Лекция 1

Введение

- Цели и задачи курса.
- Структура и назначение современной операционной системы.

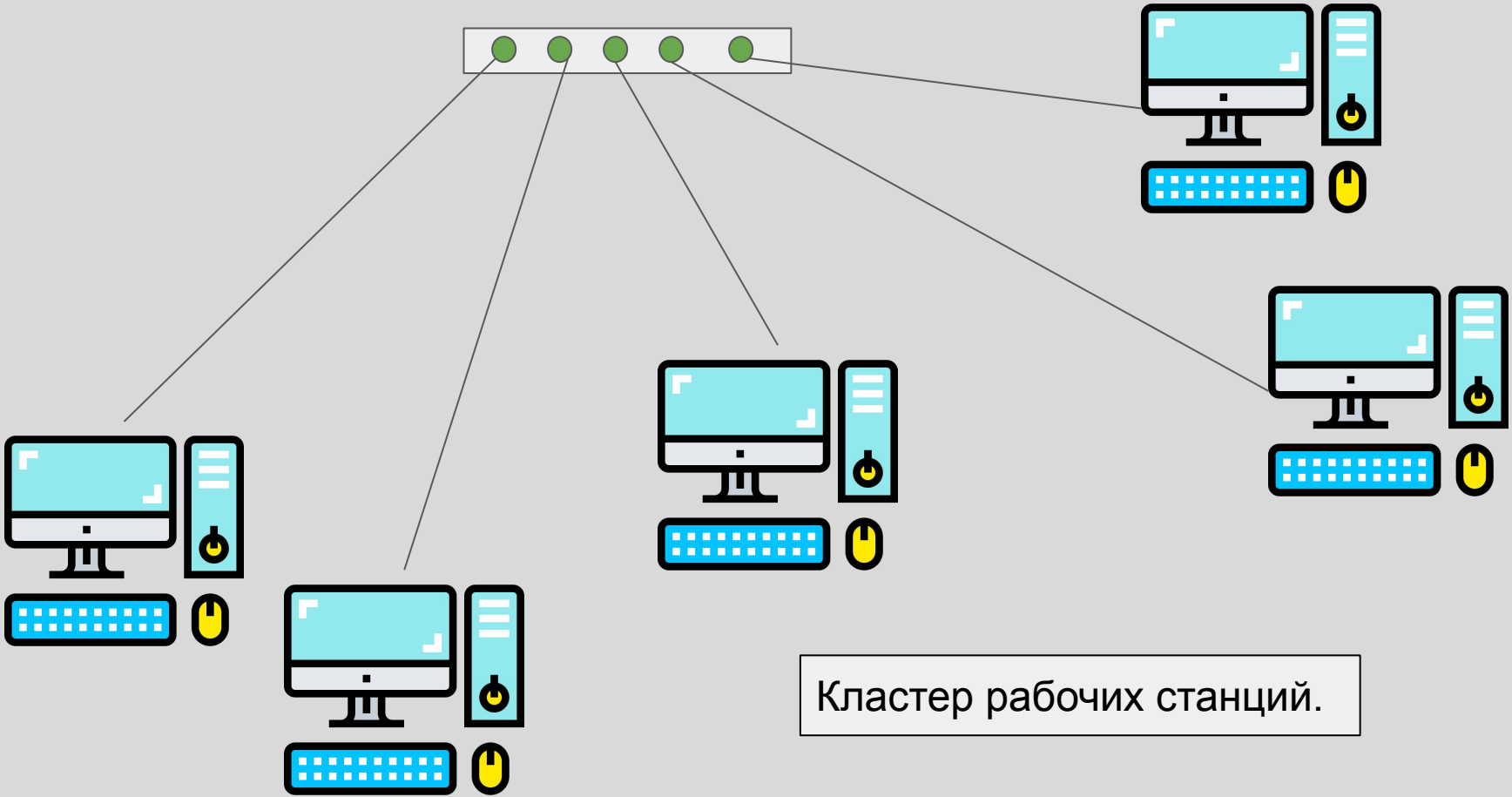
Задачи современной операционной системы

- Серверы.
- Мейнфреймы.
- Вычислительные кластеры.
- Суперкомпьютеры.
- Кластеры рабочих станций.
- Терминалы.

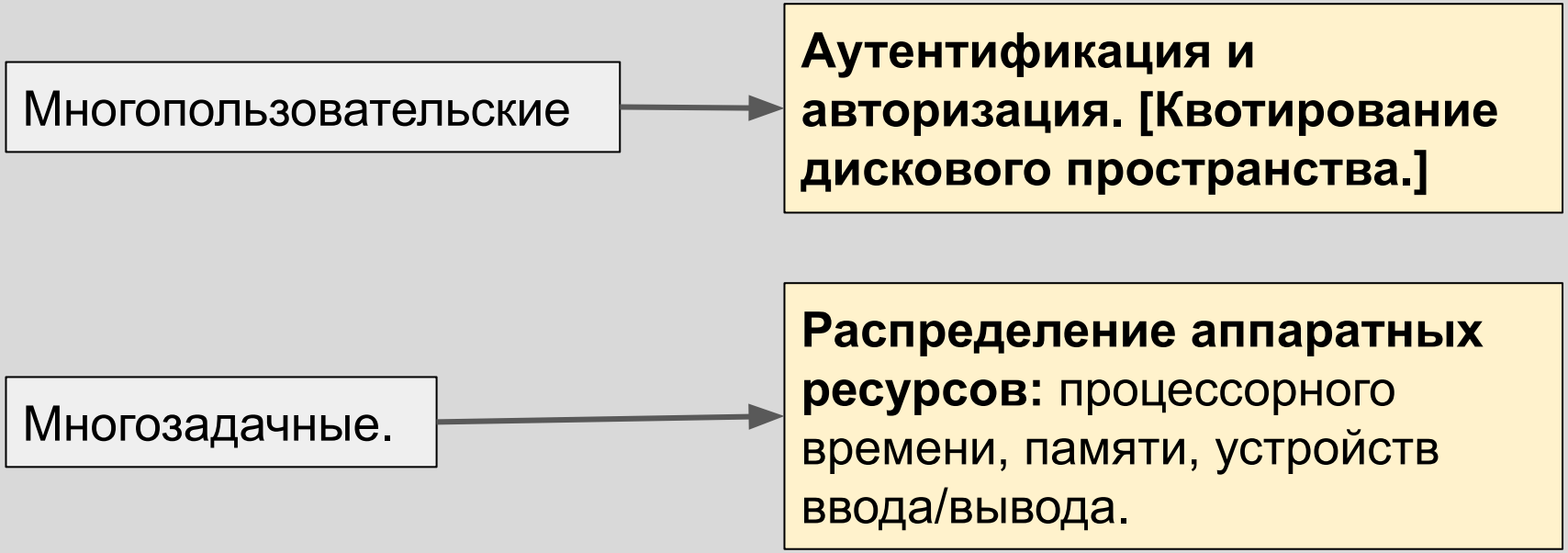
- Многопользовательская ОС.
- Многозадачная ОС (*с раделением времени*).

