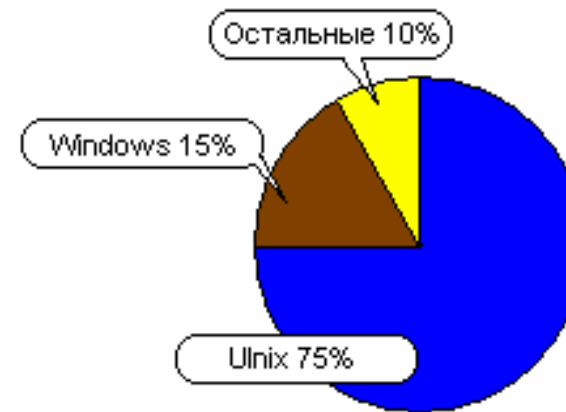
Сертифицированные отечественные ОС: MCBC (офисная) и ОС2000 «Багет» (реального времени).

Распространенность ОС

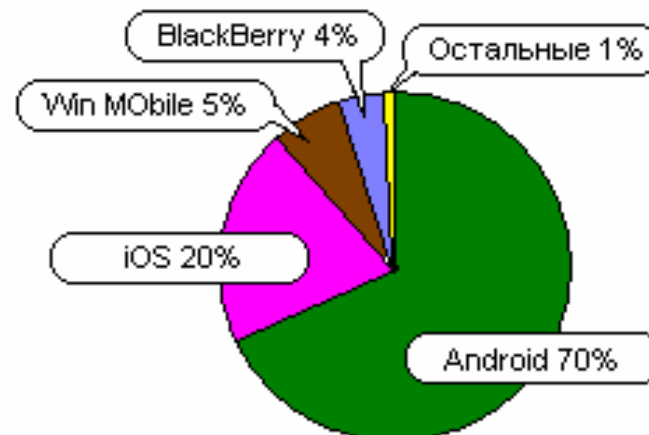
Рабочие станции



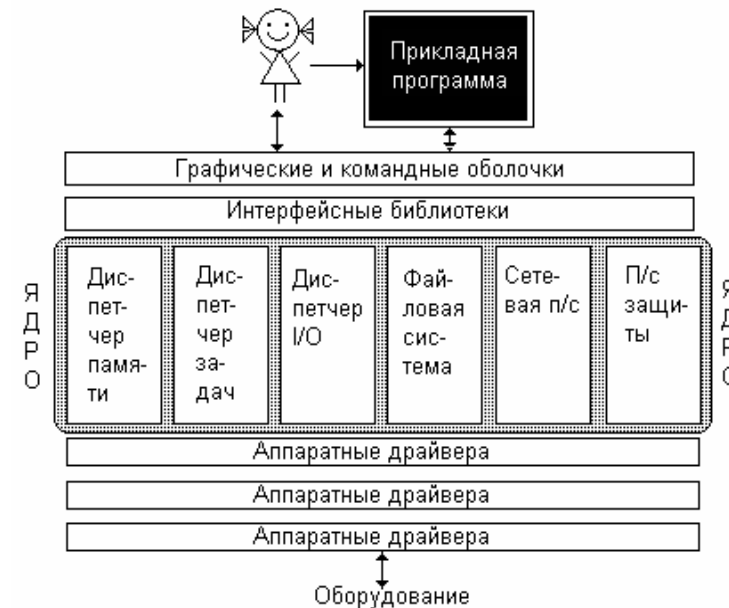
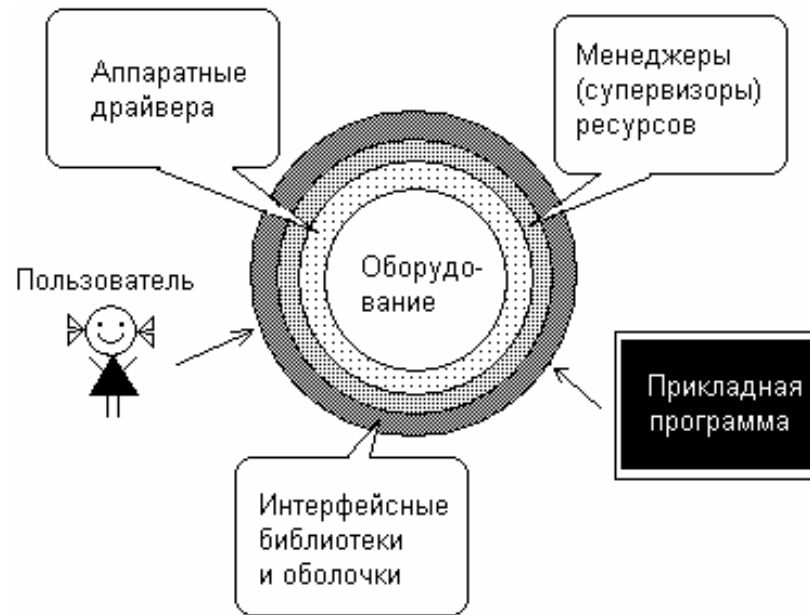
Серверы сетей и БД



Мобильные устройства



Структура вычислительной системы



Оборудование – процессор, память, интерфейсы ввода-вывода и пр.

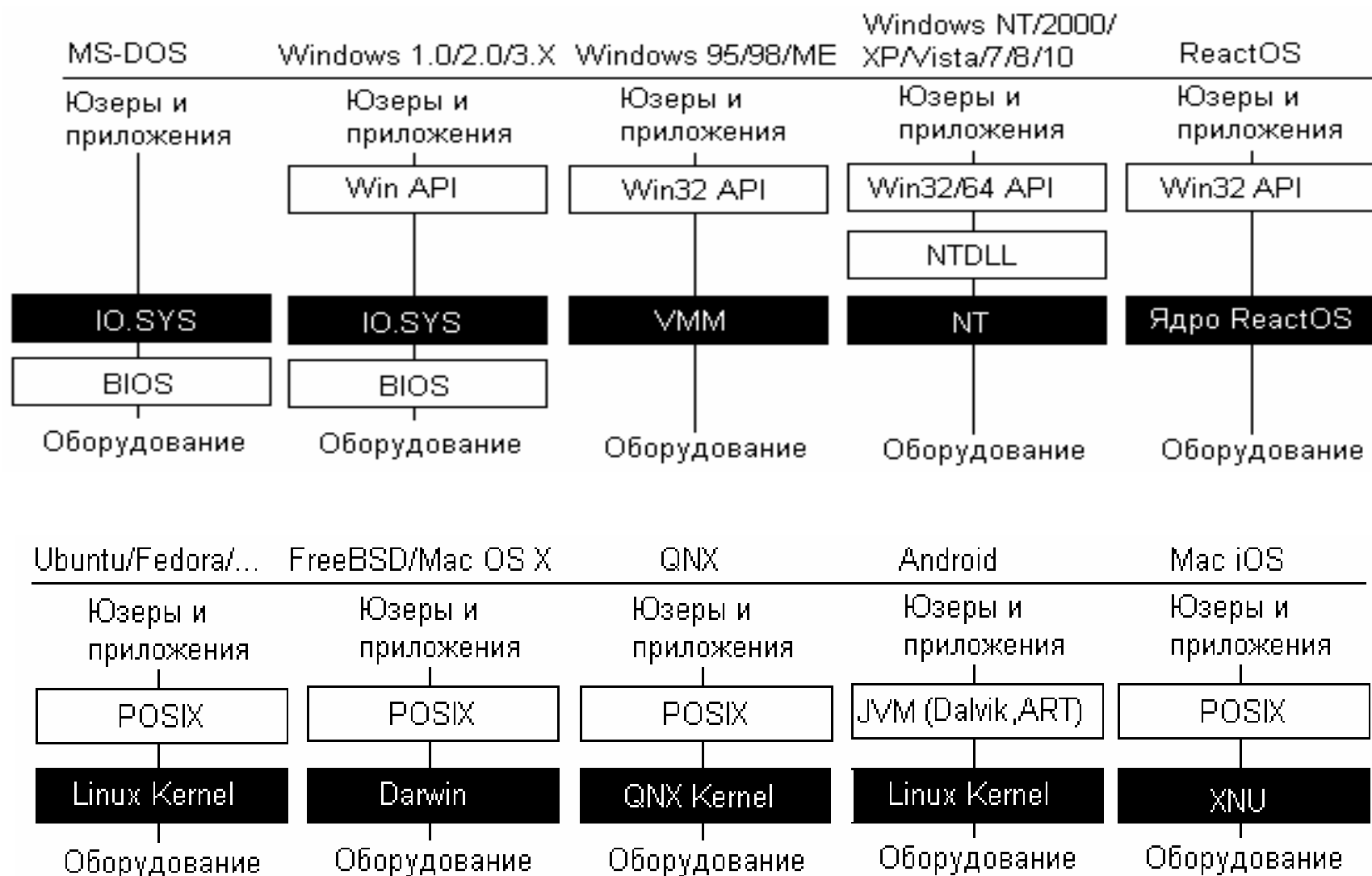
Менеджеры ресурсов – диспетчер задач, диспетчер памяти, диспетчер ввода-вывода, файловая система, сетевая система, подсистема защиты и пр.

Интерфейсные библиотеки (для программ) – внутрифирменный стандарт Microsoft Win API (KERNEL, USER, GDI), межфирменный стандарт POSIX (LibC, LibX), межфирменный стандарт JVM и пр.

Интерфейсные оболочки (для пользователя) - COMMAND.COM, CMD, KDE, Gnome, bash и пр.

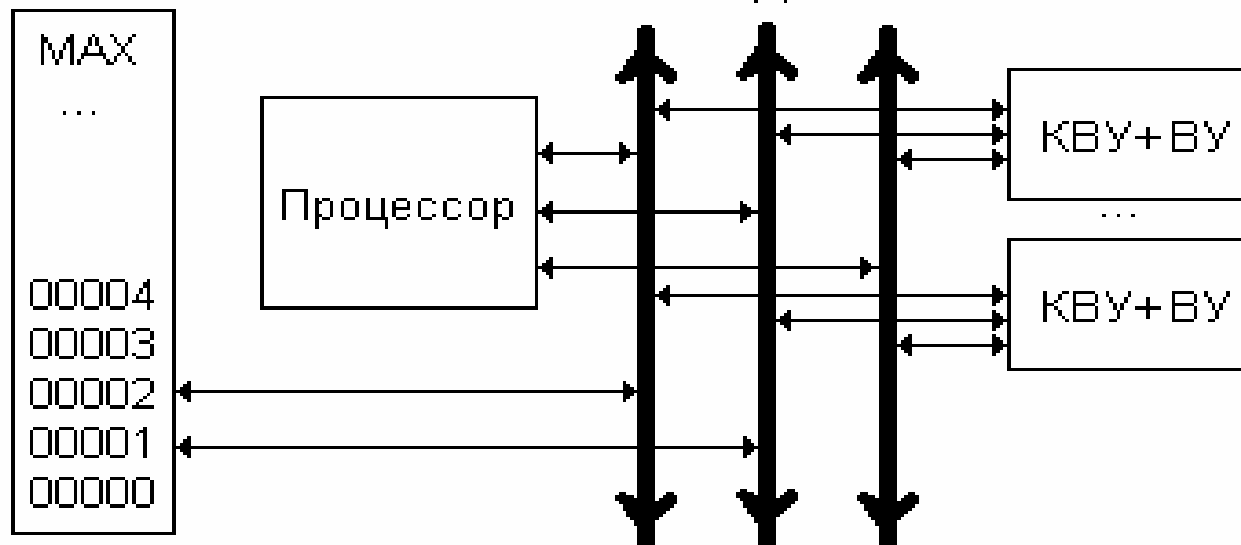
Прикладные программы – Проводник, Блокнот, MS Office, Midnight commander, GCC, Counter Strike, Пасьянс и пр.

Примеры архитектур ОС



Архитектура оборудования ЭВМ

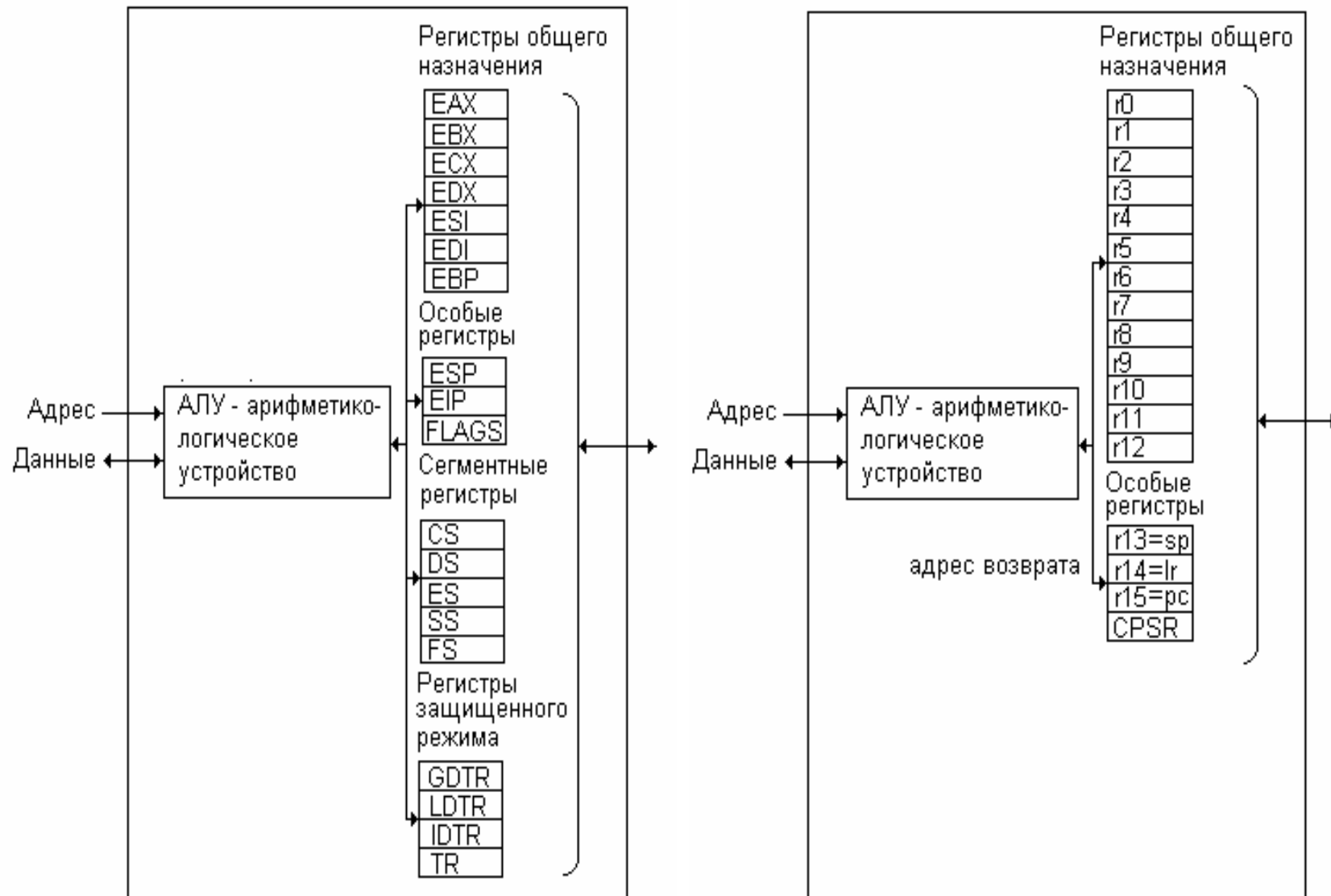
Адресное
простран-
ство



Процессор

Intel x86

ARM

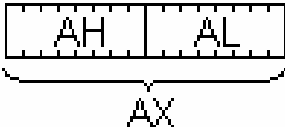


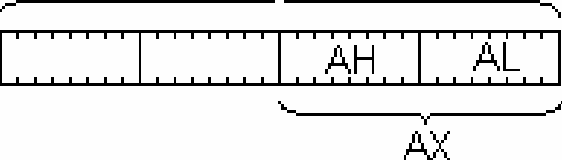
Особые регистры: 1) счетчик команд 2) регистр флагов; 3) указатель стека

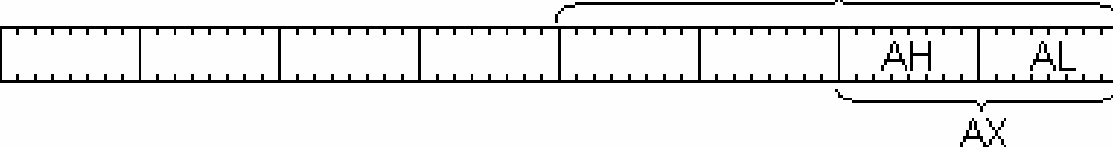
Обзор системы команд архитектуры INTEL

Структура регистров A,B,C и D:

8-разрядные процессоры A 

16-разрядные процессоры AX 

32-разрядные процессоры EAX 

64-разрядные процессоры RAX 

Примеры использования:

MOV AH, AL

MOV DL, DH

MOV BX, CX

~~MOV DX, EAX~~

MOV EDX, EBP

Обзор системы команд (продолжение)

Пример выполнения 3-местной операции сложения
при помощи 2-местной команды:

ADD [EDX+12345678h], EBP

Пусть EDX=87654321h, EBP=11111111h

99999998	22h
99999999	33h
9999999A	44h
9999999B	55h
9999999C	66h
9999999D	77h

Адрес источника/приемника: $EDX+12345678h = 87654321h+12345678h = 99999999h$

Суть операции:

$[99999999h] := [99999999h] + EBP = 66554433h + 11111111h = 77665544h$

То есть: ПРИЕМНИК:=ПРИЕМНИК+ИСТОЧНИК

Обзор системы команд (продолжение)

Количество операндов:

- Двухместные (двухоперандные) команды: **MOV Источник, Приемник**
- Одноместные (однооперандные команды): **NOT Операнд**
- Безоперандные команды: **NOP**

Способы адресации операндов:

- Регистровый: **NOT EAX**
- Непосредственный: **MOV EAX, 12345678h**
- Прямой: **MOV [12345678h], EAX** или **MOV МЕТКА, EAX**
- Косвенный: **MOV [EBX], 0**
- Косвенный со смещением: **MOV [EBX+12345678h], 0** или **MOV МЕТКА[EBX]**
- Базово-индексный: **MOV [EBX+ESI], 12345678h**
- Базово-индексный со смещением: **MOV [EBX+ESI+4], 12345678h** или **MOV МЕТКА[EBX+ESI], 12345678h**

Обзор системы команд (продолжение)

Как кодируются команды:

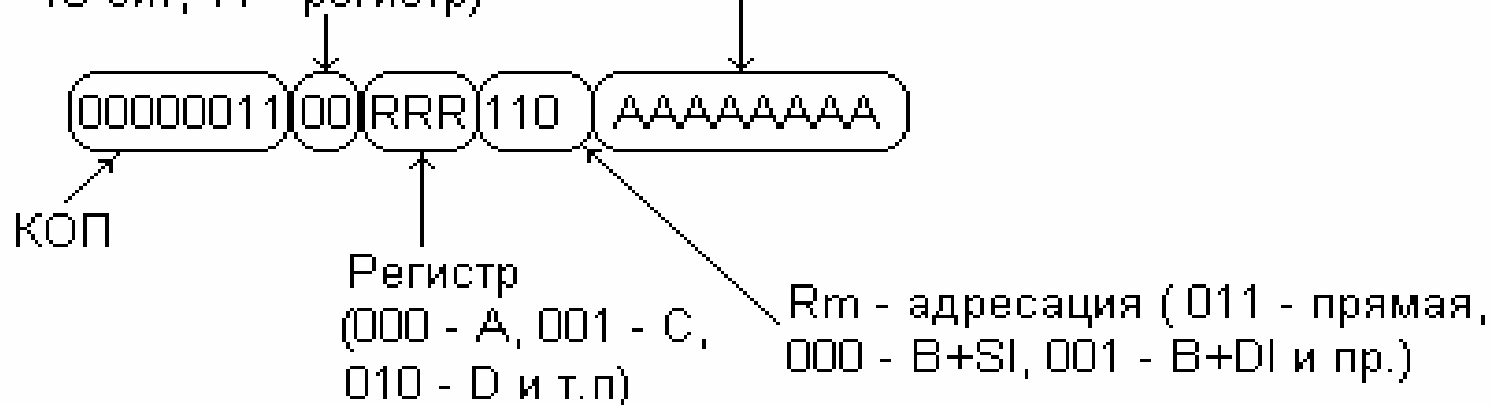
Префикс 0/1	КОП 1/2	Mod R/M 0/1	SIB 0/1	Смещение 0/1/2/4	Операнд 0/1/2/4
----------------	------------	----------------	------------	---------------------	--------------------

Пример: команда сложения регистра с памятью «**ADD R, [AAAA]**».

Mod - длина смещения

(00 - нет, 01 - 8 бит,
10 - 16 бит, 11 - регистр)

Адрес в памяти



```
0000 0011+00+000+101+11111111111111111111111111111111 =  
= 03 05 FF FF FF FF =  
= ADD EAX, FFFFFFFF
```

Обзор системы команд (продолжение)

Пример размещения команд и данных в памяти

```
0000016F: C706CE003412
00000175: C706D0007856
0000017B: A1CE00
0000017E: 0306D000
00000182: A3D200
```

```
mov [000CE], 01234
mov [000D0], 05678
mov ax, [000CE]
add ax, [000D0]
mov [000D2], ax
```

```
int a,b,c;
main() {
  a = 0x1234;
  b = 0x5678;
  c = a+b; }
```

```
0040110B: C705B8C2400034120000
00401115: C705BCC2400078560000
0040111F: A1B8C24000
00401124: 0305BCC24000
0040112A: A3C0C24000
```

```
mov [00040C2B8], 000001234
mov [00040C2BC], 000005678
mov eax, [00040C2B8]
add eax, [00040C2BC]
mov [00040C2C0], eax
```

```
int a,b,c;
main() {
  a = 0x1234;
  b = 0x5678;
  c = a+b; }
```

Обзор системы команд (продолжение)

1. Команды пересылки данных

- **MOV ПРИЕМНИК, ИСТОЧНИК** – копирует данные;
- **LEA ПРИЕМНИК, ОБЪЕКТ** – загружает адрес объекта;
- **XCHG ОПЕРАНД1,ОПЕРАНД2** – обмен операндами;
- **MOVSБ/MOVSW/MOVSD** – копирование из [SI/ESI] в [DI/EDI] с инкрементом регистров SI/ESI и DI/EDI;
- **LODSБ/LODSW/LODSD** – копирование из памяти по адресу [SI/ESI] в AL/AX/EAX с инкрементом регистра SI/ESI;
- **STOSБ/STOSW/STOSD** – копирование из AL/AX/EAX в память по адресу [DI/EDI].

Команды **MOV ПАМЯТЬ,ПАМЯТЬ** нет. Зато **MOVSD**:

```
MOV Куда-то, [ESI]  
ADD ESI, 4  
MOV [EDI], Откуда-то  
ADD EDI, 4
```

Бит D в регистре флагов !!!