Мейнфреймы, кластеры серверов, суперкомпьютеры.

Терминалы, рабочие станции, планшеты, … 😜 смартфоны.

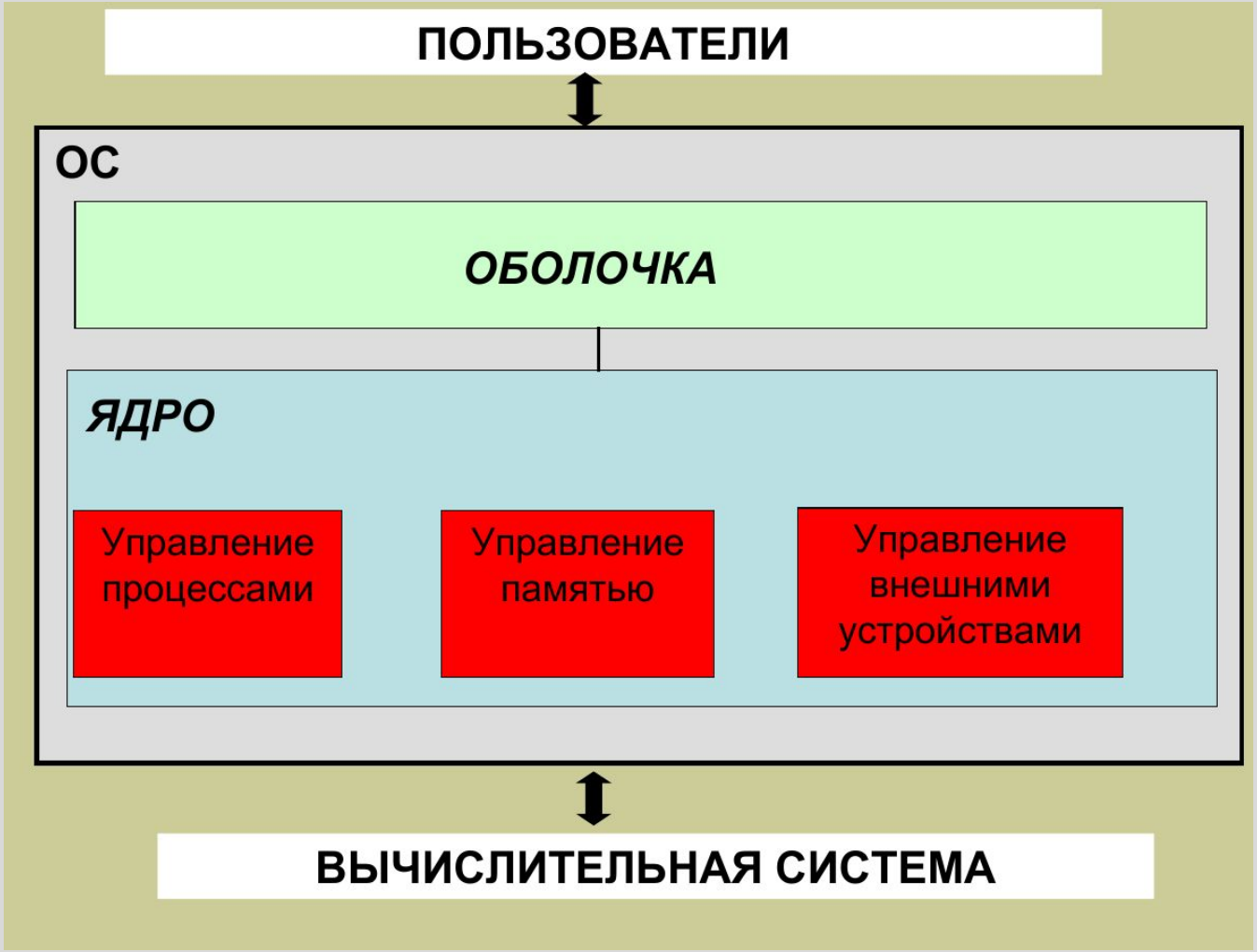


Кластер рабочих станций.


Многопользовательские
Аутентификация и авторизация. [Квотирование дискового пространства.]
Многозадачные.
Распределение аппаратных ресурсов: процессорного времени, памяти, устройств ввода/вывода.

Структура современной операционной системы

- **Оболочка** - набор программ, реализующих *интерфейс “пользователь - операционная система”.*
- **Ядро** - набор команд, реализующих *интерфейс “операционная система - аппаратные средства”.*


ПОЛЬЗОВАТЕЛИ
ОС
ОБОЛОЧКА
ЯДРО
Управление процессами
Управление памятью
Управление внешними устройствами
ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА

Взаимодействие прикладных программ и ОС

Режим ядра (режим супервизора, привилегированный режим):

- полный доступ к командам процессора;
- обработка прерываний и исключений;
- доступ к объектам ядра.