Обзор системы команд (продолжение)

Пересылка операндов разной длины

Исходное
состояние

0005	00
0004	00
0003	00
0002	00
0001	00
0000	00

MOV [0001], EAX

0005	00
0004	FF
0003	FF
0002	FF
0001	FF
0000	00

MOV [0001], AX

0005	00
0004	00
0003	00
0002	FF
0001	FF
0000	00

MOV [0001], AL

0005	00
0004	00
0003	00
0002	00
0001	FF
0000	00

2. Целочисленные арифметико-логические команды

2.1. Представление данных

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

A 5

$$= 16^5 = -91$$

Обзор системы команд (продолжение)

2.2. Арифметические команды

- **ADD ПРИЕМНИК, ИСТОЧНИК** – сложение;
- **SUB ПРИЕМНИК, ИСТОЧНИК** – вычитание;
- **ADC ПРИЕМНИК, ИСТОЧНИК** – сложение с учетом бита C;
- **SBB ПРИЕМНИК, ИСТОЧНИК** – вычитание с учетом бита C;
- **MUL МНОЖИТЕЛЬ** – умножение AL/AX/EAX на МНОЖИТЕЛЬ, результат в AL:AH, AX:DX или EAX:EDX;
- **DIV ДЕЛИТЕЛЬ** – деление AL/AX/EAX на ОПЕРАНД, результат в AL:AH, AX:DX или EAX:EDX;
- **IMUL МНОЖИТЕЛЬ** и **IDIV ДЕЛИТЕЛЬ** – умножение или деление с учетом знака;
- **INC ОПЕРАНД** – инкремент операнда;
- **DEC ОПЕРАНД** – декремент операнда;
- **NEG ОПЕРАНД** – изменение знака операнда.

Некоторые биты регистра флагов:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				O				S	Z				P		C

C – перенос, P – четность, Z – ноль, S – знак, O – переполнение

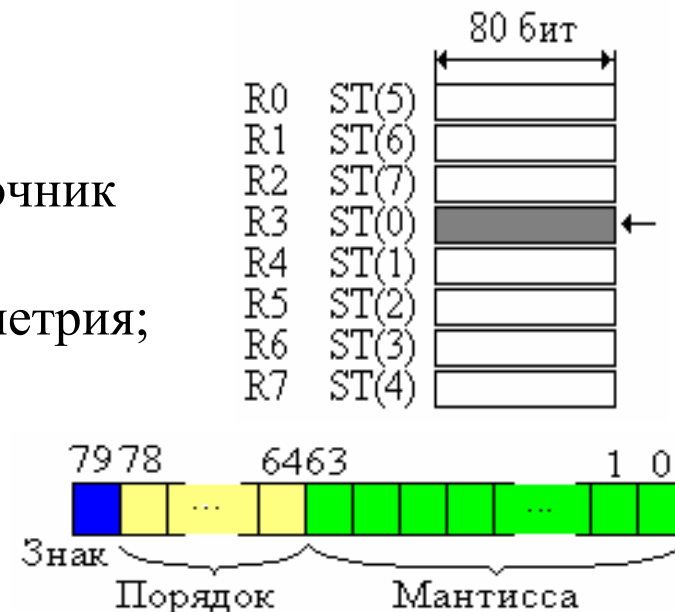
Обзор системы команд (продолжение)

2.4. Логические команды

- **AND** приемник,источник — логическое умножение операндов;
- **OR** приемник,источник — логическое сложение операндов;
- **XOR** приемник,источник — «исключающее ИЛИ» (!!!);
- **NOT** операнд — инвертирование битов;
- **SHR/SHL** операнд,количество — логический сдвиг вправо/влево;
- **ROR/ROR** операнд,количество — циклический сдвиг вправо/влево.

2.5. Команды вещественной арифметики (над 7 типами данных)

- **FLD** — загрузить число в стек;
- **FST** — скопировать число из стека;
- **FSTP** — прочесть число из стека;
- **FADD/FSUB/FMUL/FDIV** приемник,источник
— арифметика;
- **FSIN/FCOS/FPTAN/FPATAN** — тригонометрия;
- **F2XMI/FYL2X/FYL2XP1** — экспоненты и
логарифмы.



Обзор системы команд (продолжение)

2.6. Команды передачи управления

- **JMP** смещение или **JMP** адрес – безусловная передача;
- **J**** смещение - передача управления по условию.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				O	D			S	Z				P		C

- **JE** или **JZ** – переход о равенству;
- **JNE** или **JNZ** – переход по неравенству;
- **JA/JB** – переход по больше/меньше;
- **JAЕ/JBE** – переход по больше и равно/меньше и равно;
- **JCXZ** – переход, если CX/ECX=0.

2.7. Команды проверки условия (без изменения операндов)

- **CMР** операнд1,операнд2 – сравнение вычитанием;
- **TEST** операнд1,операнд2 – сравнение логическим умножением.

```
MOV EBP, 123
NEXT: . . .
DEC EBP
CMP EBP, 0 - не обязательно
JNE NEXT
```

```
CLD
MOV ECX, 123
NEXT: . . .
. . .
LOOP NEXT
```

Обзор системы команд (продолжение)

2.8. Команды работы со стеком

- **PUSH операнд/POP операнд** – втолкнуть/вытолкнуть из стека (**SP/ESP!!!**);
- **PUSHA/POPA** – втолкнуть/вытолкнуть весь набор регистров;
- **PUSHF/POPF** – сохранить/восстановить из стека регистр флагов.

2.9. Организация вызова процедур (подпрограмм)

CALL Адрес_перехода

=

PUSH Адрес_возврата
JMP Адрес

RET

=

POP Адрес_возврата
JMP Адрес_возврата

RET NNN

=

POP Адрес_возврата
JMP Адрес_возврата
ADD ESP, NNN

PPP: NOP
NOP
RET 8

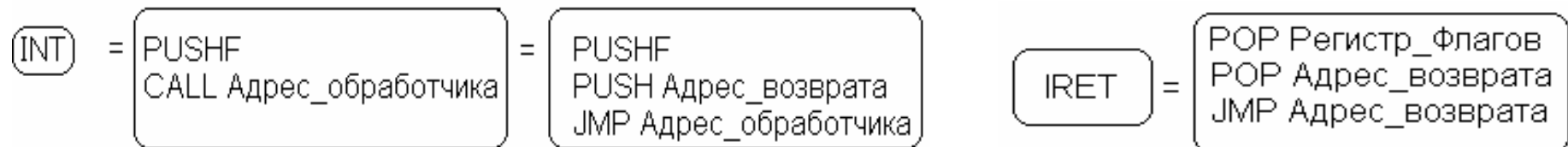
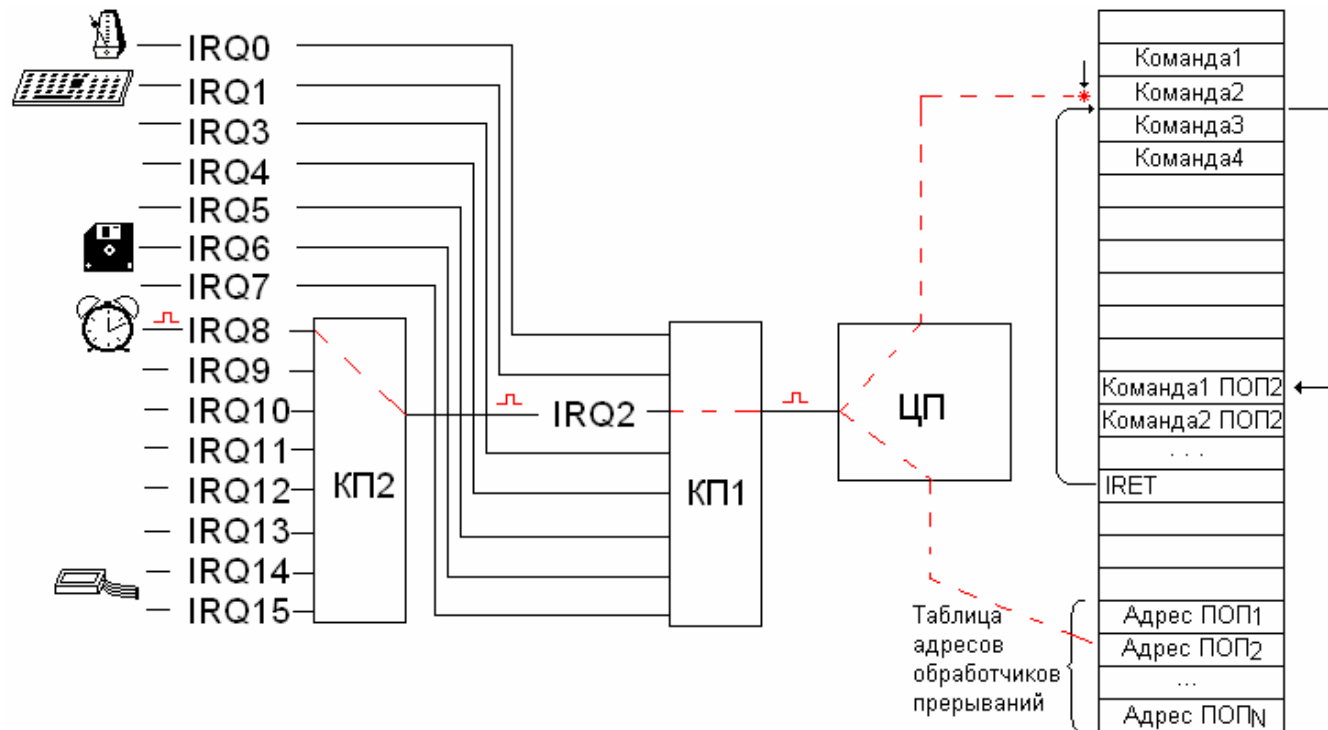
PUSH PARAM1
PUSH PARAM2
CALL PPP

; Стек пуст
; В стеке 4 байта
; В стеке 8 байтов
; В стеке 8 байтов + Адрес_возврата
; Стек опять пуст

Обзор системы команд (продолжение)

2.10. Организация системы прерываний и исключений

Прерывания обрабатываются **ПОСЛЕ** выполнения текущей команды, а исключения **В ПРОЦЕССЕ** выполнения.



Обзор системы команд (продолжение)

Некоторые исключения:

- 0 – деление на 0;
- 1 – вызывается после выполнения каждой команды;
- 3 – вызывается командой INT3 с кодом CC;
- 6 – неверная команда (например, вызывается командой UD2);
- 8 – двойная ошибка (возникла в обработчике другого исключения);
- D – общая ошибка защиты (например, **MOV [00000000], EAX**);
- E – обращение по физически отсутствующему адресу.

Некоторые логические номера прерываний:

	В реальном режиме	В Windows 9X	В Windows NT
IRQ 0	8	50	30
IRQ1	9	51	31
IRQ3	B	53	33
IRQ4	C	54	34
IRQ5	D	55	35
IRQ6	E	56	36
IRQ7	F	57	37
IRQ8	70	58	38
IRQ9	71	59	39
IRQ10	72	5A	3A
IRQ11	73	5B	3B
IRQ12	74	5C	3C
IRQ13	75	5D	3D
IRQ14	76	5E	3E
IRQ15	77	5F	3F

Обзор системы команд (продолжение)

Обработка исключений

А) в режиме супервизора

```
atikmdag.sys
PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

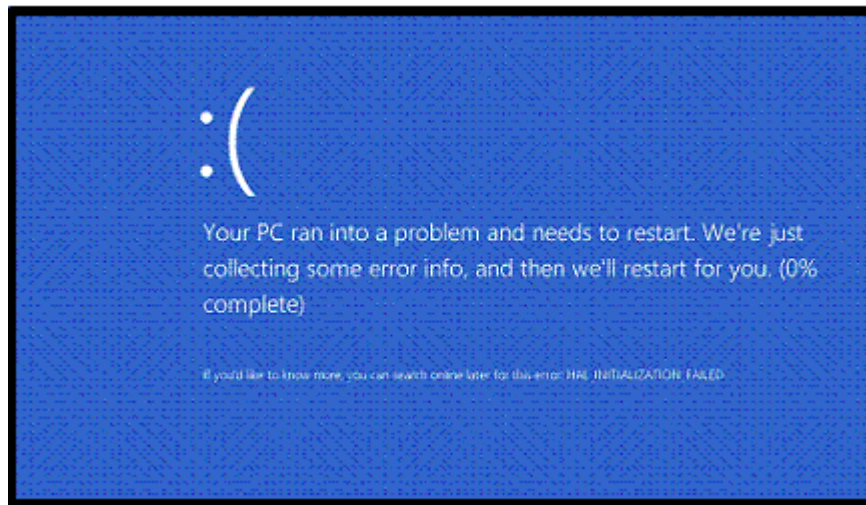
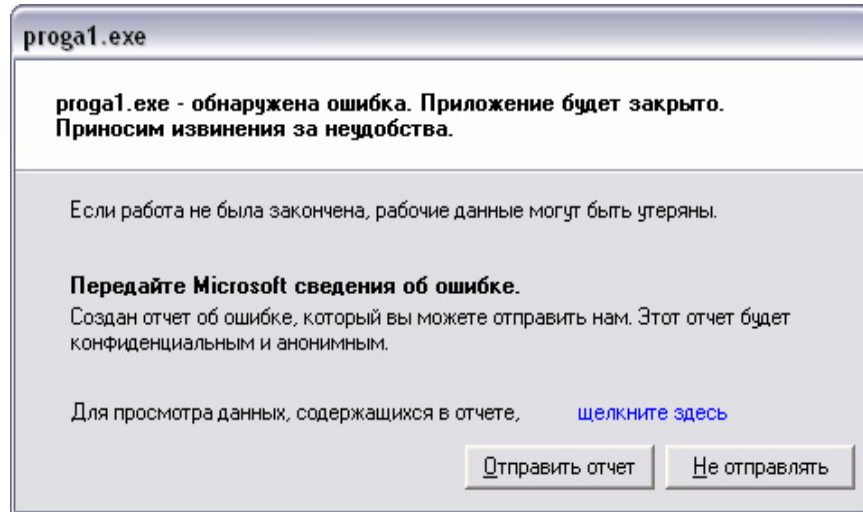
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:
*** STOP: 0x00000050 (0xFFFFF8A00309D000, 0x0000000000000000, 0xFFFFF8B004C4D92F, 0
x0000000000000000)

*** atikmdag.sys - Address FFFFF8B004C4D92F base at FFFFF8B004896000, DateStamp
4f7e4b69

Collecting data for crash dump...
Initializing disk for crash dump...
Beginning dump of physical memory...
Dumping physical memory to disk: 100
Physical memory dump complete.
Contact your system admin or technical support group for further assistance.
```

б) В пользовательском режиме



SEH – структурная обработка исключений



Обзор системы команд (окончание)

2.11. Команды доступа ко внешним устройствам

- **IN AL,Номер_порта** – ввод в регистр из порта;
- **OUT Номер_порта,AL** – вывод в порт из регистра.

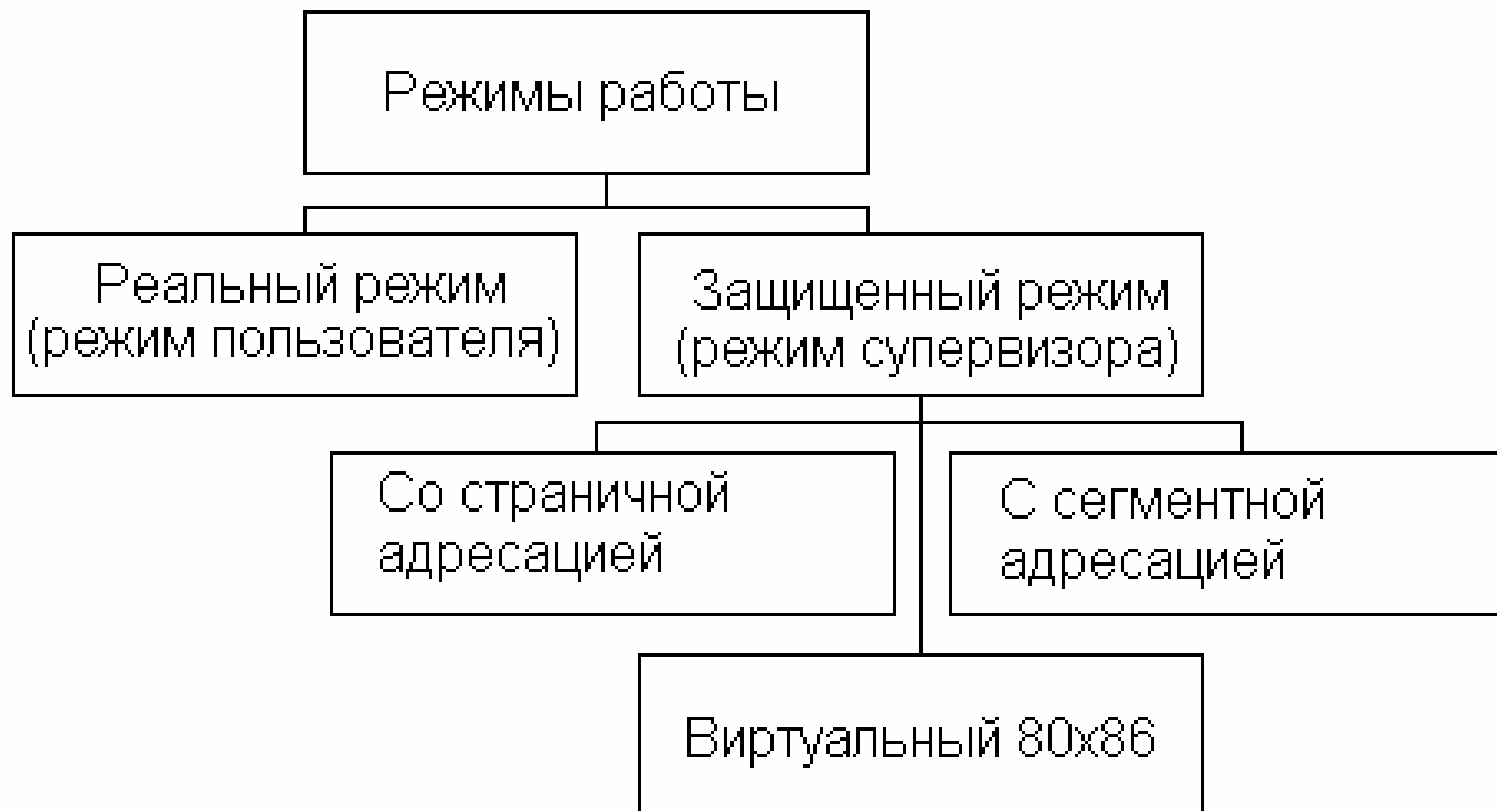
2.12. Прочие команды

- **NOP** – пустая команда;
- **UD2** – генерация исключения «неопределенная операция»;
- **INT номер** – генерация прерывания (!!!);
- **CPUID** – сброс кэшей и возврат идентифицирующей информации;
- **RDTSC** – возврат в EDX:EAX счетчика тактов процессора.

2.13. Расширенные наборы команд

- **MMX** – набор команд для массовых операция над целыми векторами;
- **3DNow!** – расширение набора MMX в процессорах AMD;
- **SSE, SSE2, SSE3, SSE4** – операции над вещественными векторами.

Режимы работы процессоров INTEL



Реальный режим

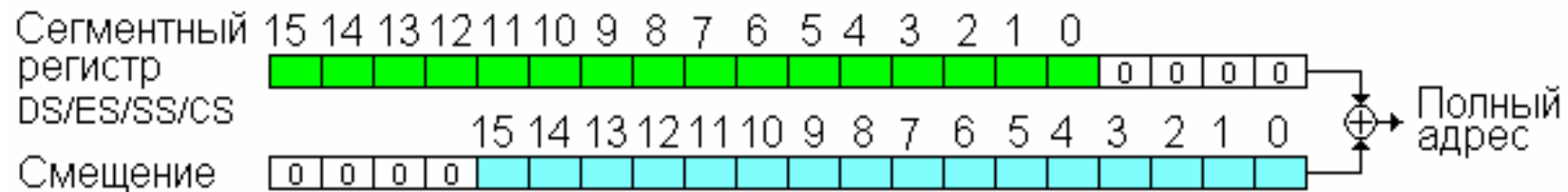
3.1. Реальный режим

- Шина данных и регистры 16-битовые (диапазон 0..65535);
- Шина адреса 20-битовая (объем памяти 1 Мб);
- Шина ввода-вывода 8-битовая (всего 256 внешних устройств)

MOV AX, [1234] ~ MOV AX, DS:[12324]
MOV AX, [SI] ~ MOV AX, DS:[SI]
MOV AX, [SP] ~ MOV AX, SS:[SP]

Логический адрес := Сегмент : Смещение

Физический адрес := Сегмент * 16 + Смещение



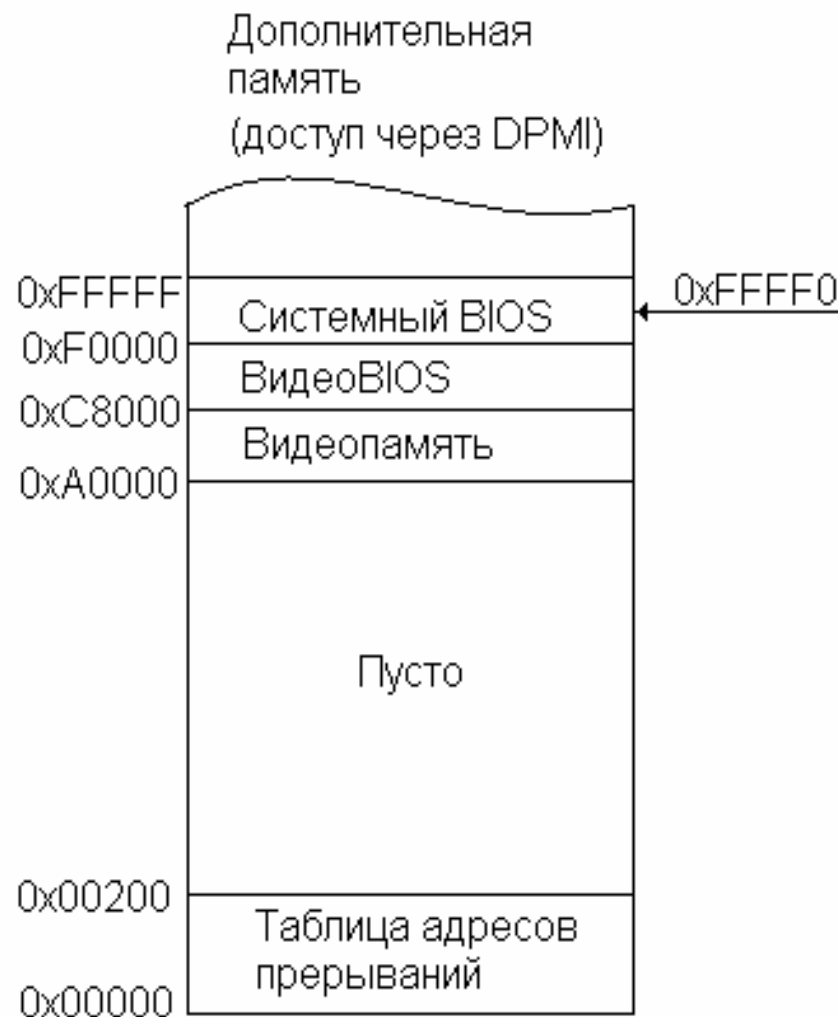
Распределение памяти в реальном режиме

Состав BIOS:

1. Программа POST
2. Программа SETUP
3. Библиотеки доступа к оборудованию
 - 3.1. К видео через INT 10h
 - 3.2. К клавиатуре через INT 16h
 - 3.3. К диску через INT 13h