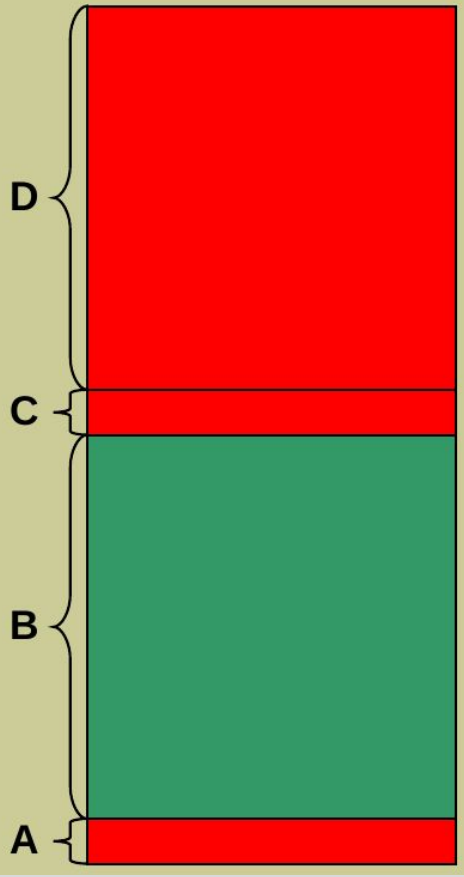
Пользовательский режим:

- ограниченный набор команд процессора;
- запрет на вызов обработчиков прерываний.

Интерфейс системных вызовов предоставляет контролируемый доступ прикладных программ к ресурсам компьютера посредством переход из пользовательского режима в режим ядра.

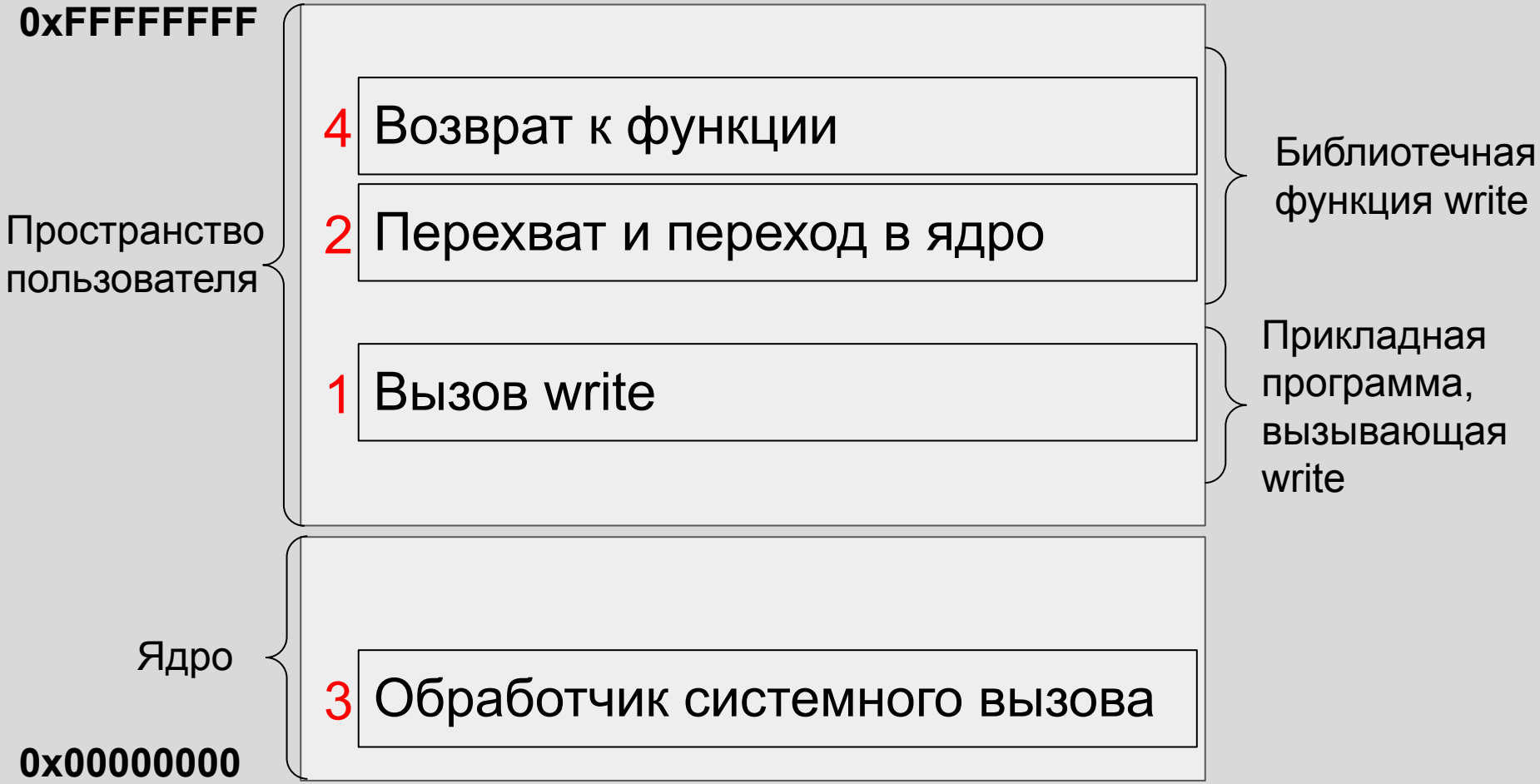
Интерфейс прикладного программирования - библиотечные функции.

Пример структуры адресного пространства 32-разрядной ОС.



A. 0x00000000 – 0x0000FFFF; используется для неинициализированных указателей; ***недоступно*** в пользовательском режиме.
B. 0x00010000 – 0x7FFEFFFF; адресное пространство процессов, содержит прикладные модули .exe и .dll, win32 (kernel32.dll, user32.dll и т.д.), файлы, отображаемые в память; ***доступно*** в пользовательском режиме.
C. 0x7FFF0000 – 0x7FFFFFFF; используется для некорректно инициализированных указателей; ***недоступно*** в пользовательском режиме.
D. 0x80000000 – 0xFFFFFFFF; зарезервировано ОС Windows для исполнительной системы, ядра и драйверов устройств; ***недоступно*** в пользовательском режиме.

write(hFile, pBuffer, nToWrite) – библиотечная функция



write(int fd, const void *buf, size_t count);
- библиотечная функция

```
mov  edx, 1       ;сколько байт записать
mov  ecx, hex     ;буфер, откуда писать
mov  ebx, 1       ;куда записывать, 1 - stdout
mov  eax, 4       ;номер системного вызова
int  80h          ;шлюз к ядру
```

Таблица системных вызовов

%eax	Name	Source	%ebx	%ecx	%edx
1	sys_exit	kernel/exit.c	int	-	-
2	sys_fork	arch/i386/kernel/process.c	struct pt_regs	-	-
3	sys_read	fs/read_write.c	unsigned int	char*	size_t
4	sys_write	fs/read_write.c	unsigned int	const char*	size_t
5	sys_open	fs/open.c	const char*	int	int
6	sys_close	fs/open.c	unsigned int	-	-

Базовые подсистемы ОС

Файловая подсистема.

Операционная система обеспечивает единообразное представление хранения информации в виде *логических единиц хранения - файлов*, абстрагируясь от физических устройств хранения. Файловая подсистема предоставляет доступ к данным на физических устройствах манипулируя файлами. К её функциям относится создание и удаление файлов, запись в файлы и чтение из них, а также изменение прав доступа к файлам. Следует различать *файловую систему* и файловую подсистему. Файловая система определяет способ хранения данных - формат, структуру и способ доступа к ним на физических устройствах. Файловая подсистема, являясь подсистемой ядра оперирует данными управляя объектами ядра - файлами. Можно сказать, что файловая система - это иерархическая база данных, а файловая подсистема - это интерфейс доступа к базам данных.

Подсистема управления процессами.

Процесс - это исполняемый экземпляр программы/приложения. Программа представляет собой набор файлов, хранящихся на внешнем носителе. Хотя бы один файл должен быть исполняемым - специального формата, соответствующего операционной системе (например, ELF файлы в Linux или PE файлы в MS Windows), и содержащий последовательности команд в машинном коде. При запуске программы создаётся *процесс*, ему присваивается идентификатор - уникальное целое число (имя процесса в системе), назначается адресное пространство, на которое (точнее, на подпространство пользователя) отображаются файлы программы и файлы необходимых программе библиотек. С “точки зрения“ процесса он выполняется один и ему принадлежат все ресурсы вычислительной системы. Программы управления процессами - *планировщик и диспетчер процессов*, а также *менеджер виртуальной памяти* обеспечивают такое представление, разделяя аппаратные ресурсы между процессами.

Ресурсы процесса:

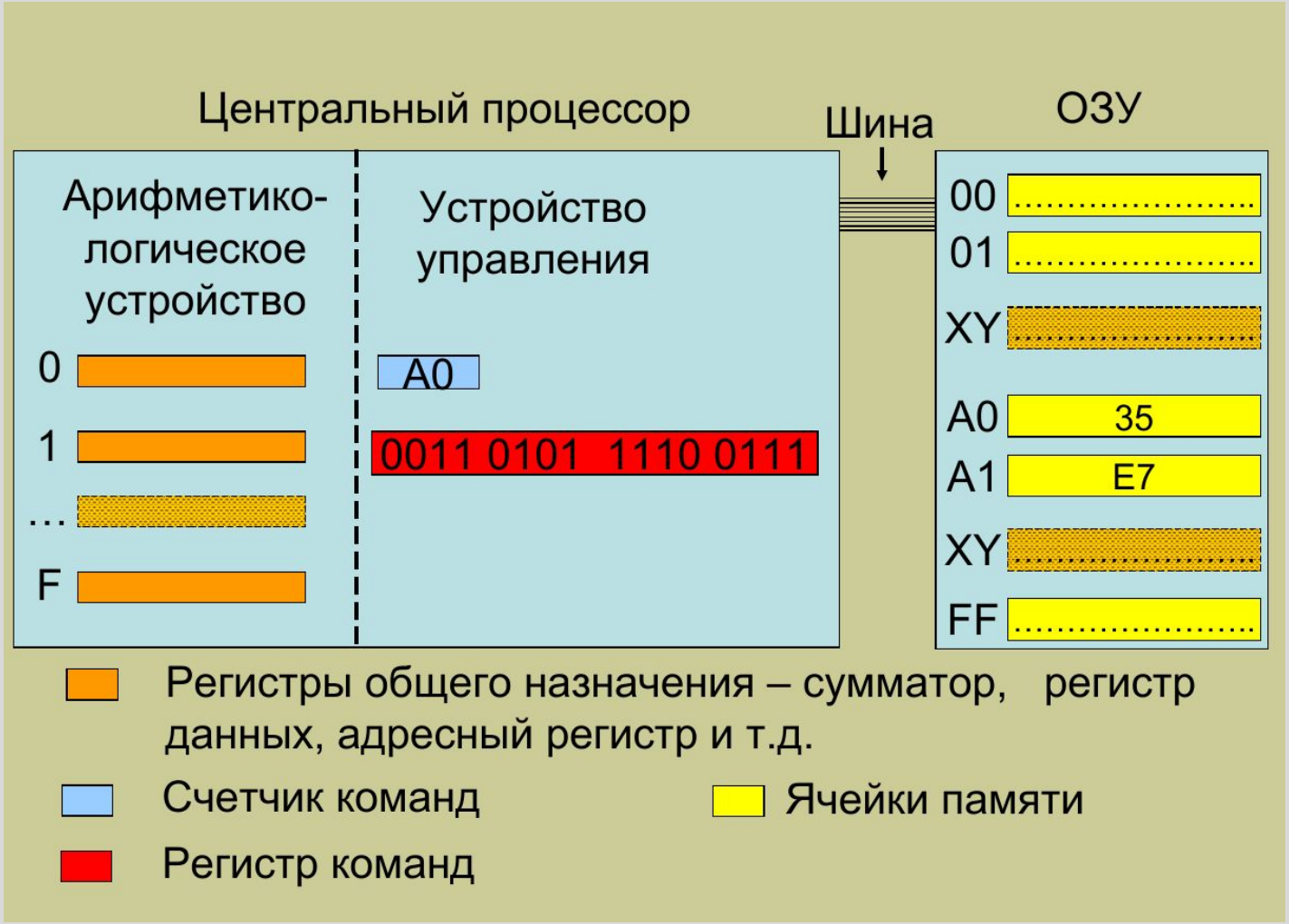
• виртуальное адресное пространство;