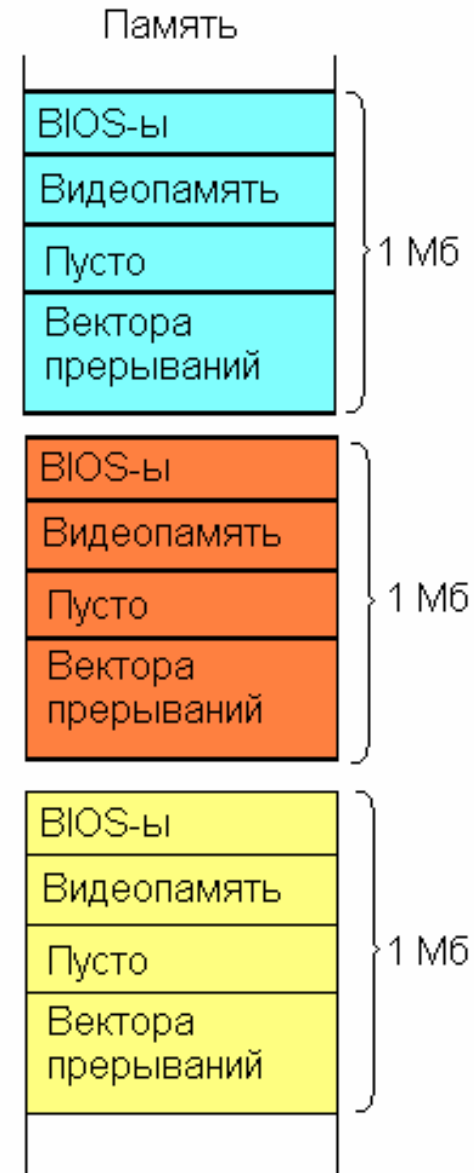
Примеры:

```
Mov ah, 0Eh    ; Код функции
Mov bh, 0      ; Номер видеостраницы
Mov al, 'A'    ; Код символа
Int 10h        ; Вызов BIOS
```



Режим виртуального 80x86

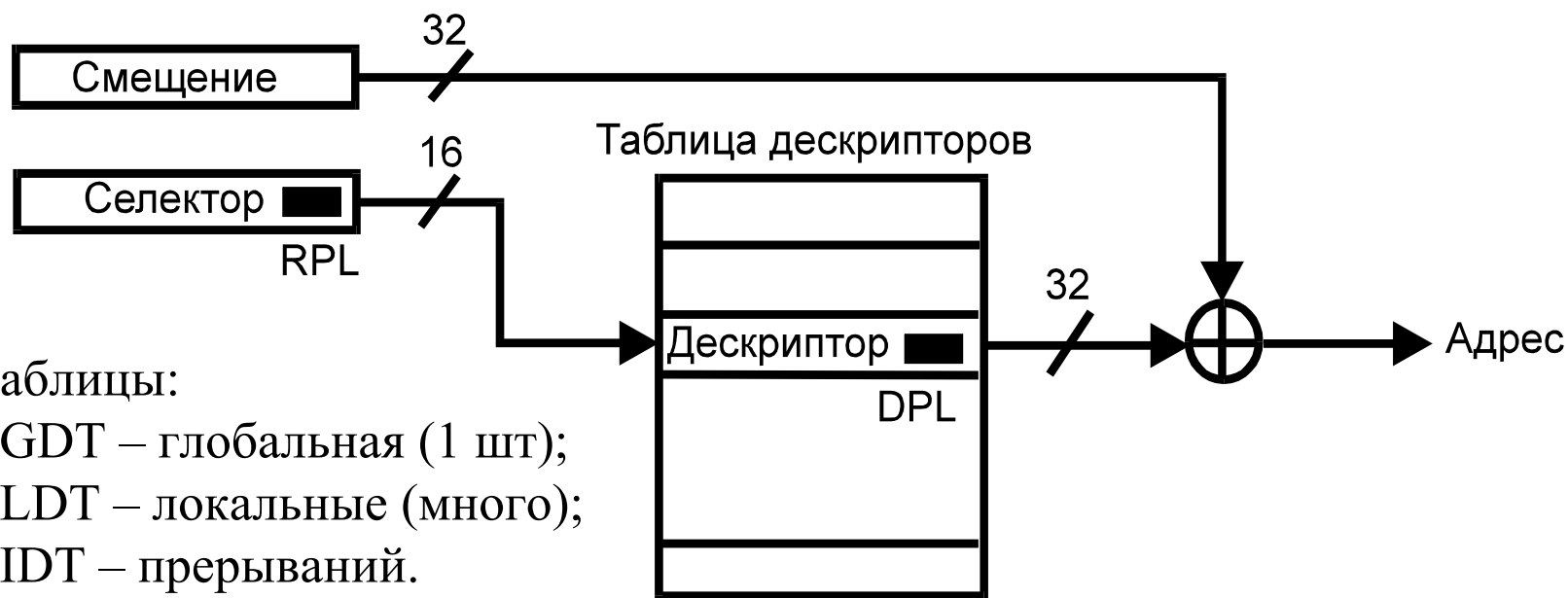
3.2. Является частью 32-битового защищенного режима. В 64-битовом не поддерживается.



Защищенный режим. Сегментная адресация

3.3. Защищенные режимы (биты 0 и 31 в CR0):

- Шина адреса 32 (макс. адр. = 4Гб ~ 4.2 млрд байтов) или 48 битов;
- Шина данных 32 или 64 бита;
- Шина ввода-вывода 16 (?) битов;
- Доп. регистры: *DTR, CR0-CR4 (упр. процессором), DR0-DR7 (отладка).

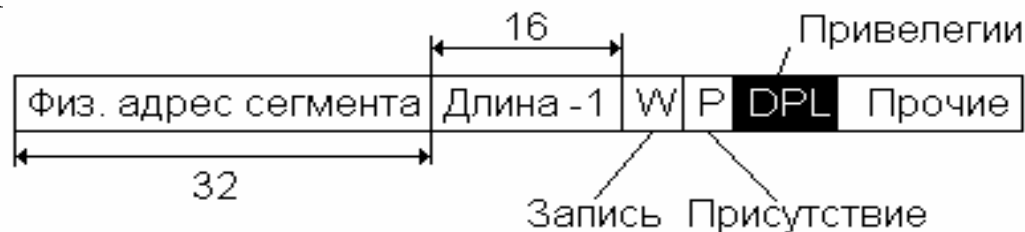


Таблицы:

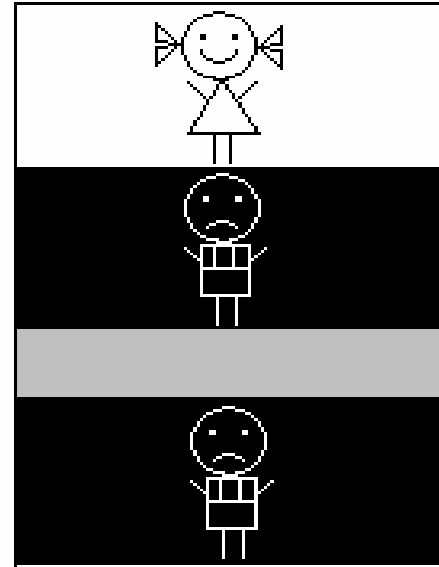
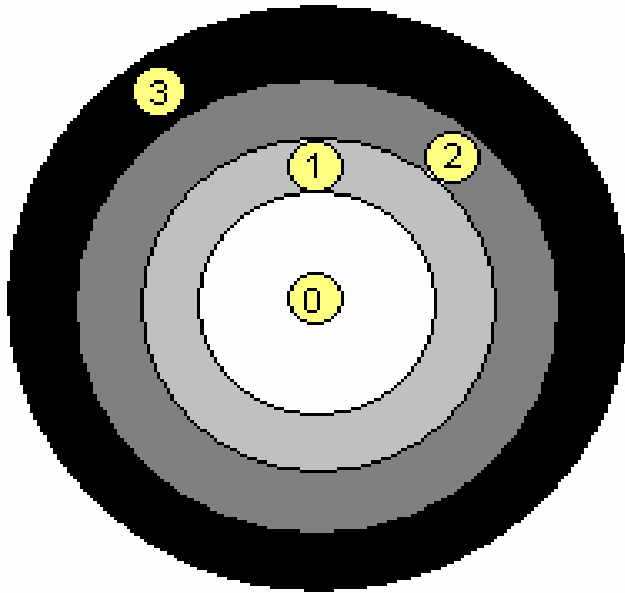
- GDT – глобальная (1 шт);
- LDT – локальные (много);
- IDT – прерываний.

Регистры GDTR, LDTR и IDTR.

Формат дескриптора:



Защищенный режим. Привелегии сегментов



CPL – уровень привилегий текущей программы;

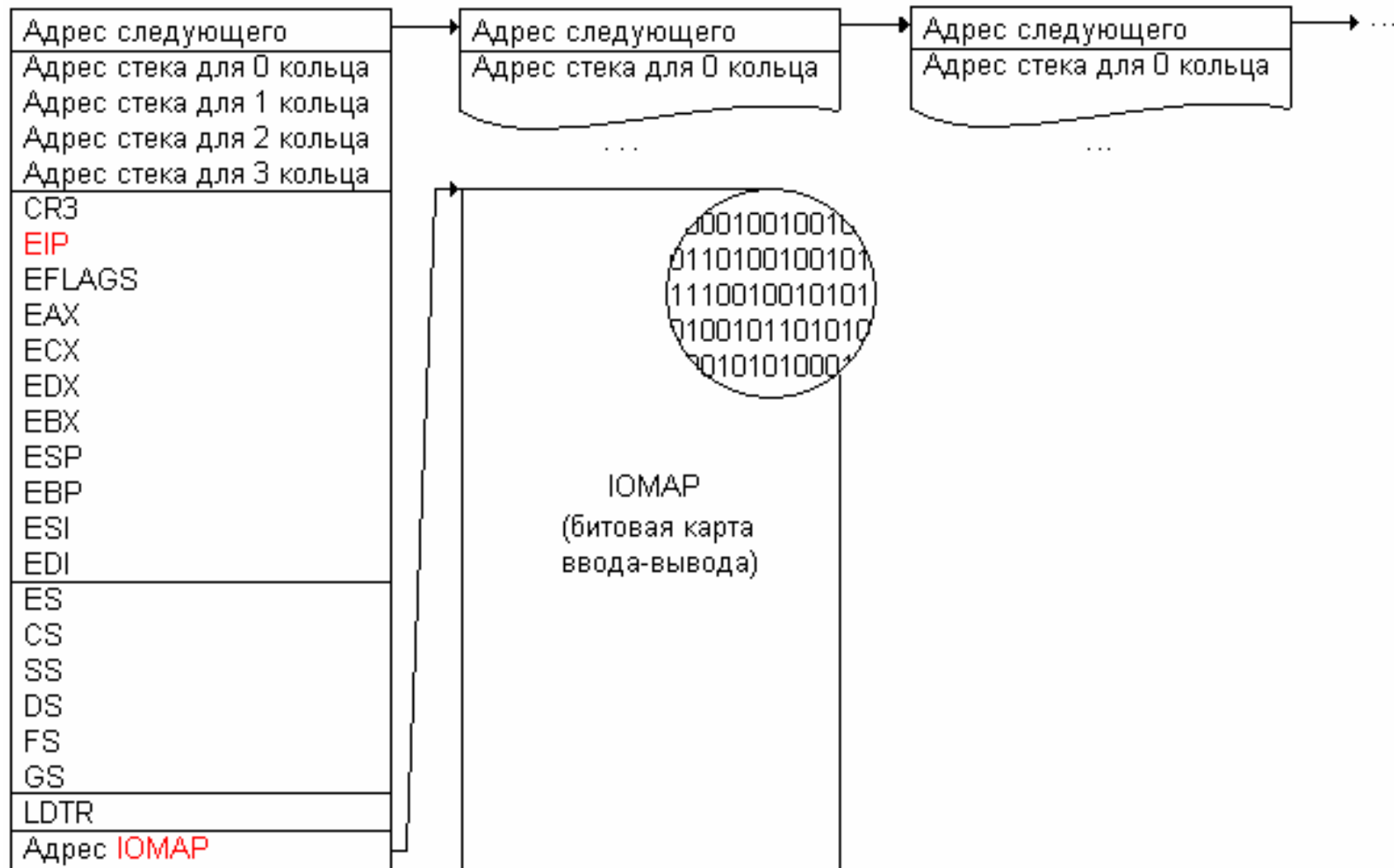
RPL – запрашиваемый уровень привилегий (в селекторе);

DPL – реальный уровень привилегий (в дескрипторе).

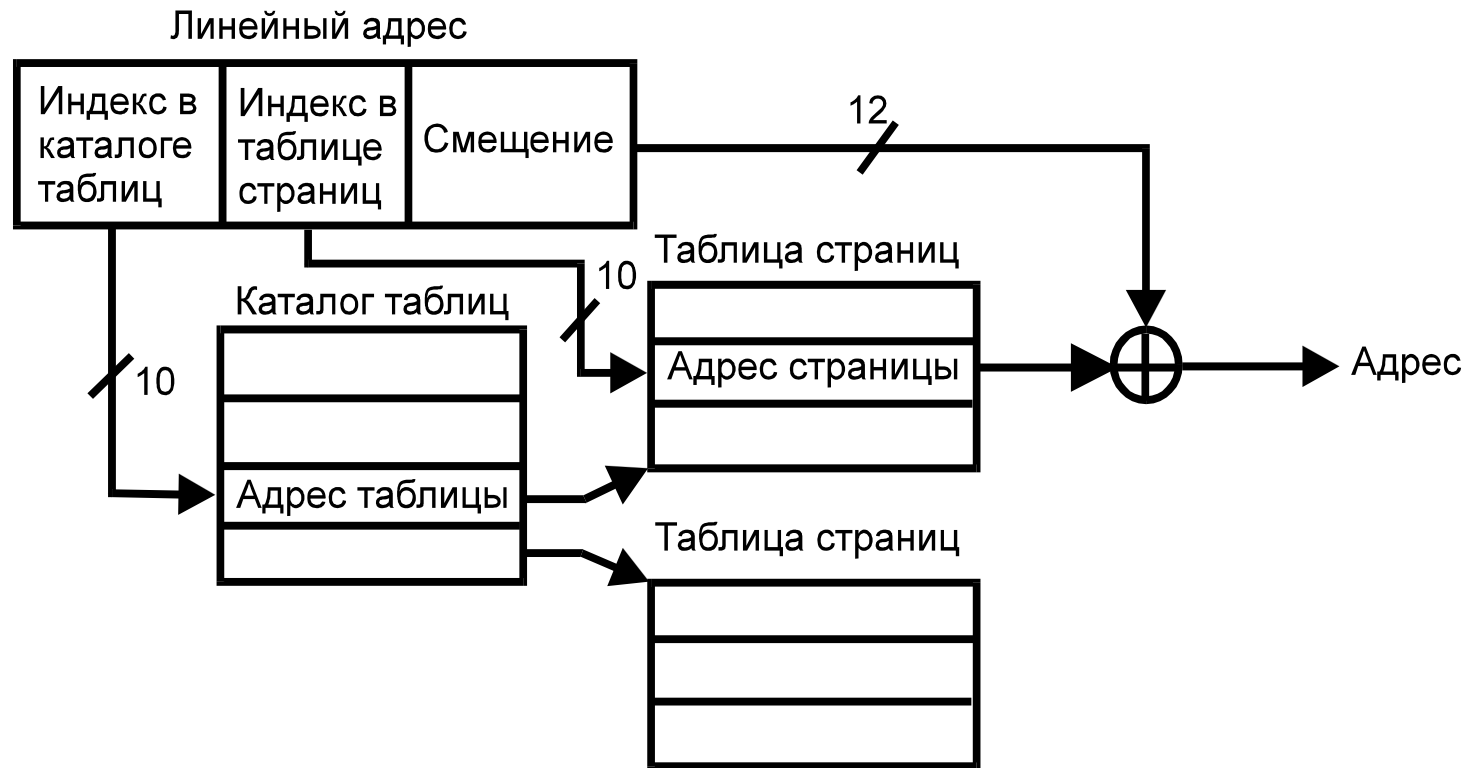
Условие доступа: $CPL \leq RPL \leq DPL$. (**Потому что $0 > 1 > 2 > 3$ ☺**)

Привилегированные команды: IN/OUT, LGDTR/LLDTR/LIDTR и пр.

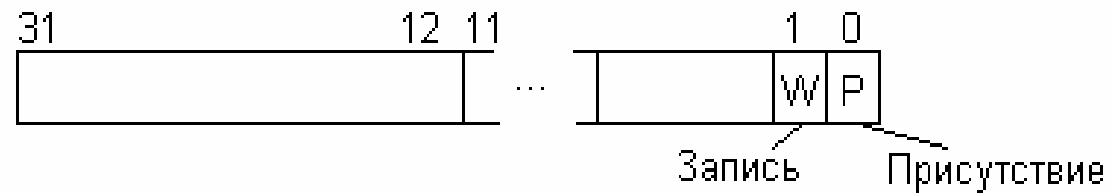
Защищенный режим. TSS – сегмент состояния задачи



Защищенный режим. Страничная адресация

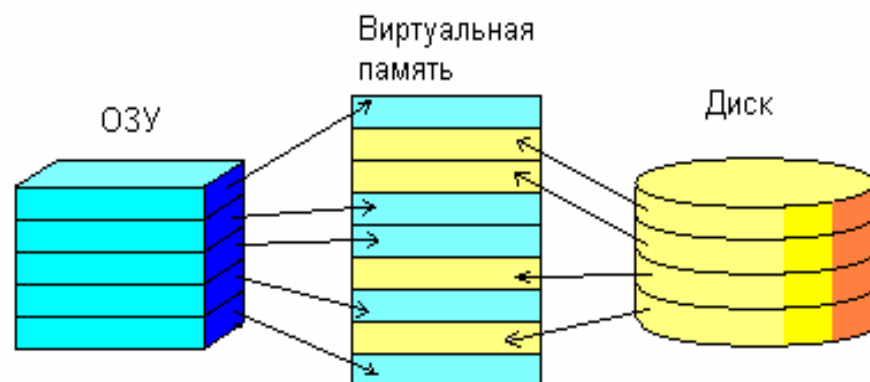


Формат строки каталога или таблицы:

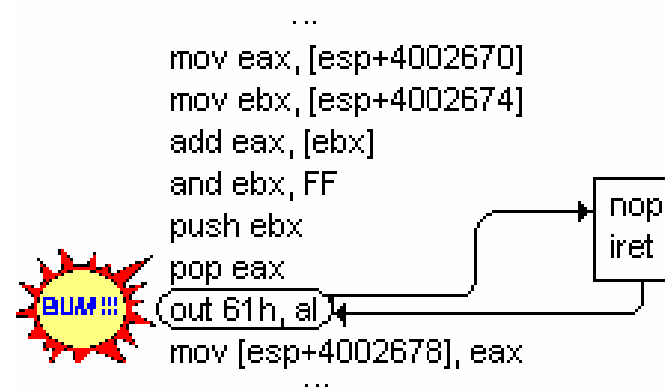


Защищенный режим. Виртуализация ресурсов

Виртуализация памяти



Виртуализация устройств



Операционная система = виртуальная машина

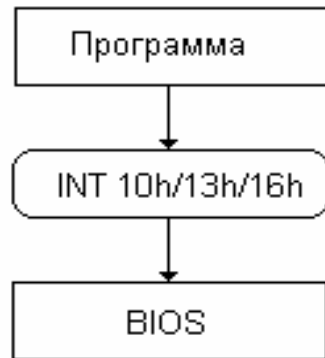
Особенности процессоров ARM

- Разрядность 32 или 64 бита;
- 16 регистров общего назначения;
- 7 режимов работы: для прикладных программ; для операционных систем; для драйверов; для обработки прерываний и исключений в разных режимах (4 шт);
- Трехместные команды: `ADD R0, R1, R2`;
- Условные команды: `MOVGT R0, R1`;
- Режим «сокращенных» команд;
- Режимы 16, 32 и 64 бита;
- Страница ввода-вывода.



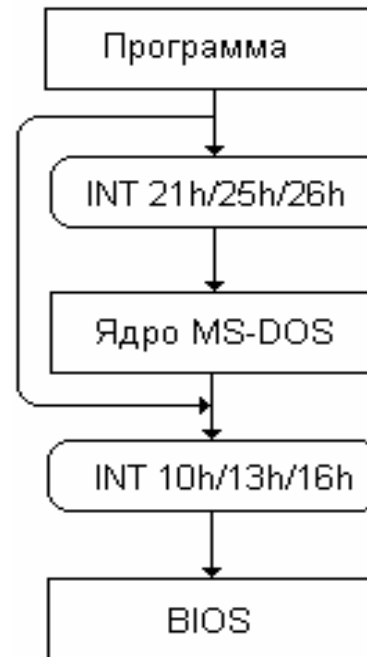
Обращение к ядру операционной системы

А) Загрузчик ОС



```
Mov ah, 0Eh    ; Код функции  
Mov bh, 0      ; Номер видеостраницы  
Mov al, 'A'    ; Код символа  
Int 10h        ; Вызов BIOS
```

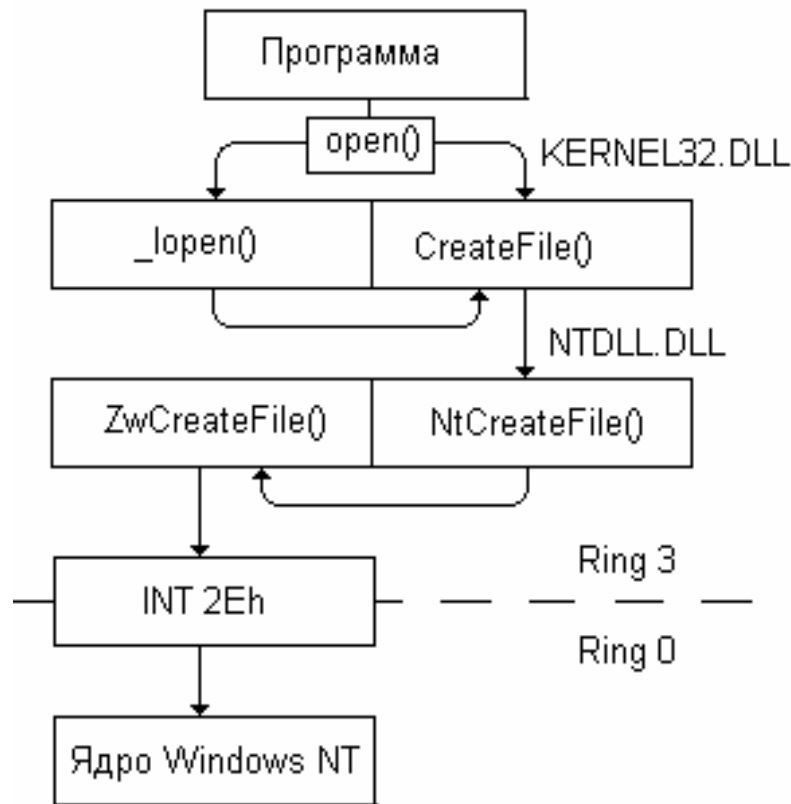
Б) MS-DOS



```
Fname db 'C:\FILE.TXT',0  
.  
.  
.  
mov ah, 3Dh    ; Код функции  
mov al, 2      ; Чтение + запись  
lea dx, Fname  ; Адрес имени  
int 21h        ; Обращение к ядру
```