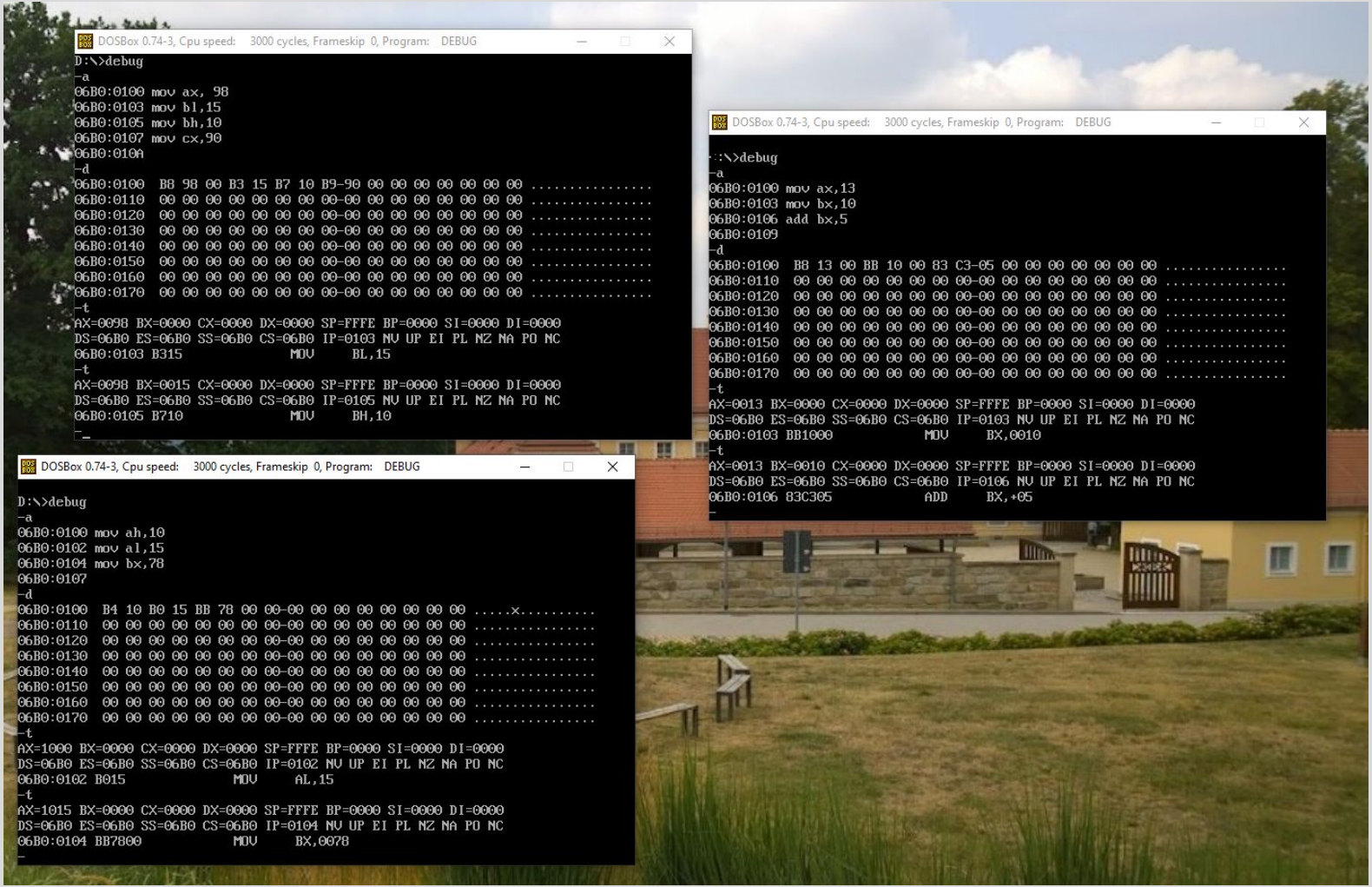
• системные ресурсы –области физической памяти, процессорное время, файлы, растровые изображения и т.д.;

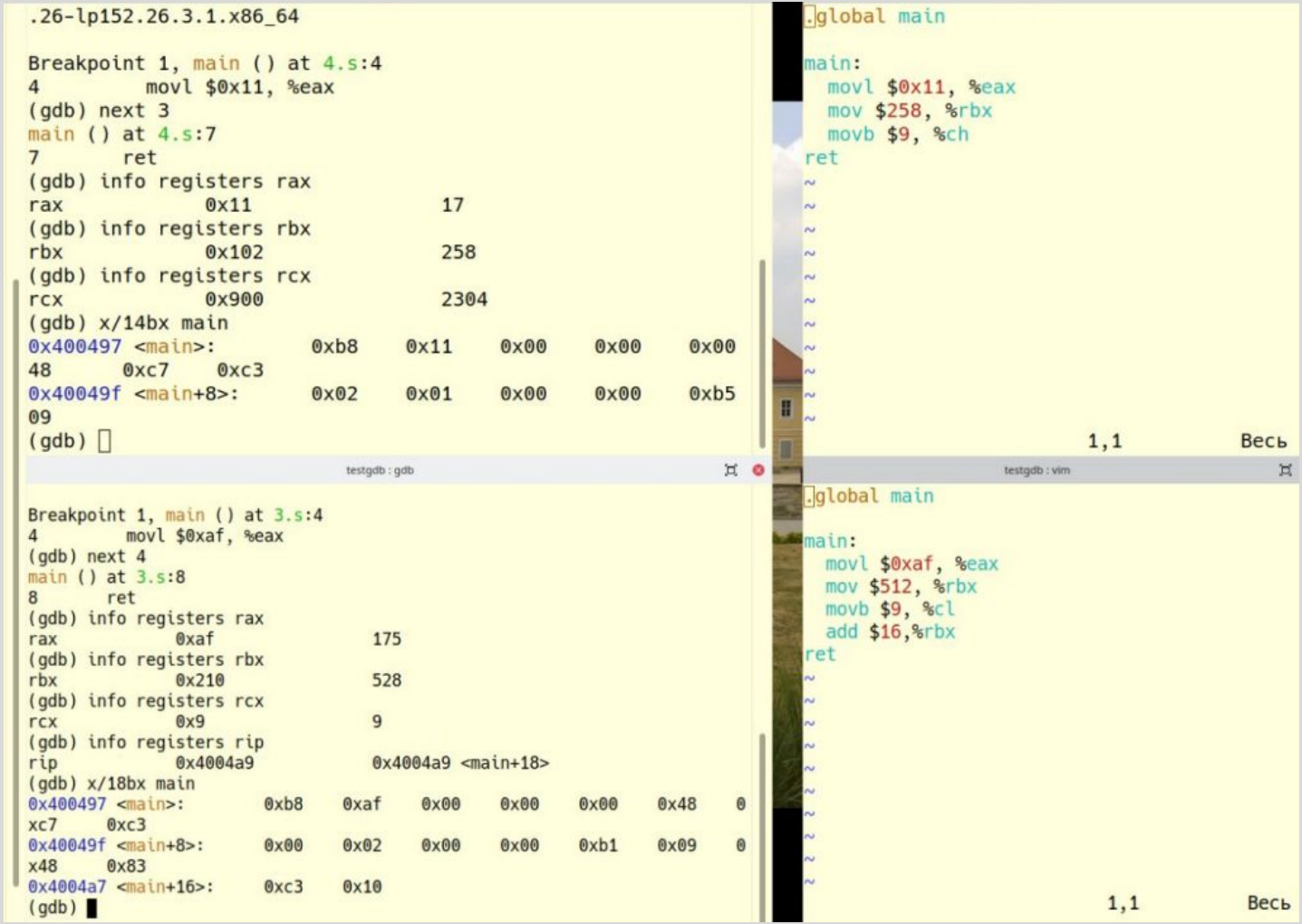
• модули процесса, то есть исполняемые модули, загруженные (отображенные) в его адресное пространство – основной загрузочный модуль, библиотеки динамической компоновки и т.д.;