Обращение к ядру операционной системы

В) Windows NT



KERNEL – память, процессы, файлы; **USER** – мышь, клавиатура, окна; **GDI** – графика; **ADVAPI** – криптография, Реестр.

```
; Через Win API
HFILE f = _lopen( "C:\\\\FILE.TXT", OF_READ|OF_WRITE );
_lwrite (f, Buf, 1 );
_lclose( f );
```

```
; Через Win32 API
HANDLE h = CreateFile("C:\\\\FILE.TXT",
    GENERIC_READ|GENERIC_WRITE,
    FILE_SHARE_WRITE, NULL,
    CREATE_ALWAYS,
    FILE_ATTRIBUTE_NORMAL,
    NULL);
WriteFile(h, Buf, 1, &HowMany, NULL);
CloseHandle(h);
```

; Через Win32 API на языке ассемблера

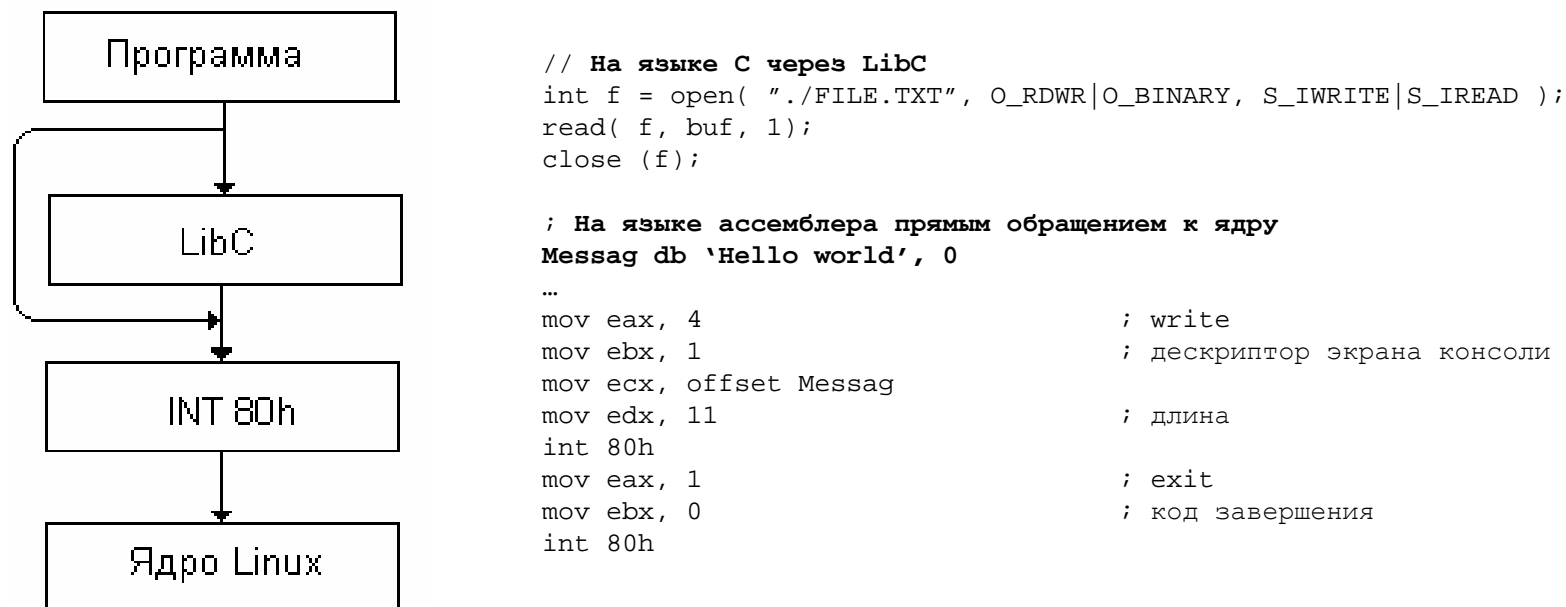
```
FileName db 'C:\\FILE.TXT',0
Buf      db ?
. . .
push 0
push 00000080h
push 2
push 0
push 1
push 40000000h
push offset FileName
call CreateFileA
mov EBP,EAX

push 0
push offset HowMany
push 1 ; Писать 1 байт
push offset Buf
push EBP
call WriteFile

push EBP
call CloseHandle
```

Обращение к ядру операционной системы

Г) Linux



LibC – память, процессы, файлы; **LibX** – графика.

Ядро ОС. Диспетчер памяти

Функции диспетчера памяти:

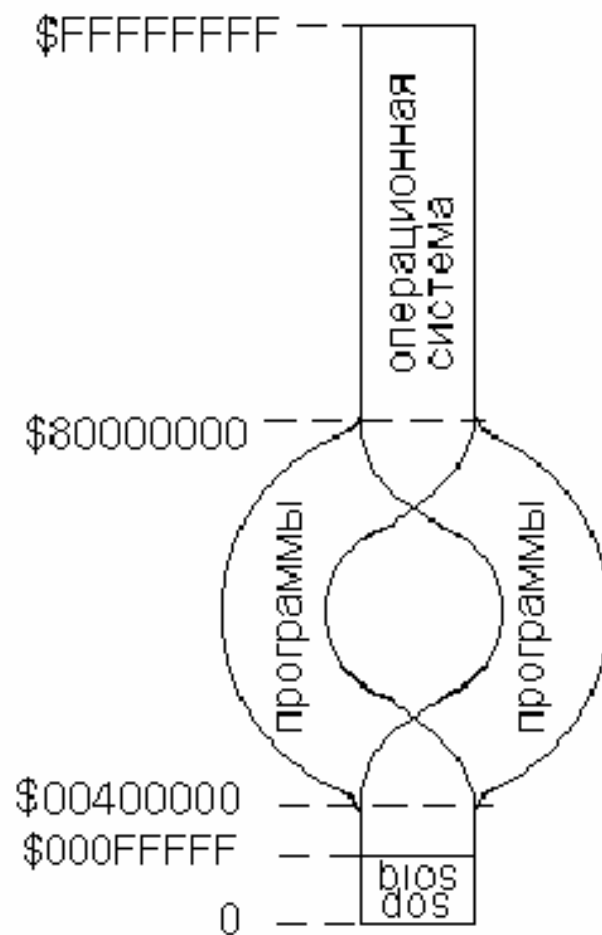
- Отслеживание, выделение, освобождение и дефрагментация памяти;
- Защита памяти
- Свопинг фрагментов виртуальной памяти
- Загрузка и выгрузка оверлеев
- Настройка символьных адресов на логические (или физические).

Отслеживание, выделение, освобождение и дефрагментация памяти

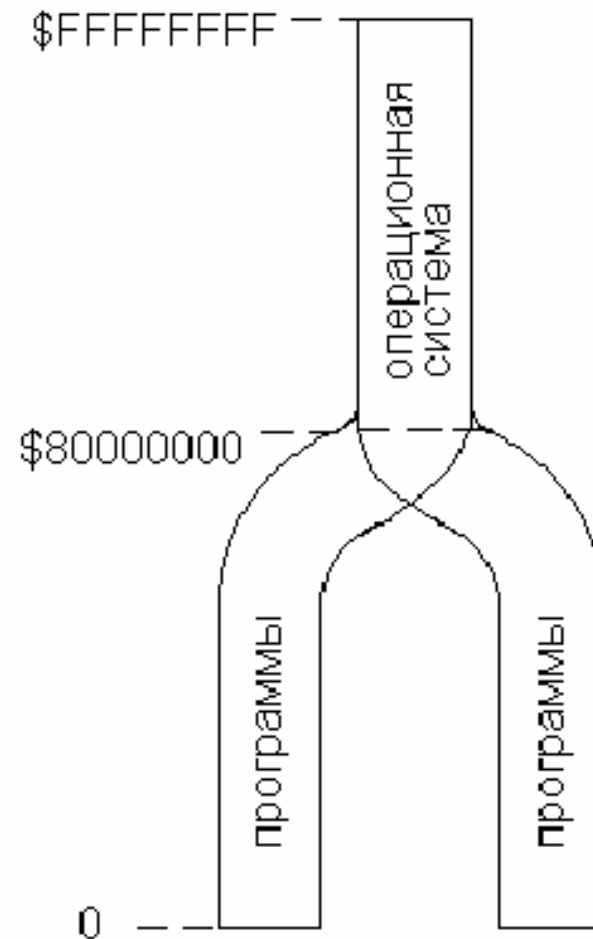
Методы организации адресного пространства:

- С постоянными разделами;
- С динамически выделяемыми разделами;
 - выделение первого подходящего;
 - выделение наиболее подходящего;
 - выделение наименее подходящего.
- С перемещаемыми разделами.

Ядро ОС. Диспетчер памяти (продолжение)



a) Windows 95/98/ME



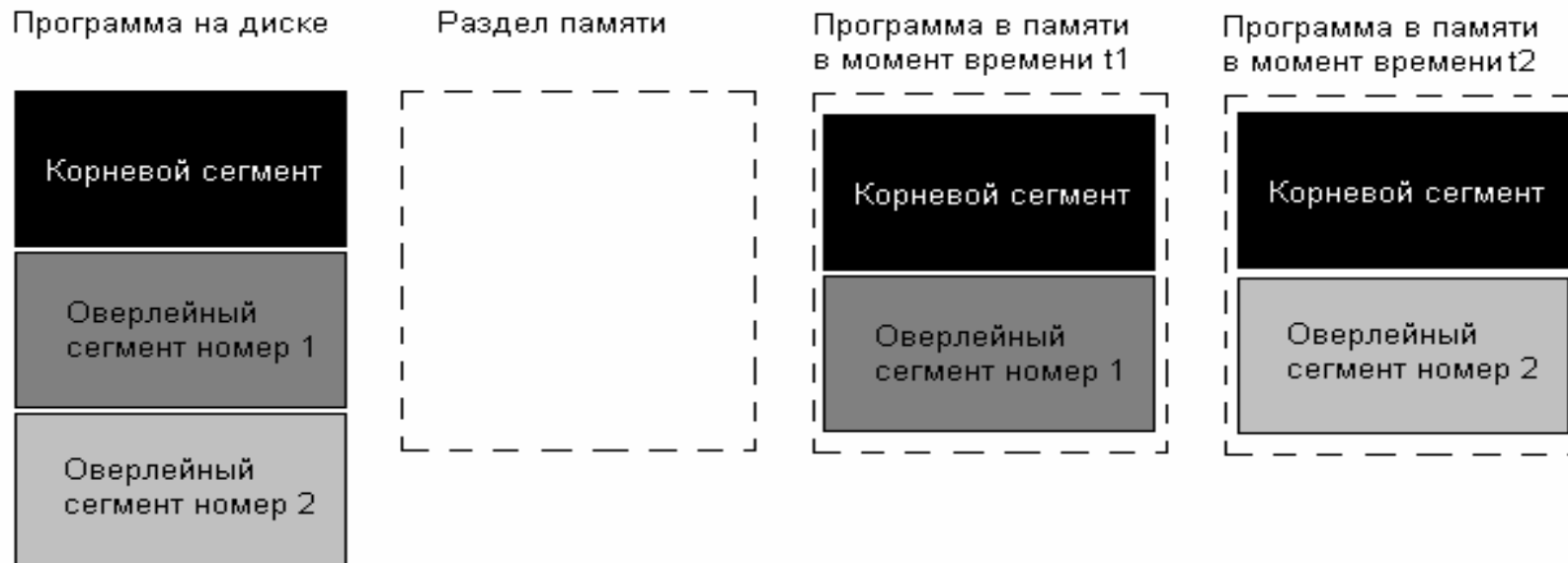
б) Windows NT/2000/XP/2003

Ядро ОС. Диспетчер памяти (продолжение)

Защита памяти

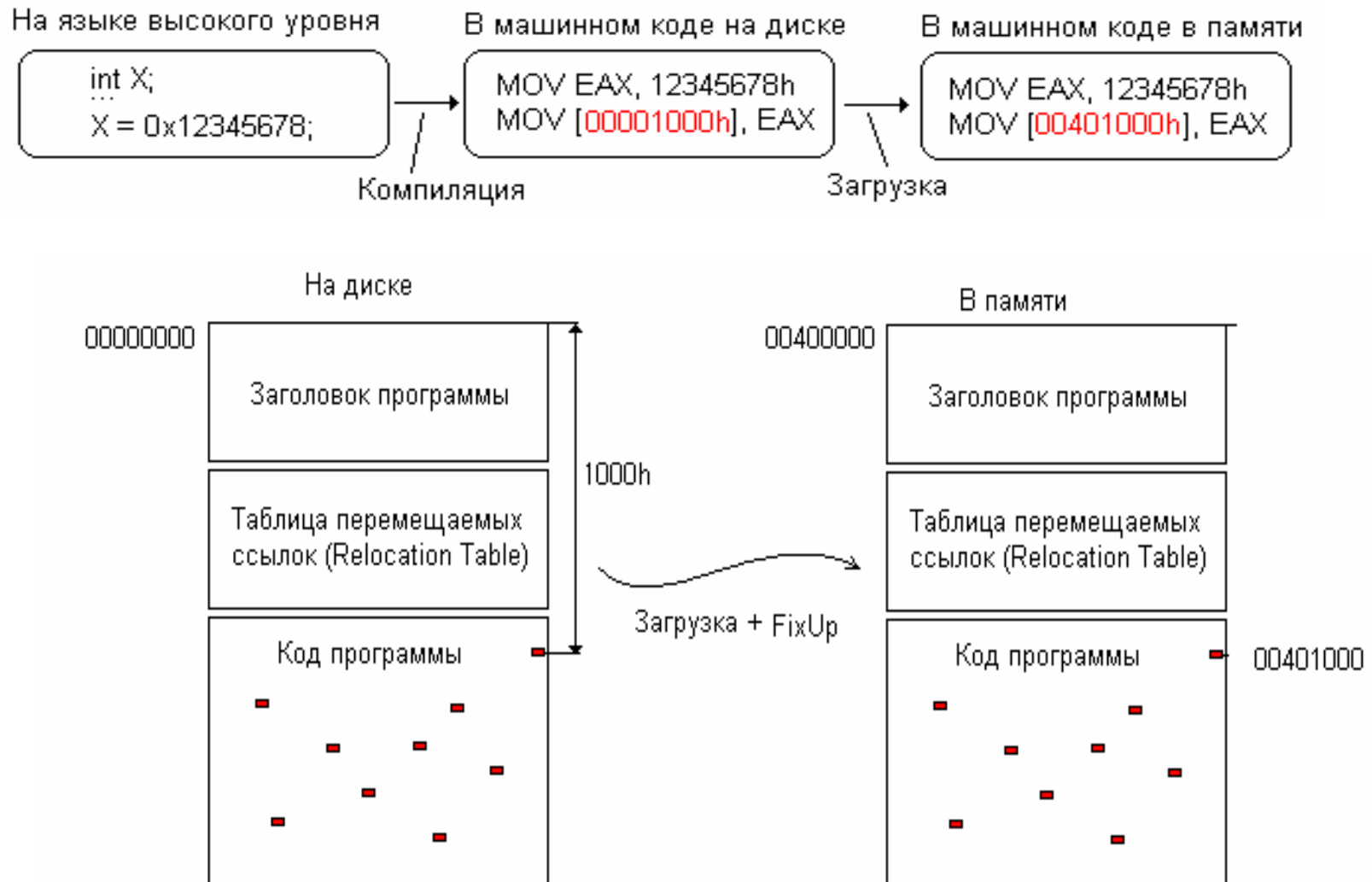
- Создание параллельных адресных пространств
- «Раскрашивание» фрагментов памяти:
 - разрешение/запрет любого доступа
 - разрешение/запрет записи
 - разрешение/запрет выполнения программного кода
 - разрешение/запрет свопинга
 - разрешение/запрет совместного доступа.

Загрузка-выгрузка оверлеев



Ядро ОС. Диспетчер памяти (окончание)

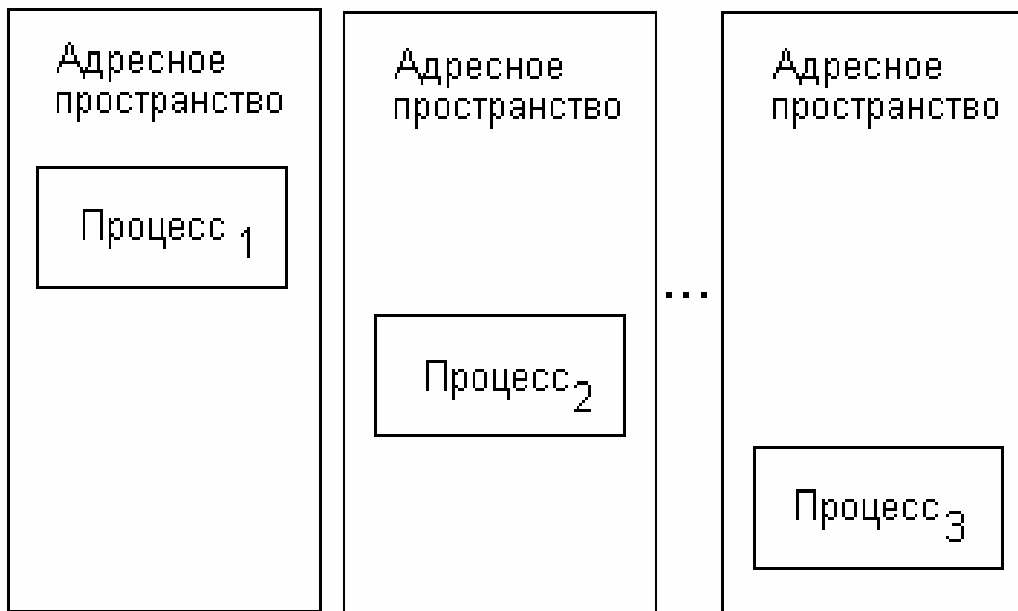
Настройка символьных адресов



Ядро ОС. Диспетчер задач

Задача = программа, предназначенная для одновременной работы с другими программами (задачи)

А) Процессы



б) Потоки (нити)



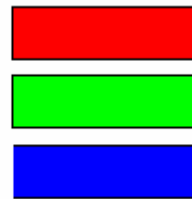
Ядро ОС. Диспетчер задач (продолжение)

Организация выполнения нескольких задач

А) Последовательная
(пакетный режим)



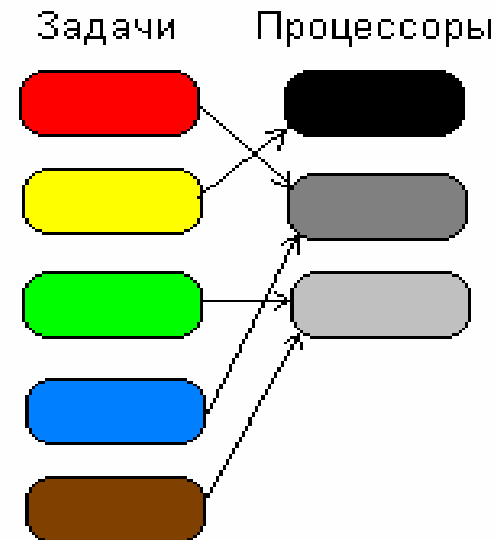
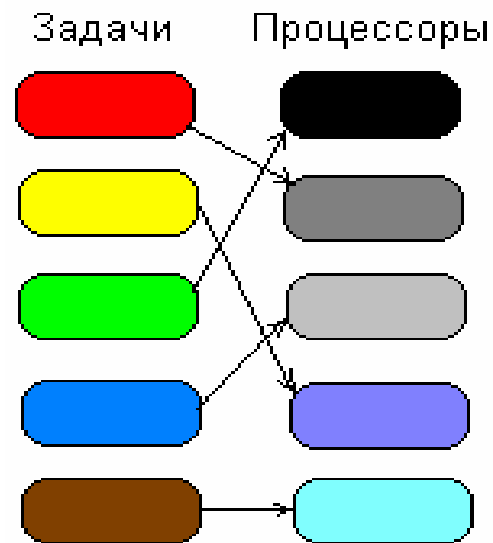
Б) Истинная
параллельность



В) Псевдо- (квази-)
параллельность



Аппаратные возможности



Ядро ОС. Диспетчер задач (продолжение)

Диспетчер задач Windows

Файл Параметры Вид Завершение работы Справка

Приложения Процессы Быстродействие Сеть Пользователи

Имя образа	PID	Имя пользоват...	ЦП	Память	Баз.пр.	Потоков
VMNAT.EXE	4024		00	2 324 КБ	Средний	4
ALG.EXE	3936	LOCAL SERVICE	00	3 376 КБ	Средний	7
wscntfy.exe	3772	Administrator	00	2 072 КБ	Средний	1
vmnetdhcp.exe	3112		00	1 968 КБ	Средний	2
WMIPRVSE.EXE	3012		00	6 548 КБ	Средний	7
wuauclt.exe	2836	SYSTEM	00	8 296 КБ	Средний	9
vmware-usbarbitr...	2384		00	2 668 КБ	Средний	2
nimxs.exe	2012		00	6 756 КБ	Средний	10
LKTSRV.EXE	1856		00	4 068 КБ	Средний	13
LKADS.EXE	1812		00	3 948 КБ	Средний	7
LKCITDL.EXE	1780		00	4 288 КБ	Средний	11
FsUsbExService.Exe	1756		00	2 524 КБ	Средний	2
SVCHOST.EXE	1704		00	23 392 КБ	Средний	89
SVCHOST.EXE	1644		00	3 208 КБ	Средний	4
taskmgr.exe	1488		01	4 500 КБ	Высокий	3
NPSAgent.exe	1296		00	4 164 КБ	Средний	4
ATI2EVXX.EXE	1276		00	4 640 КБ	Средний	5
CTFMON.EXE	1232		00	2 892 КБ	Средний	1
hqtray.exe	1224		00	4 972 КБ	Средний	2
Winampa.exe	1220		00	1 936 КБ	Средний	1
DAEMON.EXE	1164		00	7 012 КБ	Средний	2

☒ Отображать процессы всех пользователей

Завершить процесс

Процессов: 44 Загрузка ЦП: 2% Выделение памяти: 223МБ / 6

Ядро ОС. Диспетчер задач (продолжение)



Задачи переключаются по собственному желанию.

Достоинства:

- Простота реализации
- Предсказуемость поведения задач

Недостатки:

- Неустойчивость к сбоям.

Примеры: Windows 1.X - 3.X, некоторые ОСРВ.

Задачи переключаются операционной системой

Недостатки:

- Сложность реализации
- Непредсказуемость поведения задач

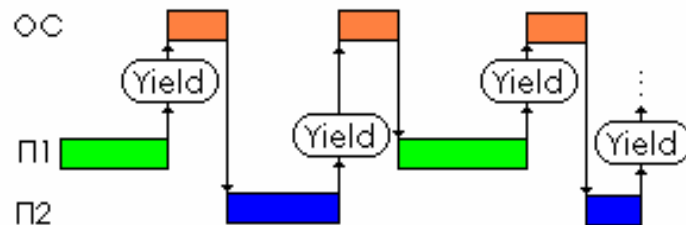
Достоинство:

- Устойчивость к сбоям.

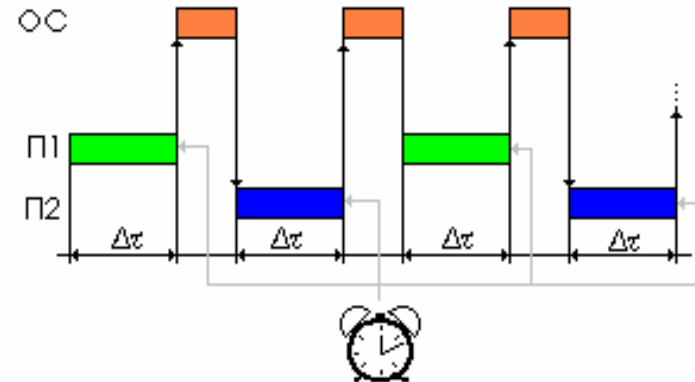
Примеры: все современные ОС

Ядро ОС. Диспетчер задач (продолжение)

А) Кооперативная м/з



Б) Вытесняющая м/з



Причины переключения задач

Кооперативная м/з	Вытесняющая м/з
	Желание ОС
Желание задачи	Желание задачи
Желание пользователя	Желание пользователя
Завершение задачи	Завершение задачи

Ядро ОС. Диспетчер задач (продолжение)

```
// Вывод строки по отдельным буквам
printstr(char *s) {
    int i = 0;
    while (s[i] != 0) cout << s[i++];
}
```

А) Кооперативная м/з

```
while (1) {
    printstr("Hello ");
    Yield();
}
```

```
while (1) {
    printstr("world!\n");
    Yield();
}
```

```
Hello world!
Hello world!
Hello world!
```

Б) Вытесняющая м/з

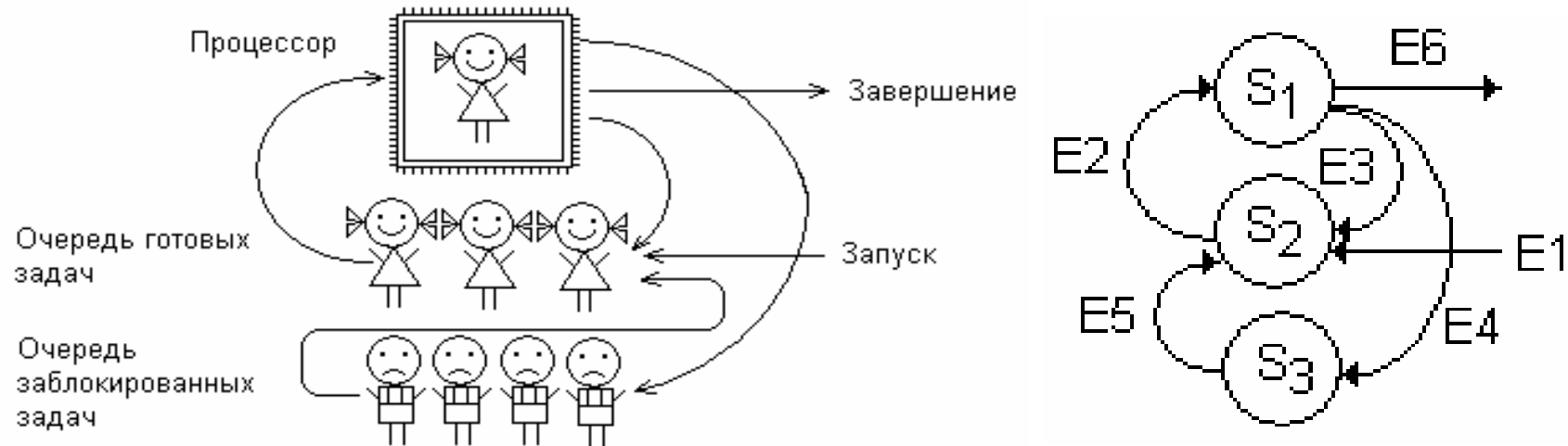
```
while (1) {
    printstr("Hello ");
}
```

```
while (1) {
    printstr("world!\n");
}
```

```
HelloHellworld!
wooHell Hellrld!
worl
```

Ядро ОС. Диспетчер задач (продолжение)

Граф многозадачности



Состояния:

S_1 — выполнение; **S_2** — ожидание; **S_3** — блокировка.

События:

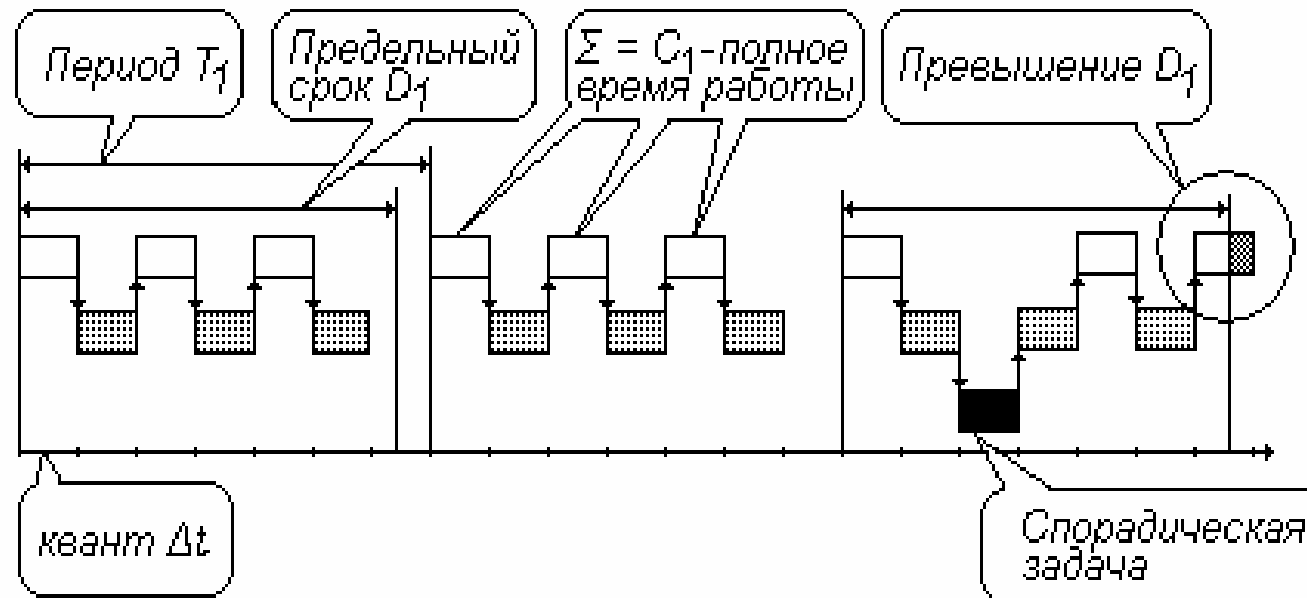
E_1 — запуск; **E_2** — начало выполнения; **E_3** — вытеснение; **E_4** — блокировка; **E_5** — снятие блокировки; **E_6** — завершение.

Ядро ОС. Диспетчер задач (продолжение)

Модель многозадачности

Типы задач:

- 1) периодические;
- 2) спорадические;
- 3) фоновые.

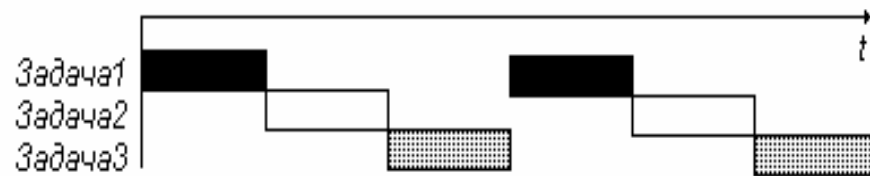


Условие работоспособности:
$$\mu = \sum_{i=1}^N \frac{C_i}{T_i} \leq 1$$

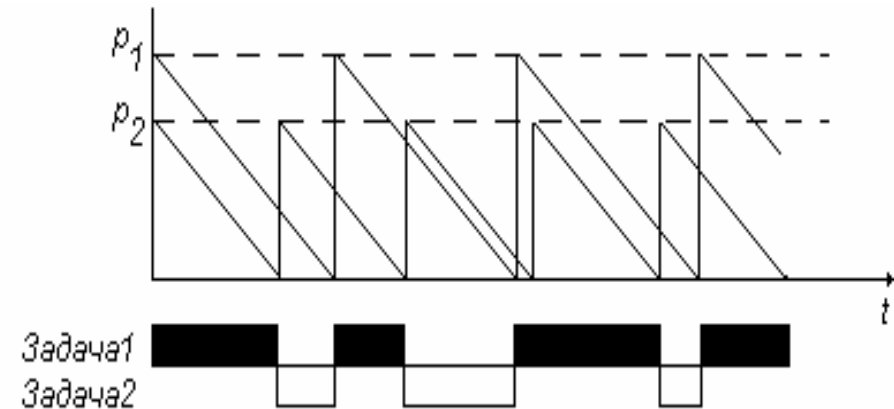
Ядро ОС. Диспетчер задач (продолжение)

Алгоритмы работы планировщика задач
(алгоритмы диспетчеризации)

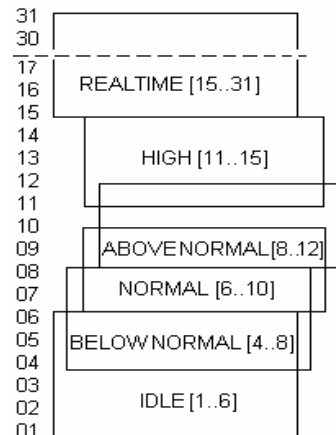
1) RR - Round Robin («карусель»)



2) UNIX (старение приоритетов)



3) Windows (классы приоритетов)



Факторы:

- Открытое окно
- Фокус ввода
- «Возраст» задачи
- «Голод» задачи

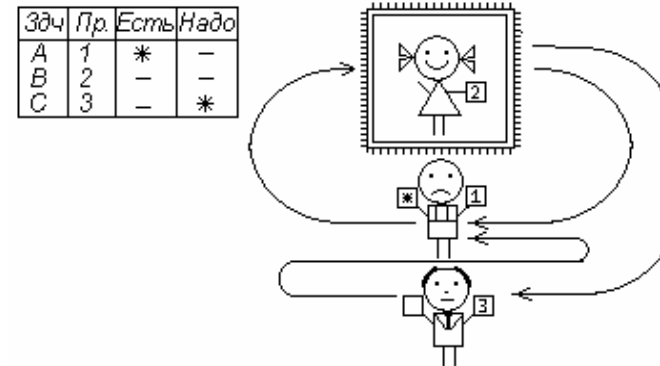
4) Алгоритмы реального времени:

- **RMS** (Rate Monotonic Scheduling) – чем меньше T , тем выше приоритет.
- **EDF** (Earliest Deadline First) – чем меньше до D , тем выше приоритет.

Ядро ОС. Диспетчер задач (продолжение)

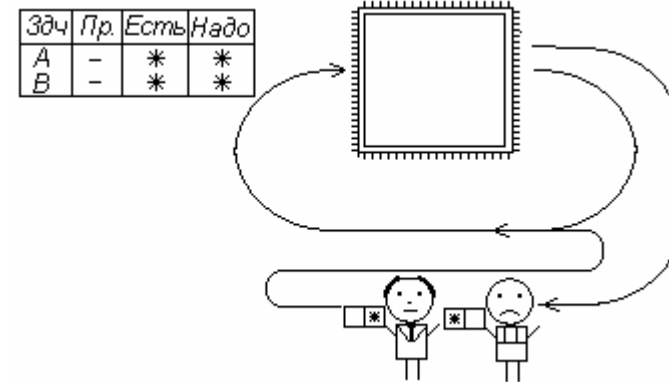
Проблемы диспетчеризации:

- 1) «Голодание» задач (job starvation),
решение – «разгон» приоритетов (priority boosting)
- 2) Инверсия приоритетов;



решение – «наследование» приоритетов;

- 3) «Гонки» (race condition);
- 4) «Взаимоблокировка» (deadlock).

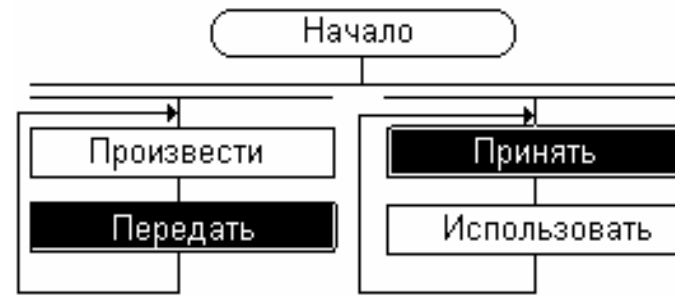


Ядро ОС. Диспетчер задач (продолжение)

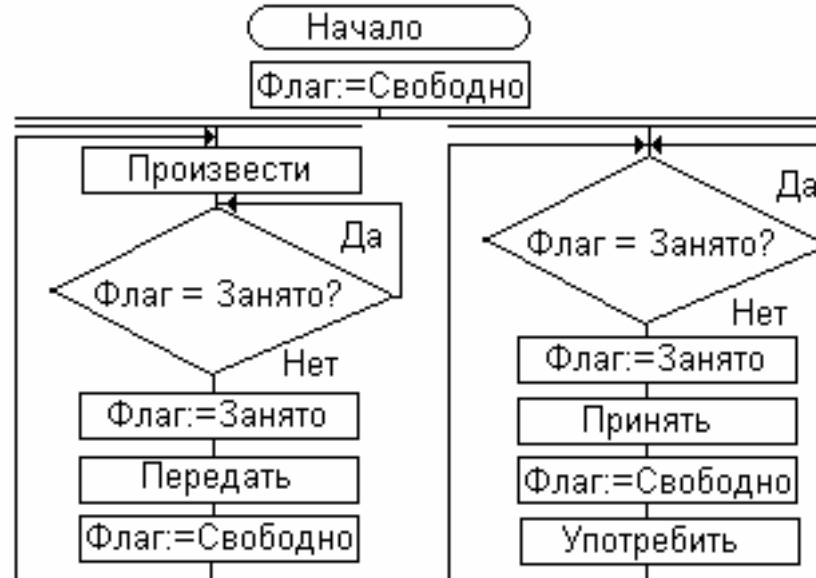
Модель «поставщик-потребитель»,

условия: 1) неодновременность доступа; 2) попеременность доступа.

- А) Однозадачный алгоритм Б) Многозадачный алгоритм без синхронизации

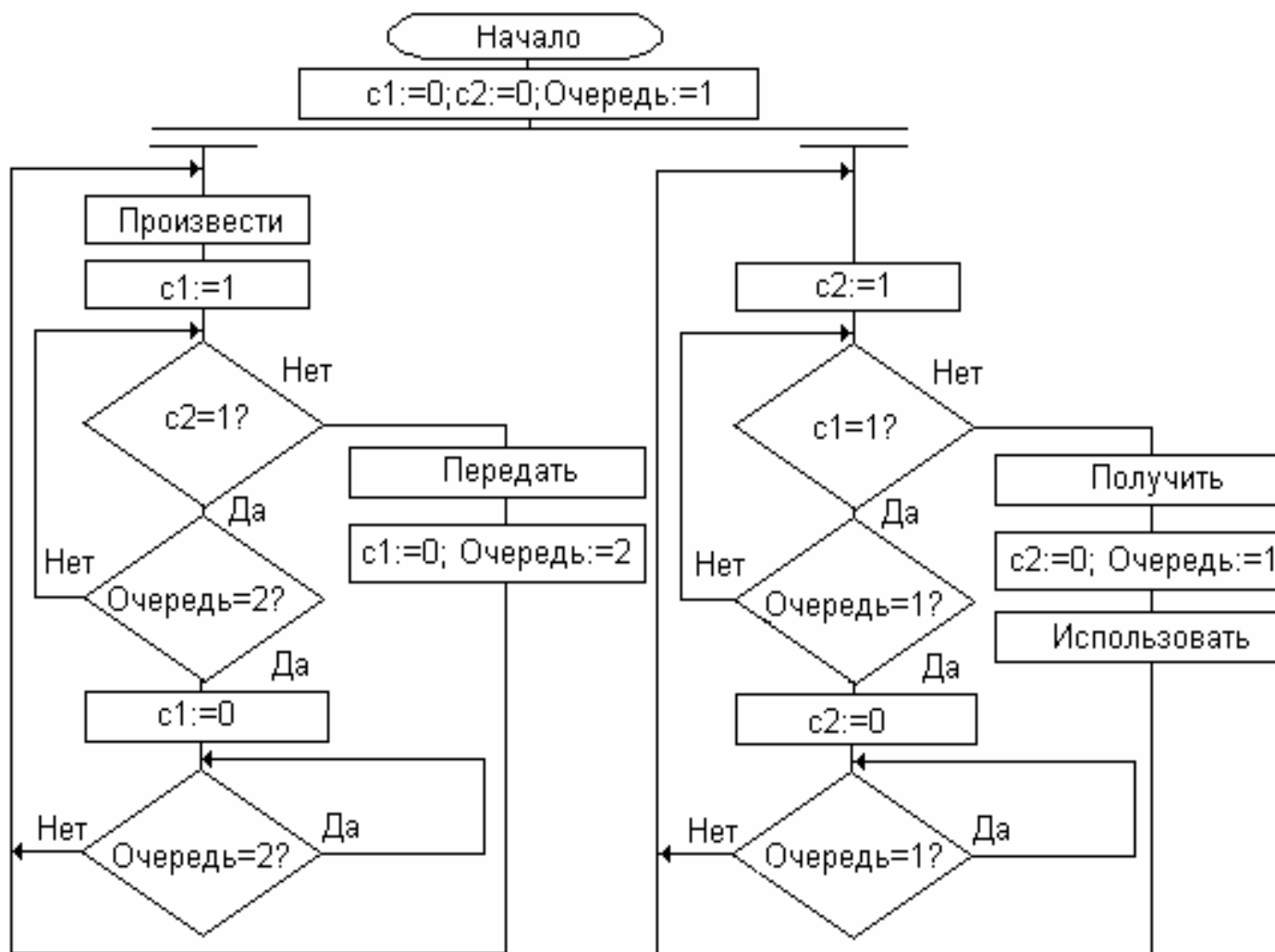


- В) алгоритм с блокирующим флагом



Ядро ОС. Диспетчер задач (продолжение)

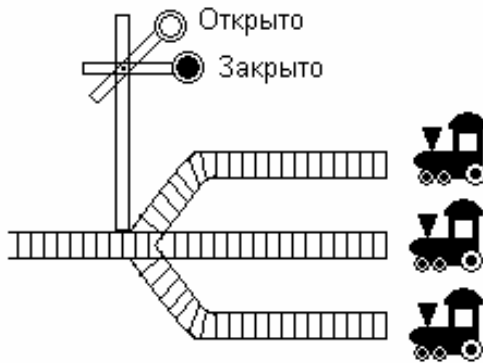
Г) Алгоритм Деккера-Холта



Другой вариант: алгоритм Петерсона

Ядро ОС. Диспетчер задач (продолжение)

Д) Алгоритм с использованием семафоров Дейкстры



1. Целая переменная S

2. $P(S)$ - операция «Оградить»

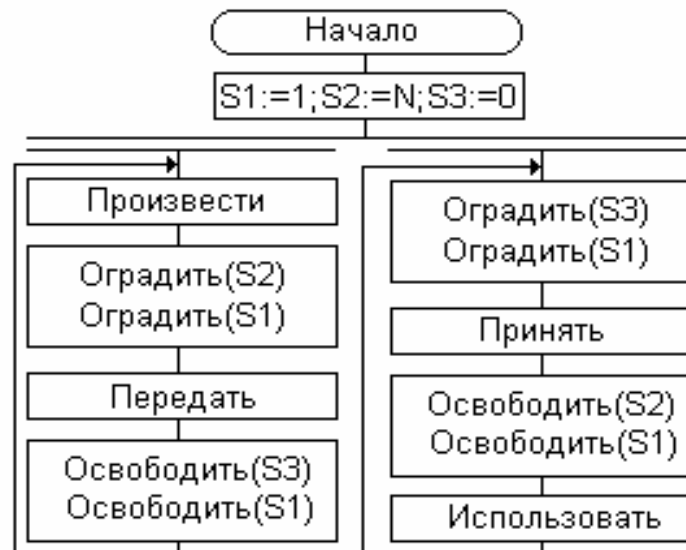
- $S := S - 1$

- Если $S < 0$, то текущая задача встает в очередь

3. $V(S)$ – операция «Освободить»

- $S := S + 1$

- Если $S > 0$, то 1-я в очереди задача продолжает работу



Ядро ОС. Диспетчер задач (продолжение)

Метод организации атоммарности - Монитор (Хоара)

1) Команды типа «Проверка и установка»

; TEST_AND_SET Локальная, Общая

Локальная := Общая

Общая := 1

2) Запрет прерываний

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				O	D	I		S	Z				P		C

CLI

...

...

...