• уникальный идентификационный номер, называемый идентификатором процесса;

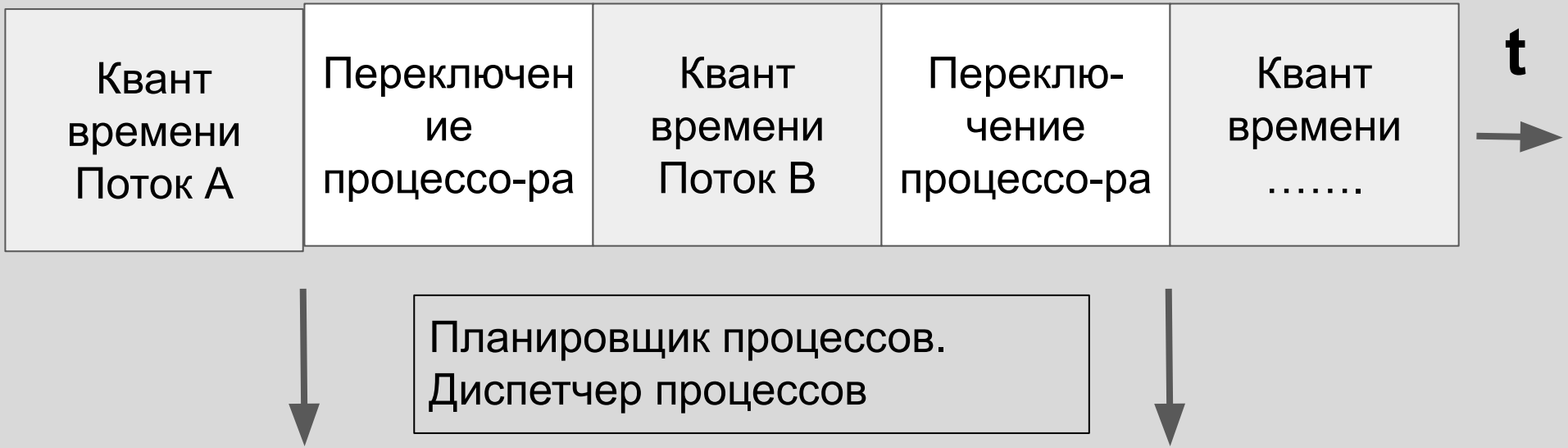
• потоки (по крайней мере, один поток) - исполняемые последовательности команд.


Центральный процессор
Шина
ОЗУ
Арифметико-
логическое
устройство
Устройство
управления
0
1
…
F
A0
0011 0101 1110 0111
00
01
XY
A0
35
A1
E7
XY
FF
Регистры общего назначения – сумматор, регистр данных, адресный регистр и т.д.
Счетчик команд
Ячейки памяти
Регистр команд


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
D:\>debug
-a
06B0:0100 mov ax, 98
06B0:0103 mov bl,15
06B0:0105 mov bh,10
06B0:0107 mov cx,90
06B0:010A
-d
06B0:0100 B8 98 00 B3 15 B7 10 B9-90 00 00 00 00 00 00 00 ....
06B0:0110 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0120 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0130 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0140 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0170 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
-t
AX=0098 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=06B0 ES=06B0 SS=06B0 CS=06B0 IP=0103 NV UP EI PL NZ NA PO NC
06B0:0103 B315 MOV BL,15
-t
AX=0098 BX=0015 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=06B0 ES=06B0 SS=06B0 CS=06B0 IP=0105 NV UP EI PL NZ NA PO NC
06B0:0105 B710 MOV BH,10
-
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
D:\>debug
-a
06B0:0100 mov ah,10
06B0:0102 mov al,15
06B0:0104 mov bx,78
06B0:0107
-d
06B0:0100 B4 10 B0 15 BB 78 00 00-00 00 00 00 00 00 00 00 .....x..........
06B0:0110 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0120 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0130 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0140 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0170 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
-t
AX=1000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=06B0 ES=06B0 SS=06B0 CS=06B0 IP=0102 NV UP EI PL NZ NA PO NC
06B0:0102 B015 MOV AL,15
-t
AX=1015 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=06B0 ES=06B0 SS=06B0 CS=06B0 IP=0104 NV UP EI PL NZ NA PO NC
06B0:0104 BB7800 MOV BX,0078
-
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
-:\>debug
-a
06B0:0100 mov ax,13
06B0:0103 mov bx,10
06B0:0106 add bx,5
06B0:0109
-d
06B0:0100 B8 13 00 BB 10 00 83 C3-05 00 00 00 00 00 00 00 ....
06B0:0110 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0120 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0130 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0140 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
06B0:0170 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
-t
AX=0013 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=06B0 ES=06B0 SS=06B0 CS=06B0 IP=0103 NV UP EI PL NZ NA PO NC
06B0:0103 BB1000 MOV BX,0010
-t
AX=0013 BX=0010 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=06B0 ES=06B0 SS=06B0 CS=06B0 IP=0106 NV UP EI PL NZ NA PO NC
06B0:0106 83C305 ADD BX,+05
-