STI

Ядро ОС. Диспетчер задач (окончание)

Особенности старта процессов в UNIX

```
x = fork();  
cout << x;  
if (!x) { cout << " Я новенький"; exec(...); }  
else      cout << " Я старенький";
```

1 Я старенький 0 Я новенький

Финальный обзор методов синхронизации

Windows			UNIX		
Спин-блокировка	(+)	?	Разделяемая память	+	+
Семафор	+	+	Файлы	+	+
Мутекс	+	+	Отображаемые файлы	+	+
Critical section	+	-	Каналы	+	+
Событие	+	-	Почтовые ячейки	+	+
Сигнал	-	+	Сокеты	+	+
Сообщение	+	+			

Ядро ОС. Диспетчер ввода-вывода

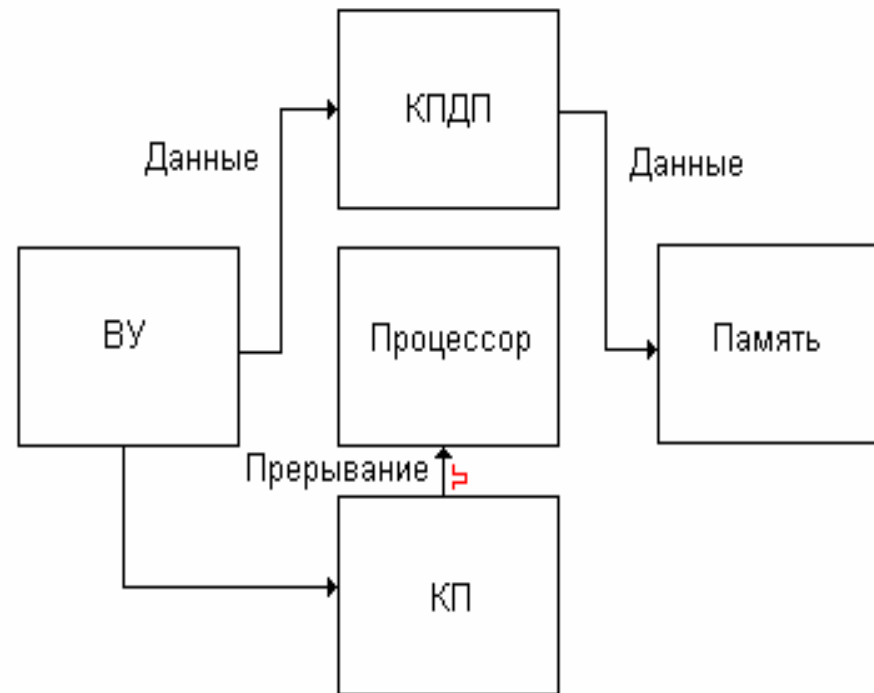
Назначение:

- Управление внешними устройствами
- Передача данных на внешние устройства
- Прием данных со внешних устройств

А) Режим PIO

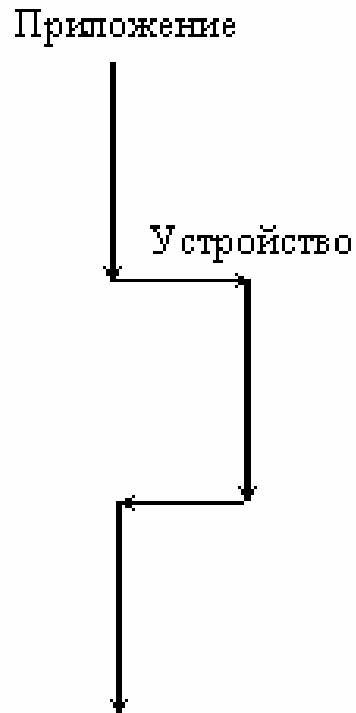


Б) Режим DIO

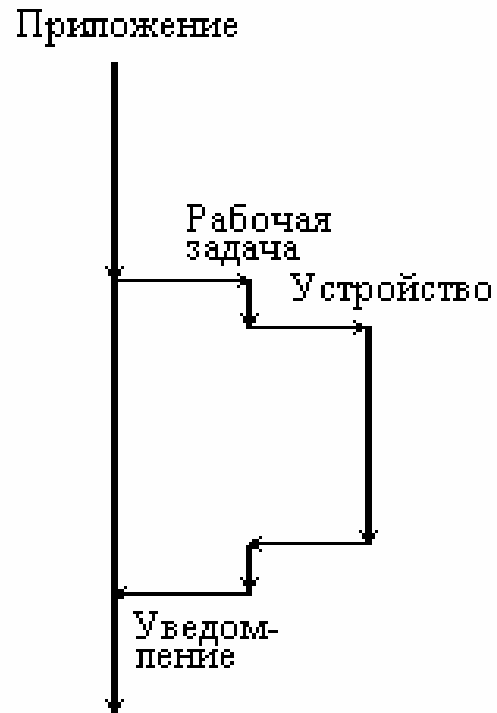


Ядро ОС. Диспетчер ввода-вывода (продолжение)

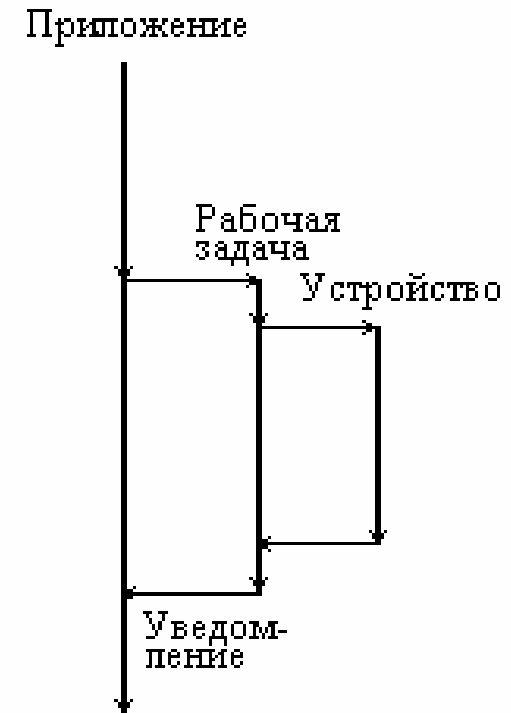
А) Синхронный
ВВОД-ВЫВОД



Б) Асинхронный
ВВОД-ВЫВОД
с ожиданием
завершения



В) Асинхронный
ВВОД-ВЫВОД с
сигналом завершения



Ядро ОС. Диспетчер ввода-вывода (продолжение)

А) Пример синхронного чтения

```
// Чтение даты создания BIOS
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>

int f; // Дескриптор файла
unsigned char buf[8]; // Буфер для чтения

main() {
    // Открыть адресное пространство
    f = open("/dev/mem", O_RDONLY, 0);

    // Переместить указатель чтения
    lseek(f, 0xFFFFF5, SEEK_SET);

    // Читать
    read(f, buf, 8);

    // Закрыть
    close (f);
}
```

б) Пример асинхронного чтения

```
// Чтение генератора случайных чисел
#include <sys/types.h>
#include <aio.h>
#include <fcntl.h>
#include <errno.h>

struct aiocb cb; // Описатель параметров I/O
int number; // Буфер для чтения
int f; // Дескриптор файла

int main() {

    // Открыть генератор случайных чисел
    f = open("/dev/random", O_RDONLY, 0);

    // Очистить описатель
    memset(&cb, 0, sizeof(struct aiocb));
    // Заполнить некоторые поля
    cb.aio_nbytes = sizeof(int); // Сколько читать
    cb.aio_fildes = f; // Из какого файла
    cb.aio_offset = 0; // С какого смещения
    cb.aio_buf = &number; // По какому адресу

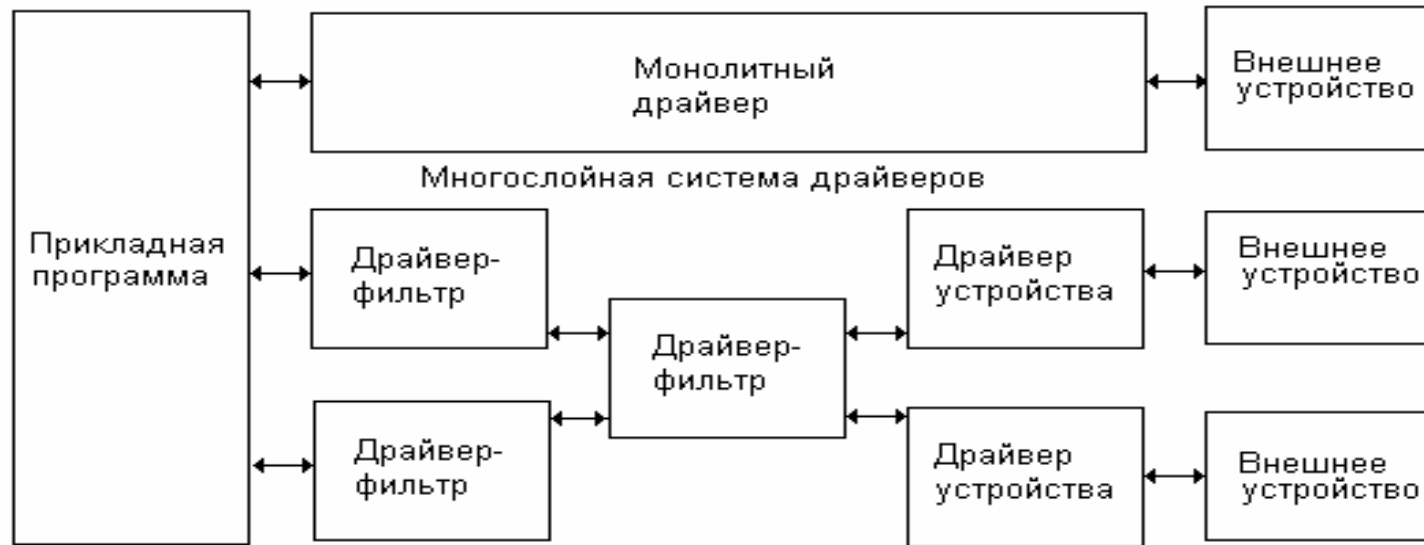
    // Запустить чтение
    aio_read(&cb);

    while (1) {
        if (aio_error(&cb) == EINPROGRESS)
            { close (f); break; }

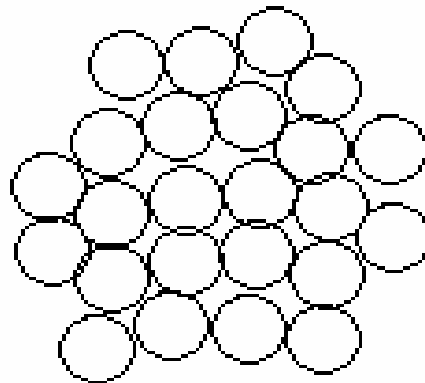
        ...
        // Здесь выполнять какую-то работу
        ...
    }
}
```

Ядро ОС. Диспетчер ввода-вывода (продолжение)

Драйвера



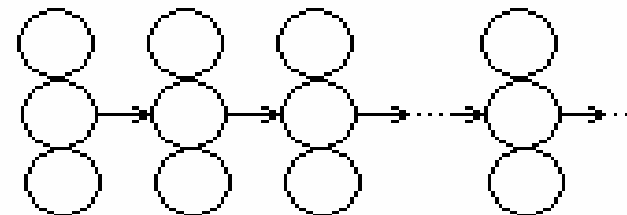
Набор данных



Символьное устройство

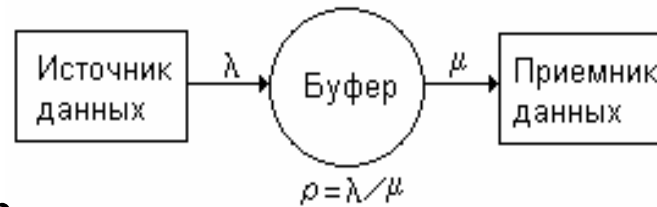


Блочное устройство

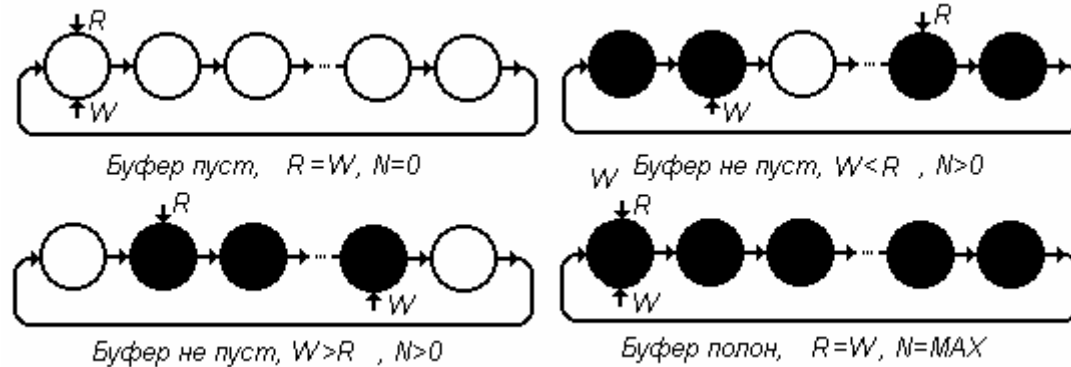


Ядро ОС. Диспетчер ввода-вывода (окончание)

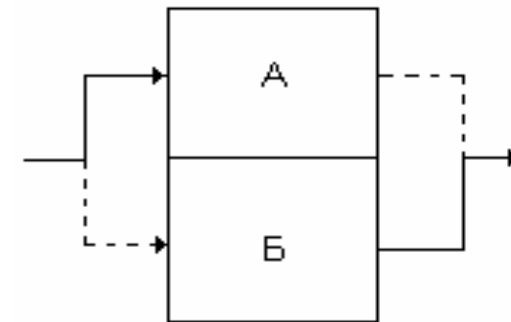
Буферизация данных



А) Кольцевой буфер



Б) Двухтактный (двойной) буфер



Вероятность переполнения буфера размером m:

$$P_m = \rho^{m+1} \frac{1 - \rho}{1 - \rho^{m+2}} = \frac{\rho^{m+1}}{\sum_{i=0}^{m+1} \rho^i}.$$

Ядро ОС. Файловая система

Файл = именованный набор данных.

Расположение:

- в Windows только на носителях;
- в UNIX – в/на любом устройстве.

Файловая система:

- 1) набор форматов и алгоритмов, описывающих расположение данных;
- 2) компонент ОС, поддерживающий этот набор форматов и алгоритмов.

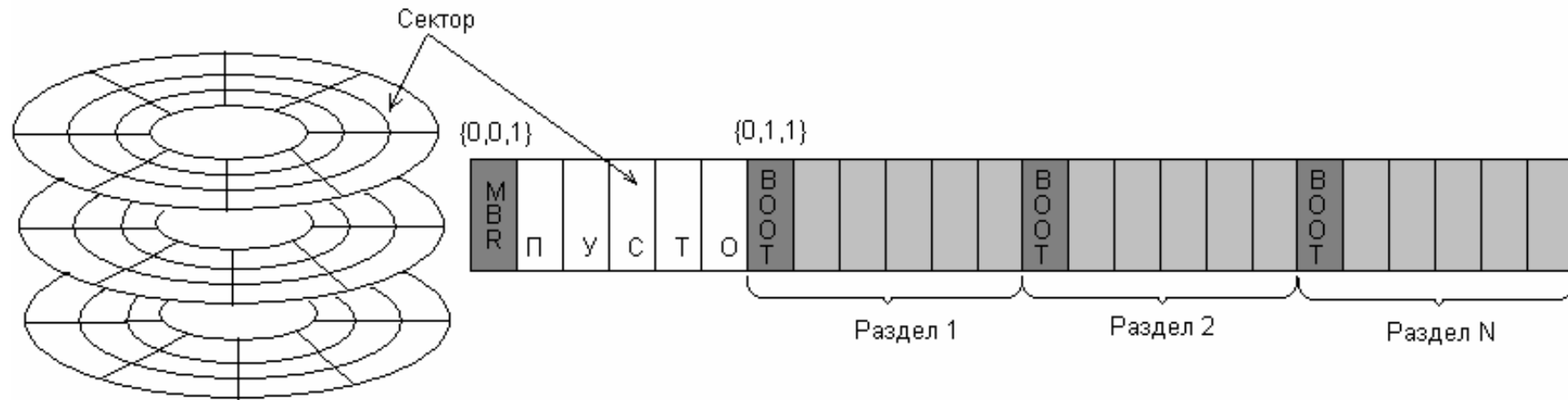
Примеры файловых систем:

FAT (FAT12, FAT16, FAT32, VFAT)	MS-DOS, Windows 9X, Windows NT, Unix
NTFS	Windows NT, Unix
ISO 9660	Windows 9X, Windows NT, Unix
CDFS	Unix
Joliet / Romeo	MS-DOS
EXT*FS (EXTFS, EXT2FS, EXT3FS, EXT4FS)	Unix
UFS	Unix

Ядро ОС. Файловая система (продолжение)

Физическая организация носителей информации

1) Жесткий диск (HDD)



Виды адресации секторов:

- CHS = {цилиндр, головка, сектор};
- $LBA = N_S \times N_H \times C + N_S \times H + S - 1$.

MBR (Master Boot Record – главная загрузочная запись) =

Таблица описания разделов диска + программа поиска и загрузки активного Boot-сектора.

Boot-сектор (загрузочный сектор) =

Таблица описания структуры раздела + программа начала загрузки ОС.

Ядро ОС. Файловая система

Доступ к MBR:

```
// Windows NT
BYTE mbr[512];
DWORD dwRead;
HANDLE hDisk =
CreateFile("\\\\.\\PhysicalDrive0",
    GENERIC_READ, FILE_SHARE_READ,
    NULL, OPEN_EXISTING, 0, NULL);
ReadFile(hDisk, &mbr, 512, &dwRead,
    NULL);
CloseHandle(hDisk);

// UNIX
unsigned char buf[512];
int f = open("/dev/hda", O_RDONLY);
read(f, buf, 512);
close(f);
```

Структура строки:

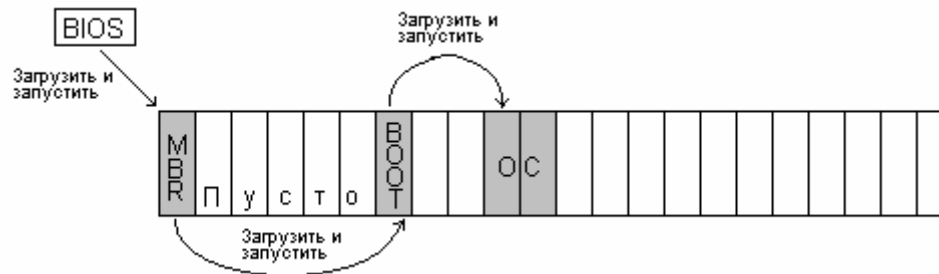
+00 – метка активности;
+04 – тип ф/с;
+1С – длина раздела.

00000	FA	33	C8	8E	D8	BC	00	7C	8B	F4	50	07	50	1F	FB	FC	ЪЗАЪРј. <ФРЪРЪЪЪ
00010	BF	00	06	B9	00	01	F2	A5	EA	1D	06	00	00	BE	BE	07	й.ЪЪ.ЪтГКЪЪ..ssЪ
00020	B3	04	80	3C	80	74	0E	80	3C	00	75	1C	83	C6	10	FE	іЪЪ<ЪтЪЪ<.uЪГЖЪЪ
00030	CB	75	EF	CD	18	8B	14	8B	4C	02	8B	EE	83	C6	10	FE	лuПНЪ<Ъ<LЪ<oГЖЪЪ
00040	CB	74	1A	80	3C	00	74	F4	BE	8B	06	AC	3C	00	74	0B	лтЪЪ<.tфs<Ъ-<.tЪ
00050	56	BB	07	00	B4	0E	CD	10	5E	EB	F0	EB	FE	BF	05	00	U>>Ъ.ГЪНЪ^лрлюіЪ
00060	BB	00	7C	B8	01	02	57	CD	13	5F	73	0C	33	C0	CD	13	>>. ъЪЪМНЪ_sЪЗАНЪ
00070	4F	75	ED	BE	A3	06	EB	D3	BE	C2	06	BF	FE	7D	81	3D	0uнсJЪлYsBЪіu}Ъ'=
00080	55	AA	75	C7	8B	F5	EA	00	7C	00	00	49	6E	76	61	6C	UEu3<xкк. ..Inval
00090	69	64	20	70	61	72	74	69	74	69	6F	6E	20	74	61	62	id partition tab
000A0	6C	65	00	45	72	72	6F	72	20	6C	6F	61	64	69	6E	67	le.Error loading
000B0	20	6F	70	65	72	61	74	69	6E	67	20	73	79	73	74	65	operating syste
000C0	6D	00	4D	69	73	73	69	6E	67	20	6F	70	65	72	61	74	m.Missing operat
000D0	69	6E	67	20	73	79	73	74	65	6D	00	00	80	36	14	20	ing system..Ъ6Ъ
000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001C0	01	00	06	1F	3F	98	3F	00	00	00	A1	B4	04	00	00	00ЪЪ
001D0	01	99	05	1F	FF	39	E0	B4	04	00	E0	B3	14	00	00	00	Ъ.ЪЪ?■?...ŸrЪ...

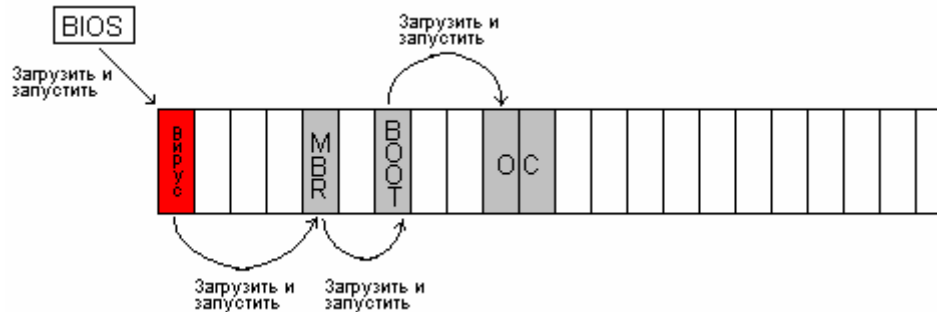
Ядро ОС. Файловая система (продолжение)

Загрузка операционных систем

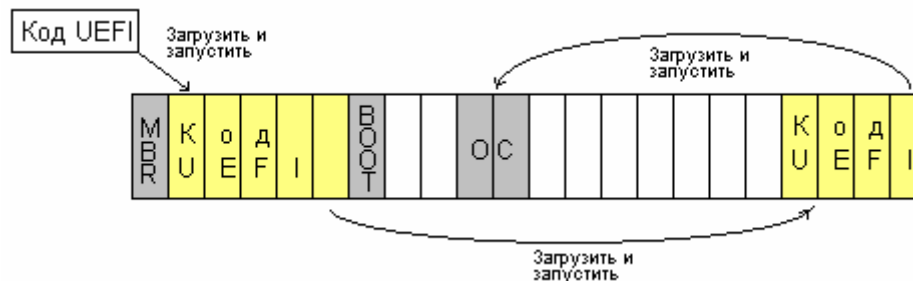
А) По правилам BIOS



Б) В случае вируса/буткита (а так же boot-менеджеров Grub, Lilo, Ontrack...)



В) В случае UEFI (диски с GPT – GUID partition table)



Ядро ОС. Файловая система (продолжение)

Отличия BIOS от UEFI

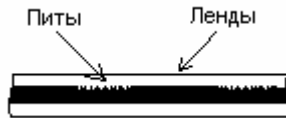
	BIOS	UEFI
Расположение программы	F000:0000	Переменное
Размер программы	64 Кб	До сотен Мб
Режим работы программы	Реальный	Защищенный
Пользовательский интерфейс	Текстовый	Графический
Поддерживаемый размер носителей	До 2 Тб	До 3 Пб
Поддерживаемое кол-во разделов	До 4	До 256
Работа с сетью	Нет	Да
Работа с гипервизором процессора	Нет	Да
Защита от несанкц. использования ОС	Нет	Да



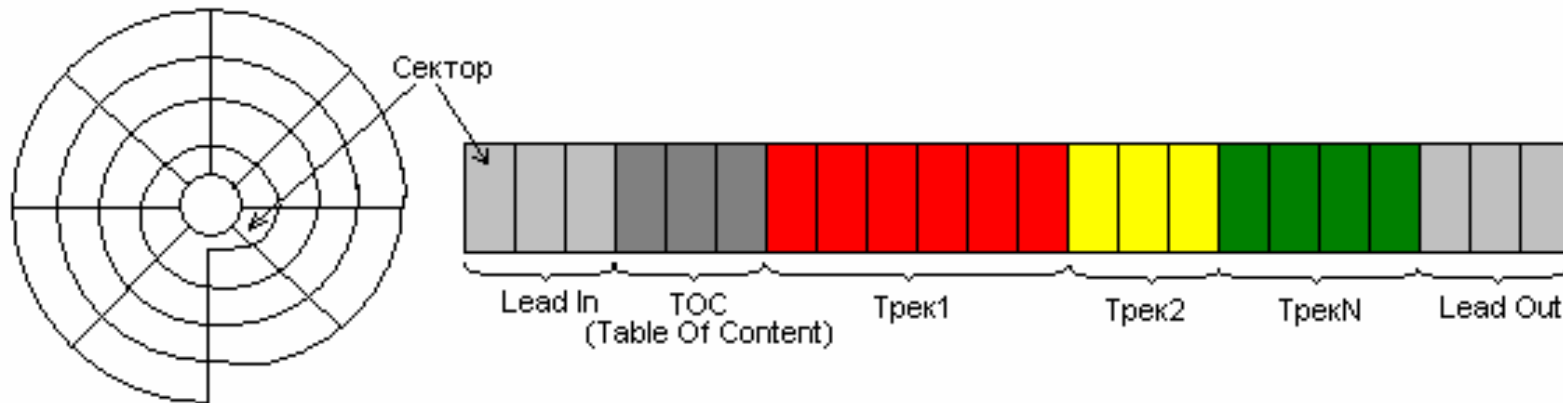
Ядро ОС. Файловая система (продолжение)

2) Компакт-диски (CD и DVD)

А) Принцип записи



Б) Физическая организация

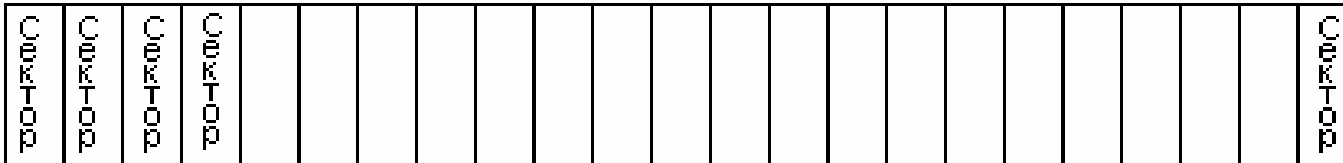


Типы треков:

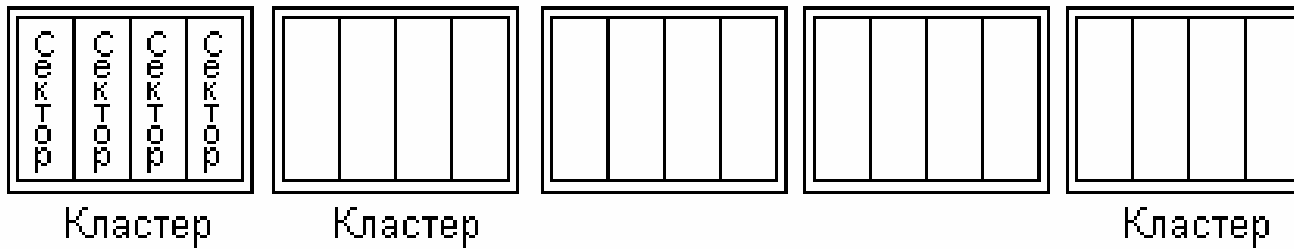
- треки данных;
- треки аудио;
- треки видео (только на DVD).

Ядро ОС. Файловая система (продолжение)

Физическая структура диска – набор секторов

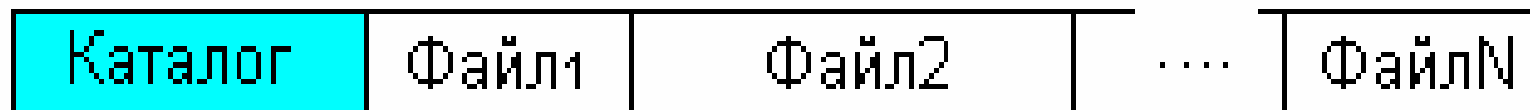


Логическая структура диска – набор кластеров



Ядро ОС. Файловая система (продолжение)

1. Простейшая последовательная ФС



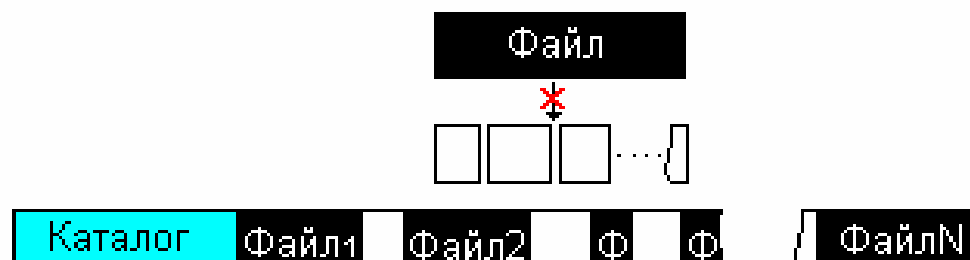
Каталог (директория) = база данных о файлах (имя, расположение на диске, длина, прочие характеристики).

Структура записи каталога:

- имя файла;
- атрибуты файла (защита от записи, дата и время создания и пр.);
- длина файла;
- адрес первого кластера.

Достоинство: простота

Недостаток: фрагментация дискового пространства.



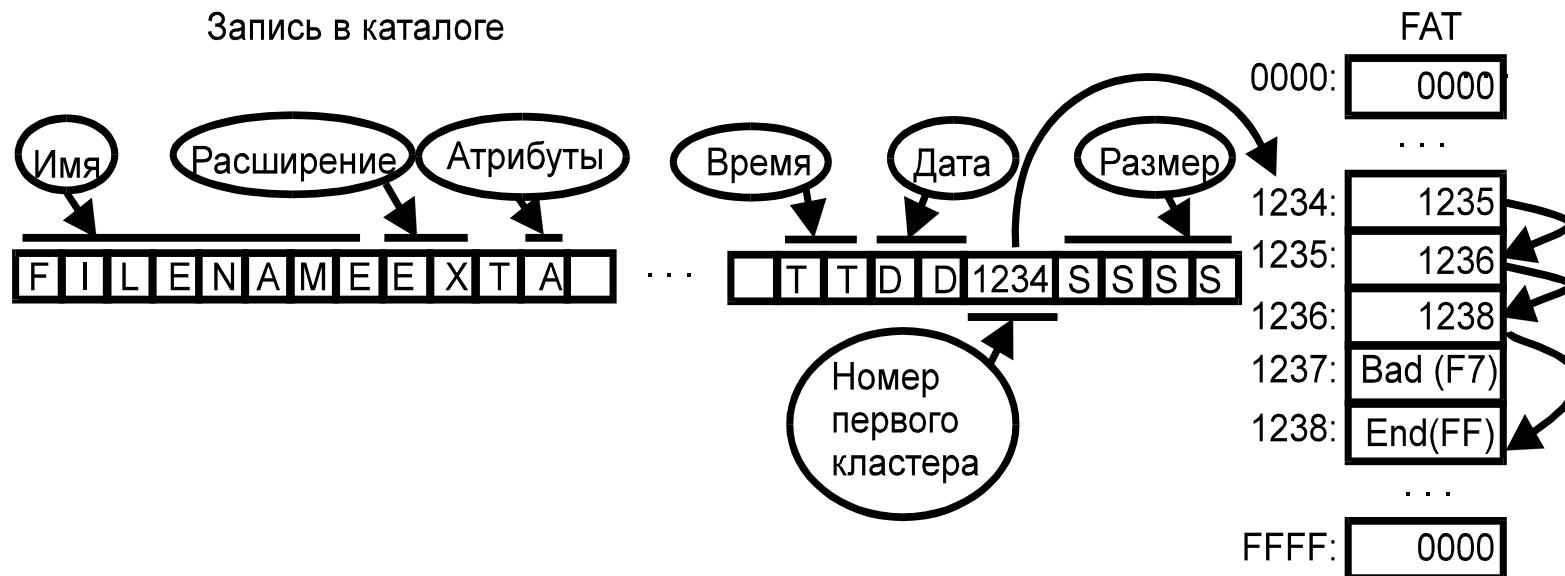
Ядро ОС. Файловая система (продолжение)

2) Файловые системы семейства FAT (MS-DOS, Windows)



Структура записи каталога:

- имя в формате либо «8.3», либо «256»);
- атрибут (защита от записи, скрытый, системный и т.п.);
- длина в байтах;
- дата и время создания файла;
- номер первого кластера в цепочке.



Ядро ОС. Файловая система (продолжение)

Каталоги FAT – файлы (кроме ROOT, который область на диске)

Имя	Атрибуты	Адрес начала цепочки
MYFILE.DAT	10
VASYA.TXT	20
MASHA.EXE	30
xANYA.JPG	??
PUPKIN	50
VASYA.TXT	60

Нарушения:

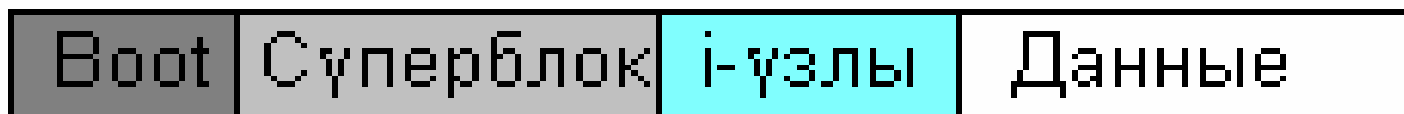
- Одинаковые имена (вирус DIR.1024)
- Пересекающиеся цепочки кластеров
- Потерянные кластеры.

Максимальные размеры:

- FAT12: $2^{12} = 2048$ кластеров
- FAT16: $2^{16} = 65535$ кластеров ~ 520 Мб
- FAT32: $2^{32} = 4.2$ млрд кластеров

Ядро ОС. Файловая система (продолжение)

3) Файловые системы для UNIX (UFS, EXT*FS,...)



Структура i-узла (описано в POSIX):

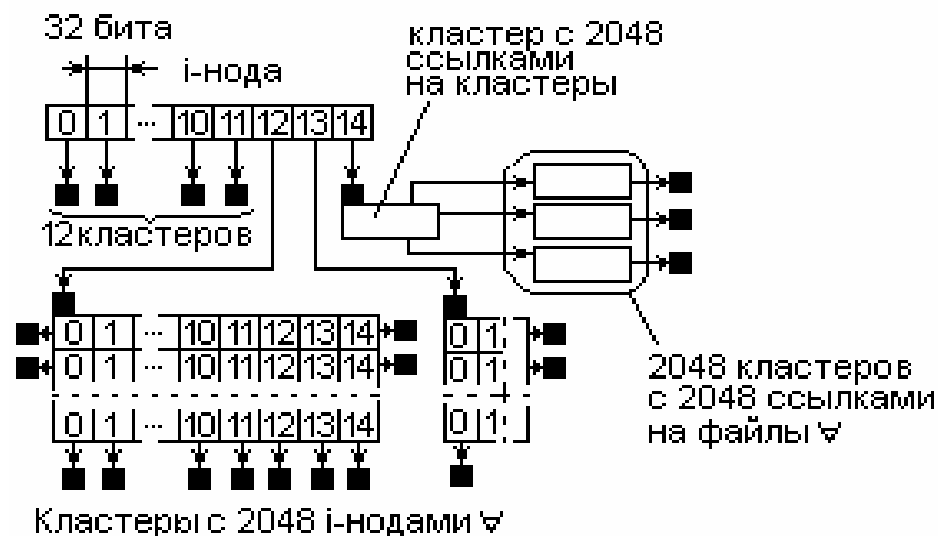
- Индекс (номер) файла;
- Тип файла (файл/каталог/канал/сокет/спецфайл);
- **Биты доступа;**
- ID хозяина;
- Время/дата создания/модификации/последнего доступа к данным;
- Длина файла;
- Счетчик количества копий;
- Описание цепочки кластеров.

Каталог – файл особого вида.

Структура:

#	Имя
4	myfile
⑤	vasya
6	masha

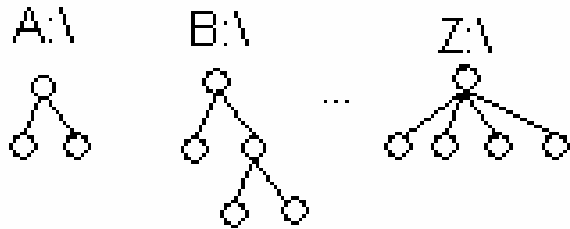
#	Имя
7	tanya
⑤	purkin



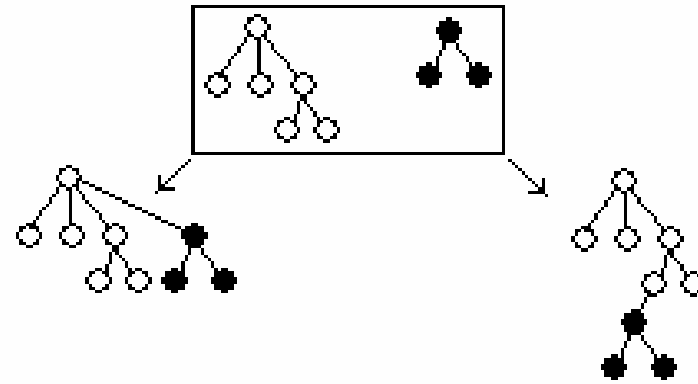
Ядро ОС. Файловая система (продолжение)

Топологии файловых систем

А) **FAT** и **NTFS** — лес деревьев



б) **UFS** — единственное дерево



Операция монтирования:

```
> mount /dev/sda1 /mnt/sda1_removable -rw  
> umount /dev/sda1
```

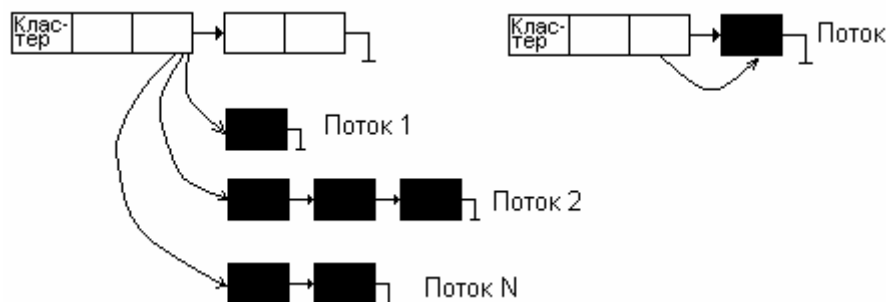
Ядро ОС. Файловая система (продолжение)

3) NTFS (только Windows NT)



Первые 16 записей MFT: \$MFT – сам MFT; \$LogFile – файл поддержки журналирования; \$Volume – служебная информация о томе; \$Bitmap – карта свободных мест и пр.

Записи в MFT:



Стандартные потоки: \$Data – данные файла; \$Date – дата создания; \$Time – время создания; \$Name – имя файла и т.п.

Программный доступ к потокам:

```
h = CreateFile("c:\\file.txt::$Time", ...);  
h = CreateFile("c:\\file.txt:$MyStream", ...);
```

Ядро ОС. Файловая система (окончание)

Особенности NTFS:

- позволяет сжимать потоки данных методом LZNT1.
- позволяет шифровать потоки данных методами DESX или AES.
- поддерживает откаты операций;
- позволяет содержать в записях MFT списки SACL и DACL.

Каталог NTFS:

- Индекс (номер) файла;
- Имя файла;
- Атрибуты;
- Списки кластеров;
- ID хозяина;
- Списки SACL и DACL;
- Служебные данные для шифрования, сжатия и журналирования.

Ядро ОС. Подсистема защиты

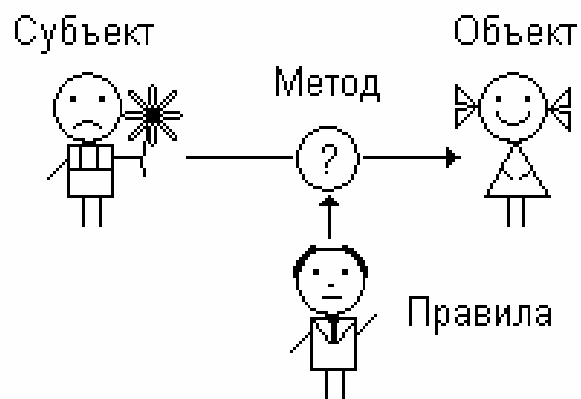
В MS-DOS, Windows 9X – нет. В Windows NT, UNIX, iOS и т.п. – есть.

В ОСРВ присутствует в минимальном объеме.



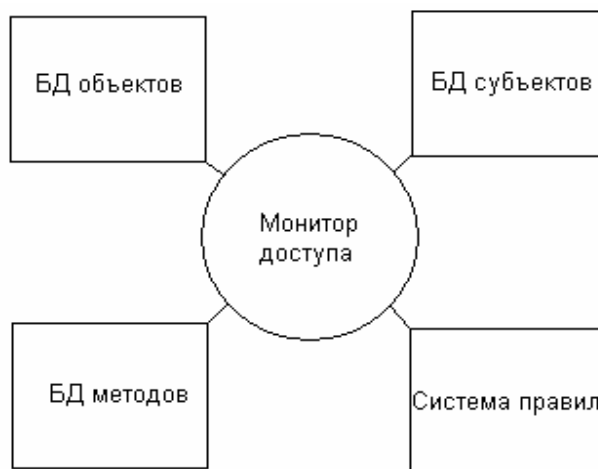
Ядро ОС. Подсистема защиты (продолжение)

1. Разграничение доступа

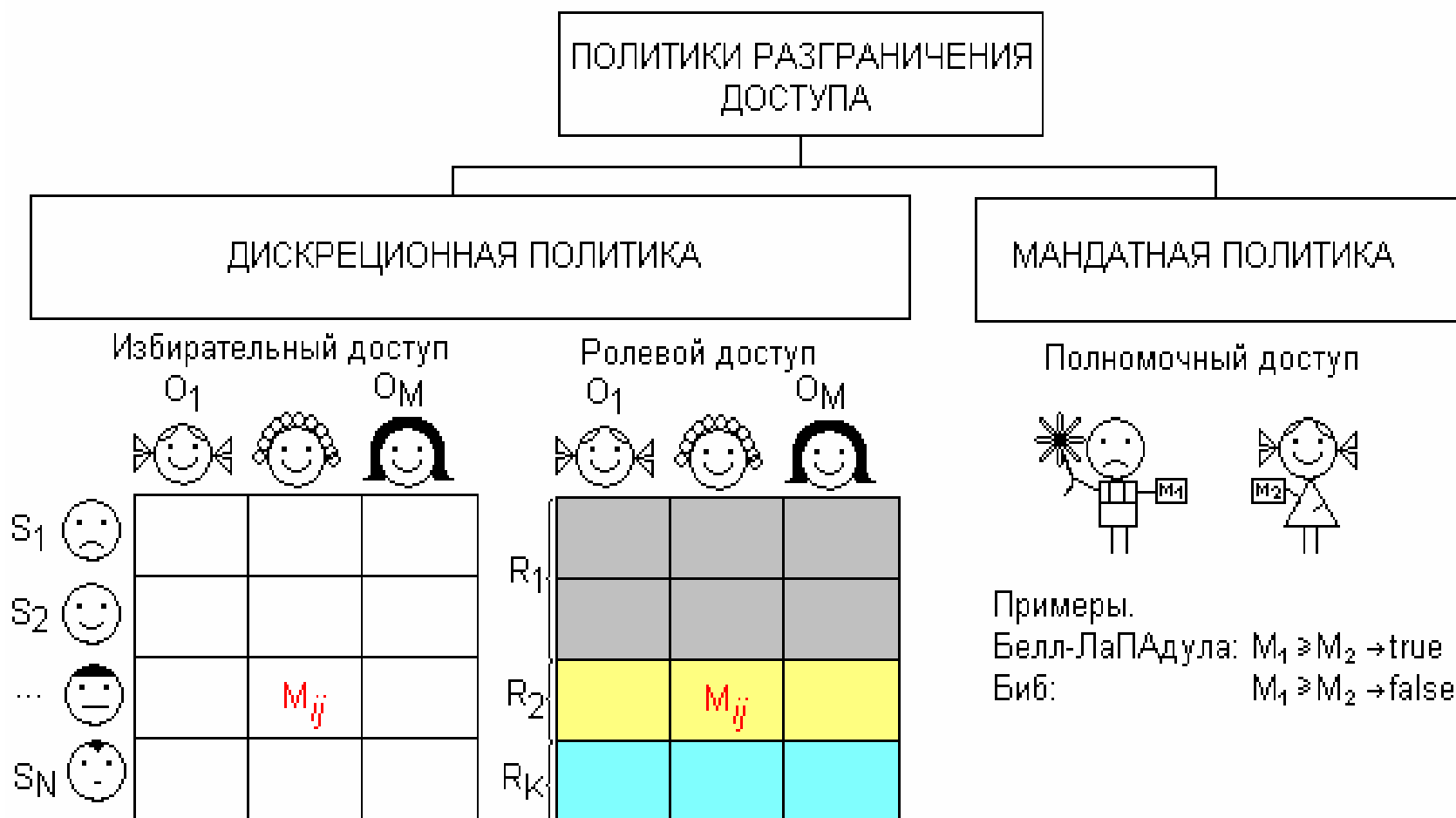


- 1) $\{S\}$ – множество субъектов;
- 2) $\{O\}$ – множество объектов;
- 3) $\{M\}$ – множество методов
- 4) $\{P\} = \{ P(S,O,M) \rightarrow (0,1) \}$ – множество правил доступа.

Архитектура п/с разграничения доступа в ОС



Ядро ОС. Подсистема защиты (продолжение)



Примеры.

- Кольца защиты
- Изоляционизм
- Программируемое р/д в файловых системах

Ядро ОС. Подсистема защиты (продолжение)

Разграничение доступа в UNIX

Субъекты:

- Пользователи и группы пользователей

Объекты:

- Файлы и устройства
- Каталоги

Поле типа:

«d» - каталог, «b» - файл на блочном устройстве,
«c» - файл на символьном устройстве, «s» - сокет, «p» - канал,
«l» - ссылка.

Тип	Владелец			Группа владельца			Остальные		
	r	w	x	r	w	x	r	w	x

Команды ls и chmod

```
root@slax:/home# ls -l
total 12
-rwxr-xr-x 1 root  root  8035 Dec 16 13:41 a.out
-rw-r--r-- 1 root  root    54 Dec 16 13:41 example.c
drwxr-xr-x 2 guest users   3 May  6 2006 guest
```

Ядро ОС. Подсистема защиты (продолжение)

Разграничение доступа в Windows

Субъекты:

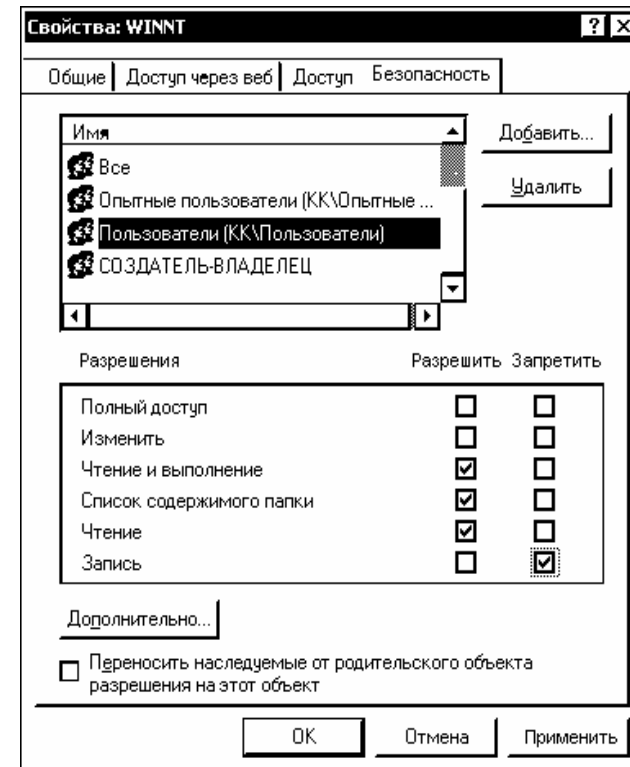
Пользователи и группы пользователей

Объекты:

файлы; каталоги (директории, папки);
устройства (диски, порты, клавиатура,
мышь и т.п); средства передачи данных
между процессами; ключи реестра;
процессы и потоки; сервисы и диспетчер
сервисов; рабочие столы и окна;
фрагменты разделяемой памяти;
символические связи; маркеры доступа;
объекты синхронизации.

Правила раскрытия противоречий:

- Приоритетность первого упоминания;
- Запрет приоритетней разрешения;
- Приоритетность группы над субъектом.