.26-lp152.26.3.1.x86_64
Breakpoint 1, main () at 4.s:4
4 movl $0x11, %eax
(gdb) next 3
main () at 4.s:7
7 ret
(gdb) info registers rax
rax 0x11 17
(gdb) info registers rbx
rbx 0x102 258
(gdb) info registers rcx
rcx 0x900 2304
(gdb) x/14bx main
0x400497 <main>: 0xb8 0x11 0x00 0x00 0x00
48 0xc7 0xc3
0x40049f <main+8>: 0x02 0x01 0x00 0x00 0xb5
09
(gdb)
testgdb : gdb
.global main
main:
movl $0x11, %eax
mov $258, %rbx
movb $9, %ch
ret
1,1 Весь
testgdb : vim
Breakpoint 1, main () at 3.s:4
4 movl $0xaf, %eax
(gdb) next 4
main () at 3.s:8
8 ret
(gdb) info registers rax
rax 0xaf 175
(gdb) info registers rbx
rbx 0x210 528
(gdb) info registers rcx
rcx 0x9 9
(gdb) info registers rip
rip 0x4004a9 0x4004a9 <main+18>
(gdb) x/18bx main
0x400497 <main>: 0xb8 0xaf 0x00 0x00 0x00 0x48 0
xc7 0xc3
0x40049f <main+8>: 0x00 0x02 0x00 0x00 0xb1 0x09 0
x48 0x83
0x4004a7 <main+16>: 0xc3 0x10
(gdb)
.global main
main:
movl $0xaf, %eax
mov $512, %rbx
movb $9, %cl
add $16,%rbx
ret
1,1 Весь

Модель процесса:

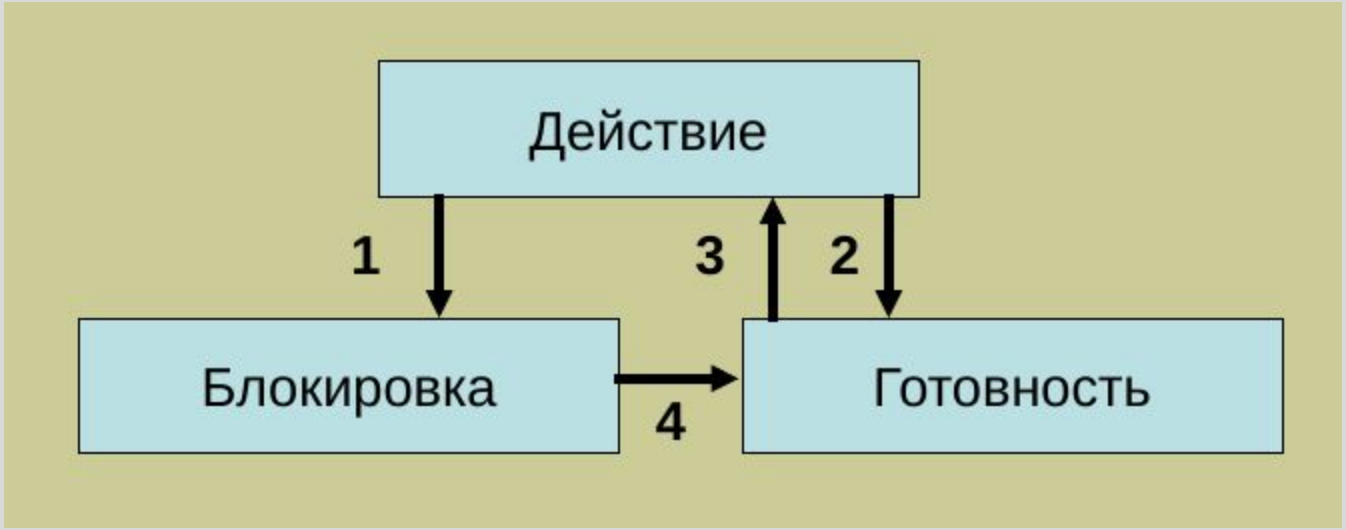


- Сохранение контекста текущего потока, регистров, стека и областей памяти.
- Определение очередного потока.
- Восстановление контекста очередного потока.

Последовательность исполнения потоков в среде с вытесняющей многозадачностью:

В системе определен **квант времени** (порядка десятков миллисекунд) – процессорное время выделяемое одному потоку (каждому - своё). **Длительность выполнения одного потока не может превышать одного кванта.** Когда это время заканчивается, диспетчер процессов переключает процессор на выполнение другого потока. При этом состояние регистров, стека и областей памяти – *контекст потока*, сохраняется в стеке потока. Очередность потоков определяется их ***состоянием и приоритетом***.

Состояние процессов:



1. Процесс заблокирован в ожидании ввода.
2. Диспетчер выбирает другой процесс.
3. Диспетчер выбирает данный процесс.
4. Входные данные стали доступны.

Реализацией процессов является **таблица процессов**, программно реализованная, как список структур) (***P**rocess **C**ontrol **B**lock*).

Информация о процессах хранится в таблице процессов и обновляется планировщиком процессов.