Ядро ОС. Подсистема защиты (продолжение)

Примеры SID:

·S-1-1-0 – группа «все пользователи»;

·S-1-5-21-1078081533-1364589140-839522115-1003 – типичный «Администратор».

```
HANDLE CreateFile(  
    LPCTSTR lpFileName,  
    DWORD dwDesiredAccess,  
    DWORD dwShareMode,  
    LPSECURITY_ATTRIBUTES lpSecurityAttributes,  
    DWORD dwCreationDistribution,  
    DWORD dwFlagsAndAttributes,  
    HANDLE hTemplateFile  
);
```

Атрибуты безопасности

nLength
lpSecurityDescriptor
bolInheritHandle

Описатель безопасности объекта

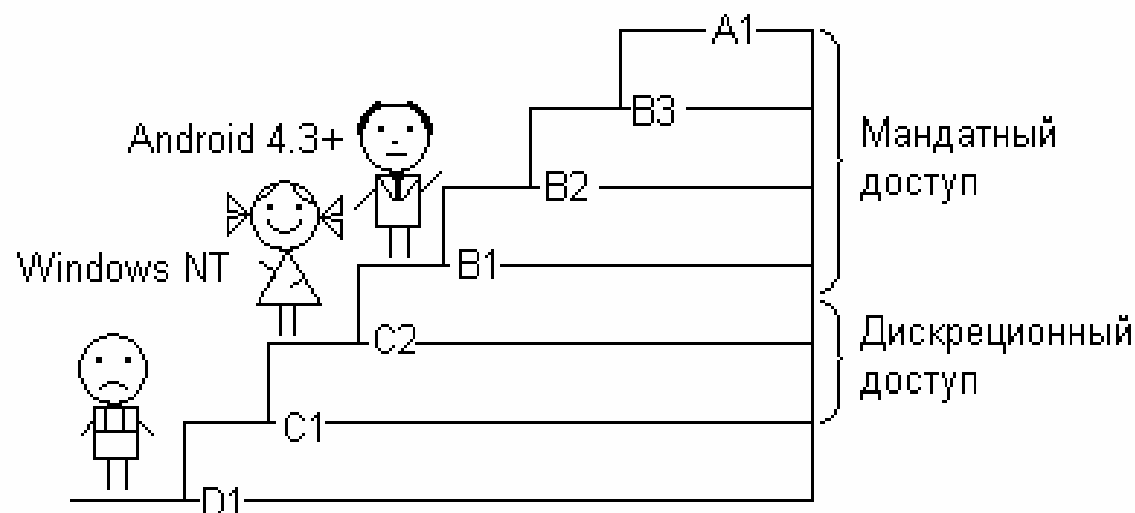
SID владельца			
▶ SID группы			
SID1	Метод	Разр./запр.	DACL
SID2	Метод	Разр./запр.	
SIDN	Метод	Разр./запр.	SACL
SID1	Аудит: да/нет		
SID2	Аудит: да/нет		
SIDN	Аудит: да/нет		

Ядро ОС. Подсистема защиты (продолжение)

«Оранжевая книга»



- Критерии оценки безопасности компьютерных систем (USA);
- Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности (РФ);
- Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации (РФ);
- Критерии безопасности информационных технологий (EU).



Ядро ОС. Подсистема защиты (продолжение)

2. Криптография – дисциплина, изучающая методы обеспечения конфиденциальности и аутентичности информации.

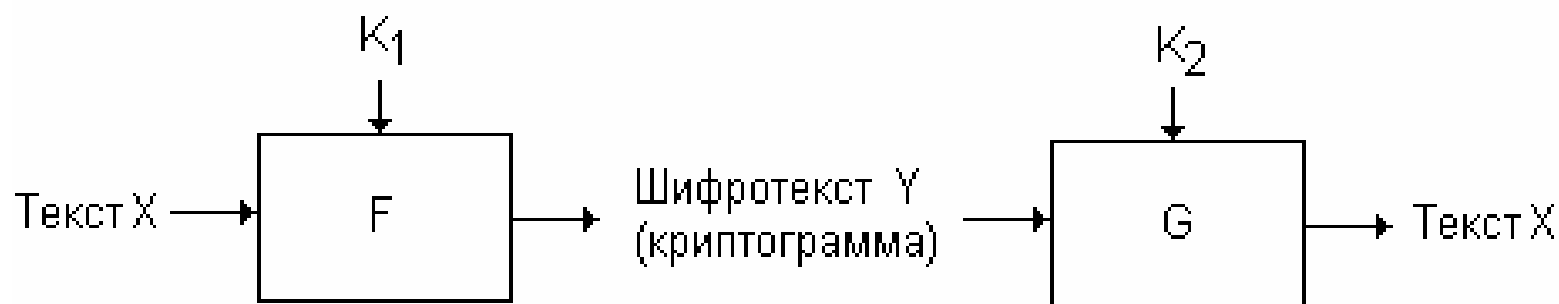
$Y = F(X, K_1)$ – шифрование данных;

$X = G(Y, K_2)$ – расшифрование данных;

X – «текст», Y – «шифротекст» или «криптограмма»;

K_1 и K_2 – ключи;

F и G – алгоритмы.



Если $K_1 = K_2$, то шифр **симметричный**; иначе – **асимметричный**

Принцип Кирхгофа (Керкхоффа, Керкхоффена) = стойкость шифра зависит только от секретности ключа.

Ядро ОС. Подсистема защиты (продолжение)

Исторические примеры шифровальных ключей

Считала



Маршрутный шифр

К	Р	И	П	К	Т	Д	К	Т	Д
Т	О	Г	Р	Р	О	Г	О	Г	Р
А	Ф	И	Я	И	П	Я	П	Р	Я

Шифр Вижинера

⊕ СОБАКА
ГАВГАВ
ХПДДЛГ

Простая моноалфавитная замена



Поворотная решетка Кардано

а	л										
е									ю	с	
		к									
						т	а		с		
		с									
		у									
						р	а	в			
		п									
			о			л	я				
		з	д								

АЛЕКС ЮСТАСУ
ПОЗДРАВЛЯЕМ
С ДНЕМ РОЖДЕ
НЬЯ



у	а	ю	л	д	з
т	е	у	у	ю	с
р	д	а	к	в	т
о	т	п	а	о	с
л	я	ю	с	о	ю
а	к	з	в	д	а

Ядро ОС. Подсистема защиты (продолжение)

Стойкость шифров ко взлому (криптостойкость)

Шифры типа «сложение по модулю мощности алфавита»:

- буквы текста ('К'+ 'Я') $\text{mod } 33 = \text{'Л'}$ – **шифр Вижинера**;
- биты данных $(1 + 1) \text{ mod } 2 = 0$ – **шифр Вернама**.

⊕ криптография
йцукенгшщзхъ
щфжътьчьёищью

⊕ 0000111101011010
1001001101100010
1001110000111000

Условия абсолютной стойкости:

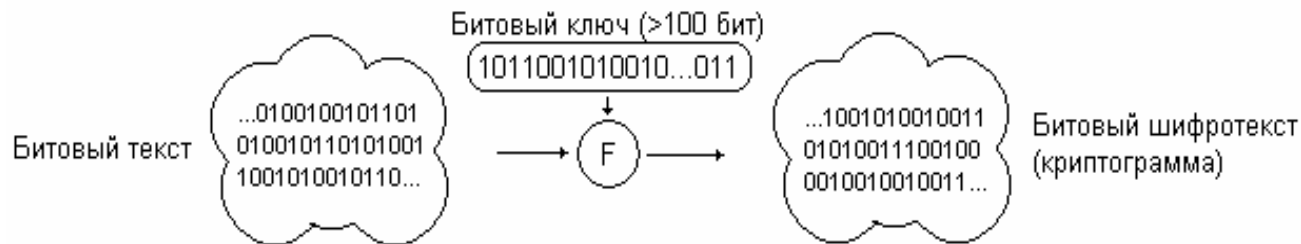
- Длина ключа = длине сообщения;
- Элементы ключа случайны;
- Ключ однократен.

Такой шифр = **одноразовый шифроблокнот** (кодовая книга).

Страница 6											
Страница 5											
Страница 4											
Ж	К	Л	Т	О	П	И	С	Ь	В	Ц	А
Ф	Х	М	Н	Д	В	Э	З	О	П	С	Т
Х	Ь	В	Ц	Ю	Ф	В	Э	А	О	П	С
С	М	В	Г	Л	Ж	В	М	Ч	С	Т	Г
Т	Р	У	Х	Ч	И	Ф	В	Ч	С	Т	Г
Ж	Э	Ь	В	У	К	Н	Ч	М	Т	Г	И
Ч	С	И	В	Э	Ъ	Й	М	Т	Г	И	У
Р	А	Ф	Д	П	Н	Г	Г	И	У	И	Й
Н	П	О	З	У	И	Й					

Ядро ОС. Подсистема защиты (продолжение)

Современные (компьютерные) шифры



Криптосистема = шифр + правила использования:

- ключевое расписание (правила генерации ключей);
- предобработка текста (имитовставка, salting...);
- постобработка шифротекста (сцепление блоков...).

#	Наименование	Страна, город	Выч. ядер	Производит., TF
1	RoadRunner	США	129000	1100
2	Jaguar	США	150000	1000
	Ломоносов	РФ, Москва, МГУ		0900
3	JUGENE	ФРГ	294000	0825
4	Playades/NASA	США	51000	0487
...				
500	Гос. выч. сист	ФРГ	600	0017
...				
	Сергей Королев	РФ, Самара, СГАУ	1712 (+4216 граф)	0010

Пусть ядро подбирает 1000000000 кл/сек $\approx 2^{28}$ кл/сек.

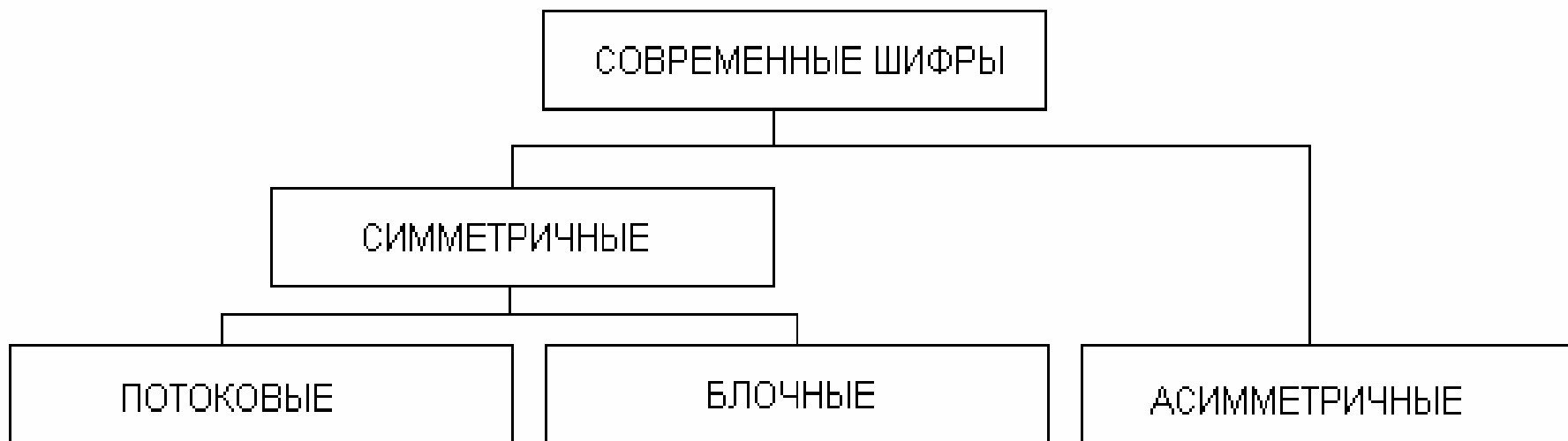
Тогда суперкомпьютер подберет 1290000000000000 $\approx 2^{43}$ кл/сек.

В году 31 536 000 сек $\approx 2^{25}$ сек., тогда подберет $2^{43+25} \approx 2^{67}$ кл/год.

Население Земли 7 млрд $\approx 2^{33}$ чел, тогда все вместе подберут 2^{100} кл/г.

По закону Мура 2^{128} научится подбирать через $28 \times 1.5 = 44$ г.

Ядро ОС. Подсистема защиты (продолжение)



Симметричные: $K_1 = K_2$. Общий недостаток – сложность распределения ключей.

Асимметричные: $K_1 \neq K_2$. Общие недостатки – невысокая криптостойкость, малая скорость работы.

Потоковые: шифруют поэлементно (посимвольно или побитно).

Блочные: шифруют блоками элементов.

Ядро ОС. Подсистема защиты (продолжение)

Симметричные потоковые шифры



Шифр	Длина ключа	Применение
RC4	8, 16, 24...512	Протокол SSL
A5	64	GSM
SEAL	160	

```
// RC4
unsigned char S[ 256 ];
unsigned int i, j;

/* ключевое расписание */
void rc4_init( unsigned char* key, unsigned int key_length ) {
    unsigned char temp;
    for( i = 0; i != 256; ++i ) S[ i ] = i;
    for( i = j = 0; i != 256; ++i ) {
        j = ( j + key[ i % key_length ] + S[ i ] ) % 256;
        temp = S[ i ];
        S[ i ] = S[ j ];
        S[ j ] = temp;
    }
    i = j = 0;
}
```

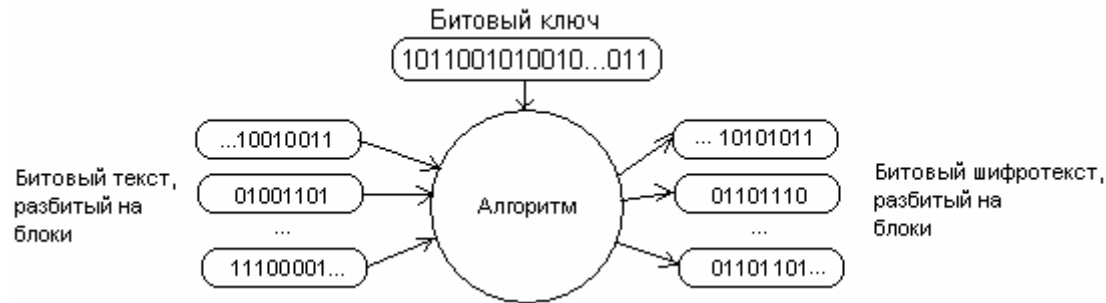
```
/* Вывод одного псевдослучайного байта гаммы */
unsigned char rc4_output() {
    unsigned char temp;
    i = ( i + 1 ) % 256;
    j = ( j + S[ i ] ) % 256;
    temp = S[ j ];
    S[ j ] = S[ i ];
    S[ i ] = temp;
    return S[ ( temp + S[ j ] ) % 256 ];
}
```

Достоинства: простота, высокое быстродействие, легкость программной и аппаратной реализации (на LFSR).

Назначение: применение в связи (с одноразовыми «сеансовыми» ключами).

Ядро ОС. Подсистема защиты (продолжение)

Симметричные блочные шифры



Название	Размер блока	Размер ключа	Примечание
Люцифер	128	128	Демонстрационный шифр Х. Фейстела
DES	64	56	Старый национальный стандарт США
3DES	64	3×56=168	Тройной DES: $Y = \text{DES}_{K1}(\text{DES}_{K2}(\text{DES}_{K3}(X)))$
DESX	64	3×56=168	DES и XOR: $Y = K1 \oplus \text{DES}_{K2}(X \oplus K3)$
AES	128, 192, 256	128, 192, 256	Новый национальный стандарт США
ГОСТ – 89	64	256	Российский стандарт шифрования
IDEA	128	128	Не запатентован
CAST	64, 128	128, 256	Не запатентован
Blowfish	128	64-448	Не запатентован, опубликован у Б. Шнейера
TEA	64	128	Не запатентован, очень компактен

// TEA

```
/* Шифрование блока */
void encrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0, i;
    uint32_t delta=0x9e3779b9;
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];
    for (i=0; i < 32; i++) {
        sum += delta;
        v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
    }
    v[0]=v0; v[1]=v1;
}
```

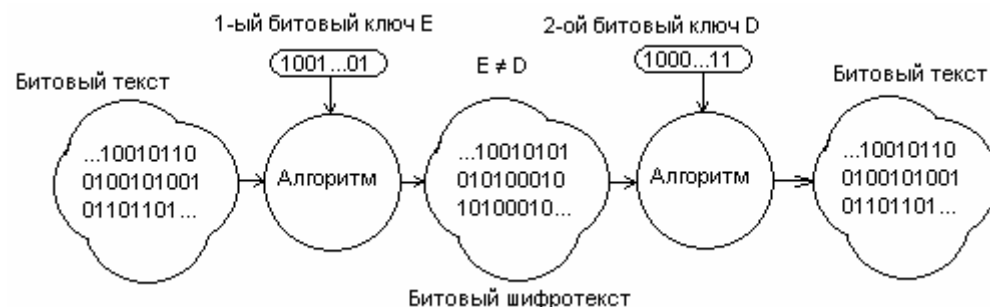
```
/* Расшифрование блока */
void decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0xC6EF3720, i;
    uint32_t delta=0x9e3779b9;
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];
    for (i=0; i<32; i++) {
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta;
    }
    v[0]=v0; v[1]=v1;
}
```

Достоинства: высокая криптостойкость

Применение: для шифрования хранимой информации (в т.ч. с многократно использованными ключами), в цифровой связи.

Ядро ОС. Подсистема защиты (продолжение)

Асимметричные шифры и ЭЦП



Шифр/метод	Длина ключа	Мат. проблема	Применение
RSA	>1024	Факторизация целых чисел	Шифрование с о/к, ЭЦП
DHE/DHS (схемы Эль Гамала)	-«-	Дискретное логарифмирование в целых числах	Протокол распределения ключей Диффи-Хеллмана, шифрование с о/к, ЭЦП (ГОСТ-1994)
(Схемы Коблица)	>384 ???	Факторизация или дискретное логарифмирование на эллиптических кривых	Шифрование с о/к, ЭЦП (ГОСТ-2001)

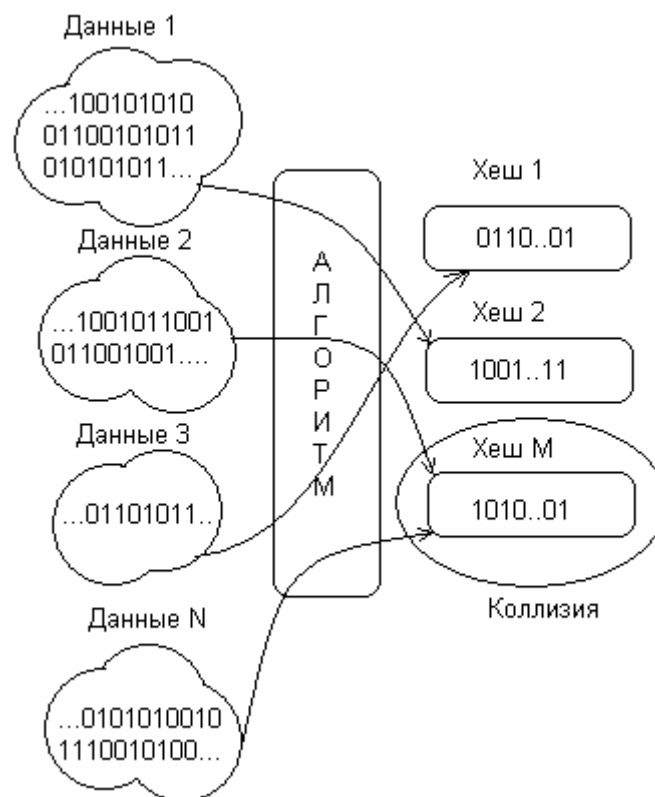
Достоинства: уникальные возможности.

Недостатки: низкая криптостойкость, невысокая скорость работы.

Применение: шифрование с открытым ключом, ЭЦП, распределение секретных ключей по открытому каналу.

Ядро ОС. Подсистема защиты (продолжение)

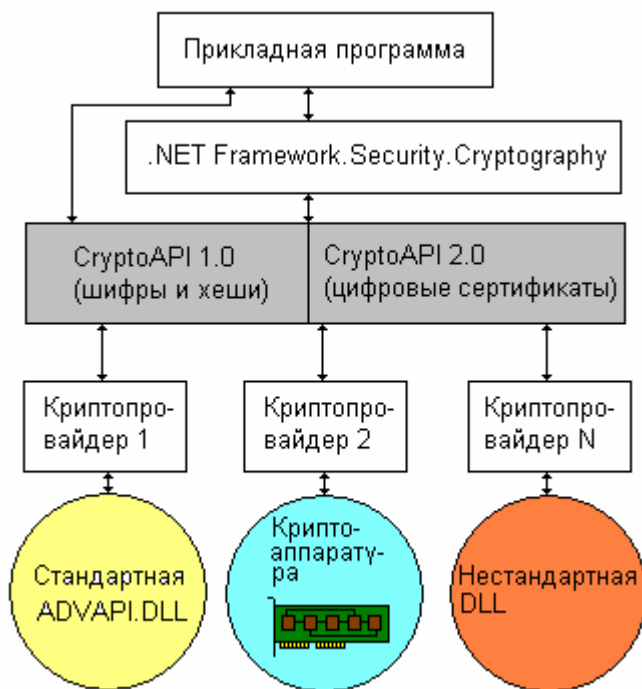
Хеш-функции и контрольные суммы



Алгоритм	Длина результата	Применение	Примечание
CRC-16,-32	16, 32	В технике	Нестоек
Adler-32	32	- « -	Нестоек
MD5	128	В криптографии	Скомпрометирован
SHA-1	160	- « -	
SHA-2	256, 512	- « -	

Ядро ОС. Подсистема защиты (окончание)

Microsoft CryptoAPI



Состав по умолчанию:

- RC4 (до Vista)
- RC2, DES, 2DES, 3DES, DESX, AES (с XP)
- RSA
- MD5, SHA1, SHA2.

```
// CryptoAPI
#include <windows.h>
#include <stdio.h>
#define _WIN32_WINNT 0x0400
#include "wincrypt.h"

char *szPassword = "PASSWORD"; // Пароль из которого генерируем ключ

TudaSuda( char* datain, long lendatain) {
    HCRYPTPROV hCryptProv;           // Хэндл криптопровайдера
    HCRYPTHASH hCryptHash;           // Хеш-объект для алгоритма MD5
    DWORD cryptBlockSize;           // Длина данных
    DWORD bytesback;                 // Длина новых данных
    HCRYPTKEY hCryptKey;              // Ключ для шифрования
    CryptAcquireContext(&hCryptProv, NULL, NULL, PROV_RSA_FULL, 0);
    CryptCreateHash(hCryptProv, CALG_MD5, 0, 0, &hCryptHash);
    CryptHashData(hCryptHash, (BYTE*)szPassword, strlen(szPassword), 0);
    CryptDeriveKey(hCryptProv, CALG_SEAL, hCryptHash, 0, &hCryptKey);
    cryptBlockSize=lendatain; bytesback=lendatain;
    CryptEncrypt(hCryptKey, NULL, TRUE, 0, (BYTE *)datain, &cryptBlockSize, 0);
    BYTE* bData = new BYTE[cryptBlockSize];
    memcpy(bData, datain, lendatain);
    CryptEncrypt(hCryptKey, NULL, TRUE, 0, bData, &bytesback, cryptBlockSize);
    CryptDestroyKey(hCryptKey);
    CryptDestroyHash(hCryptHash);
    CryptReleaseContext(hCryptProv, 0);
    delete[] bData;
}

int main() {
    TudaSuda("SSAU forever!", strlen("SSAU forever!"));
}
```

Загрузчик программ

Загрузчик программ – часть диспетчера памяти. Назначение:

- создание нового адресного пространства;
- разбиение адресного пространства на составные части;
- размещение в памяти программы и вспомогательных компонентов;
- запуск программы .