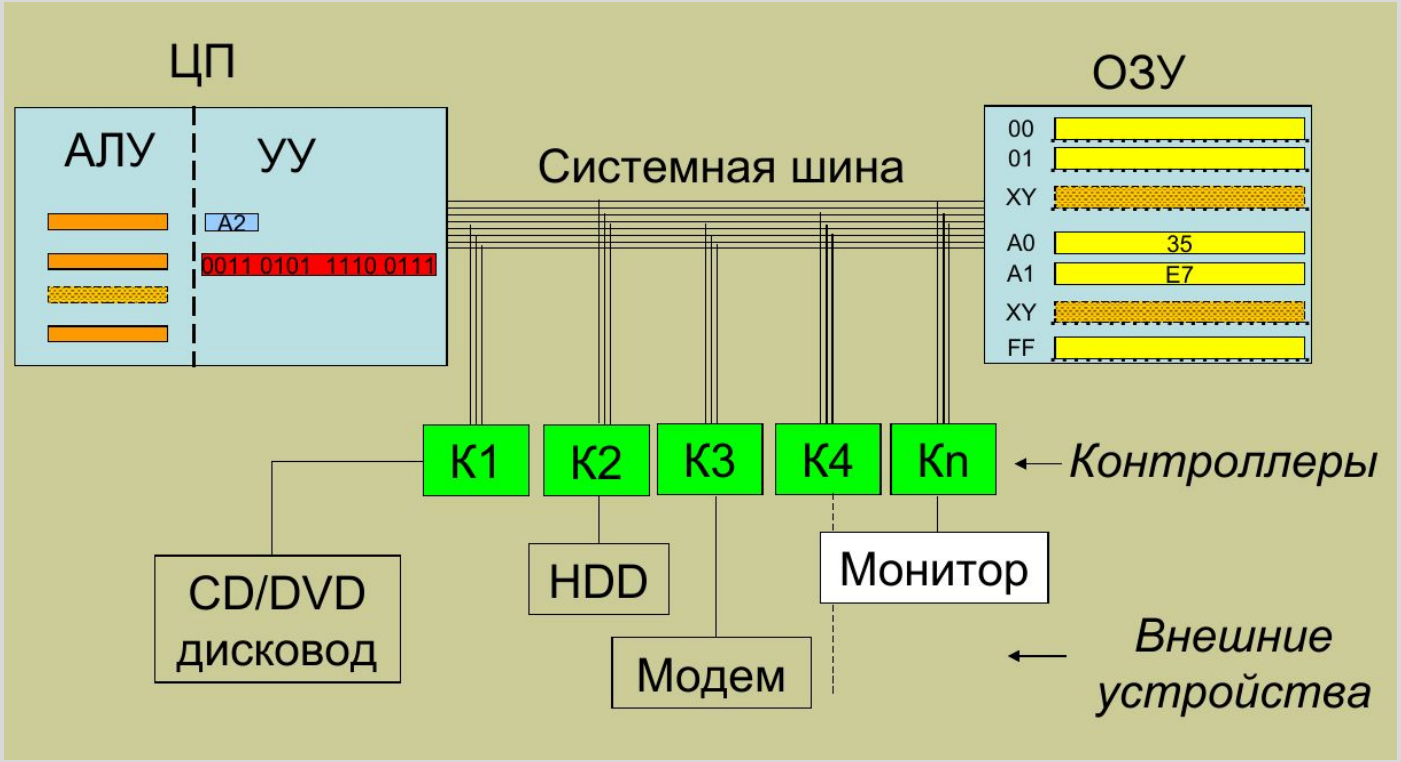
Некоторые поля типичной записи таблицы процессов:

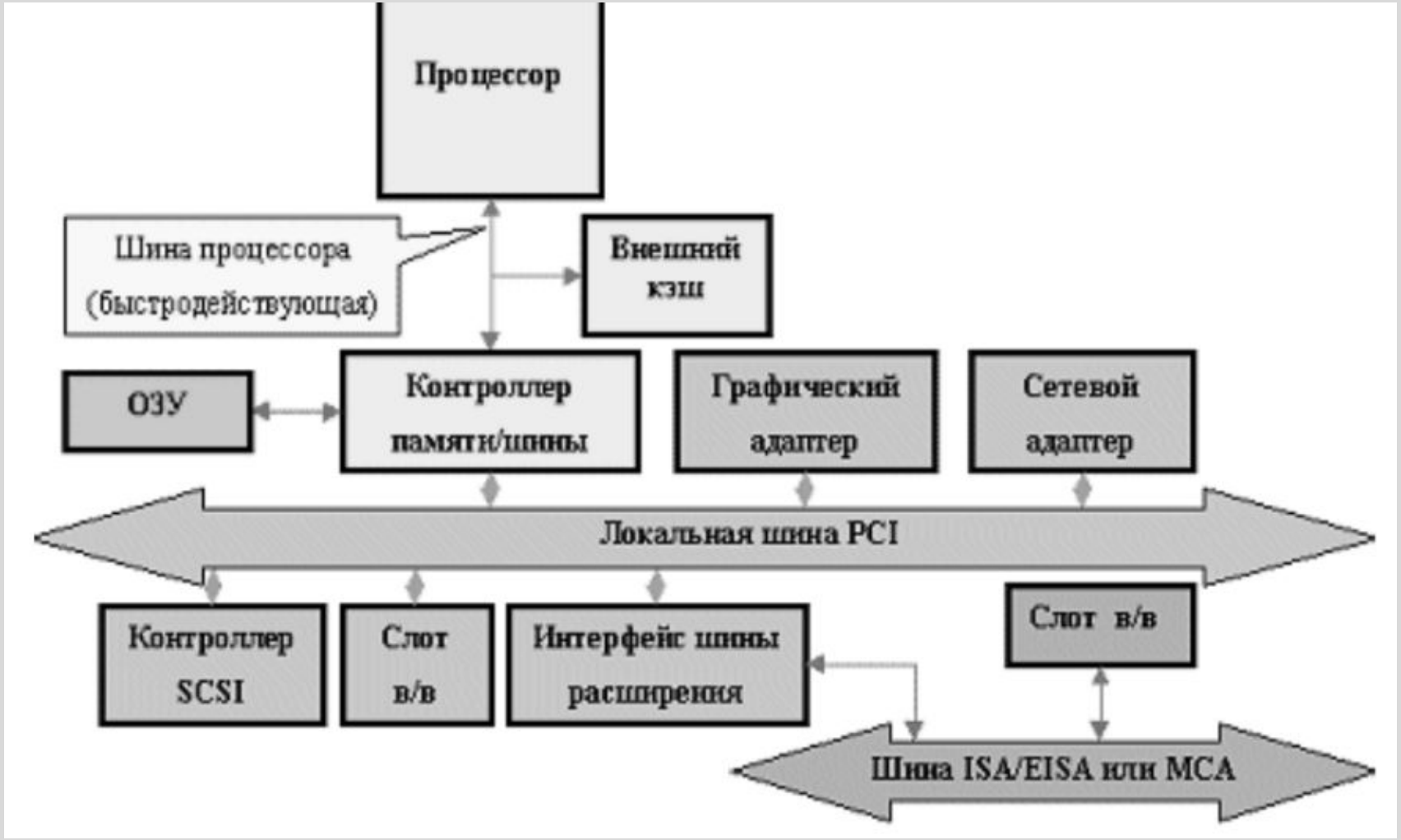
Регистры
Счетчик команд
Состояние процесса
Приоритет
Идентификатор процесса
Родительский процесс
Время запуска процессора
Использованное время процессора

Корневой каталог
Рабочий каталог
Дескрипторы файлов
Идентификатор пользователя

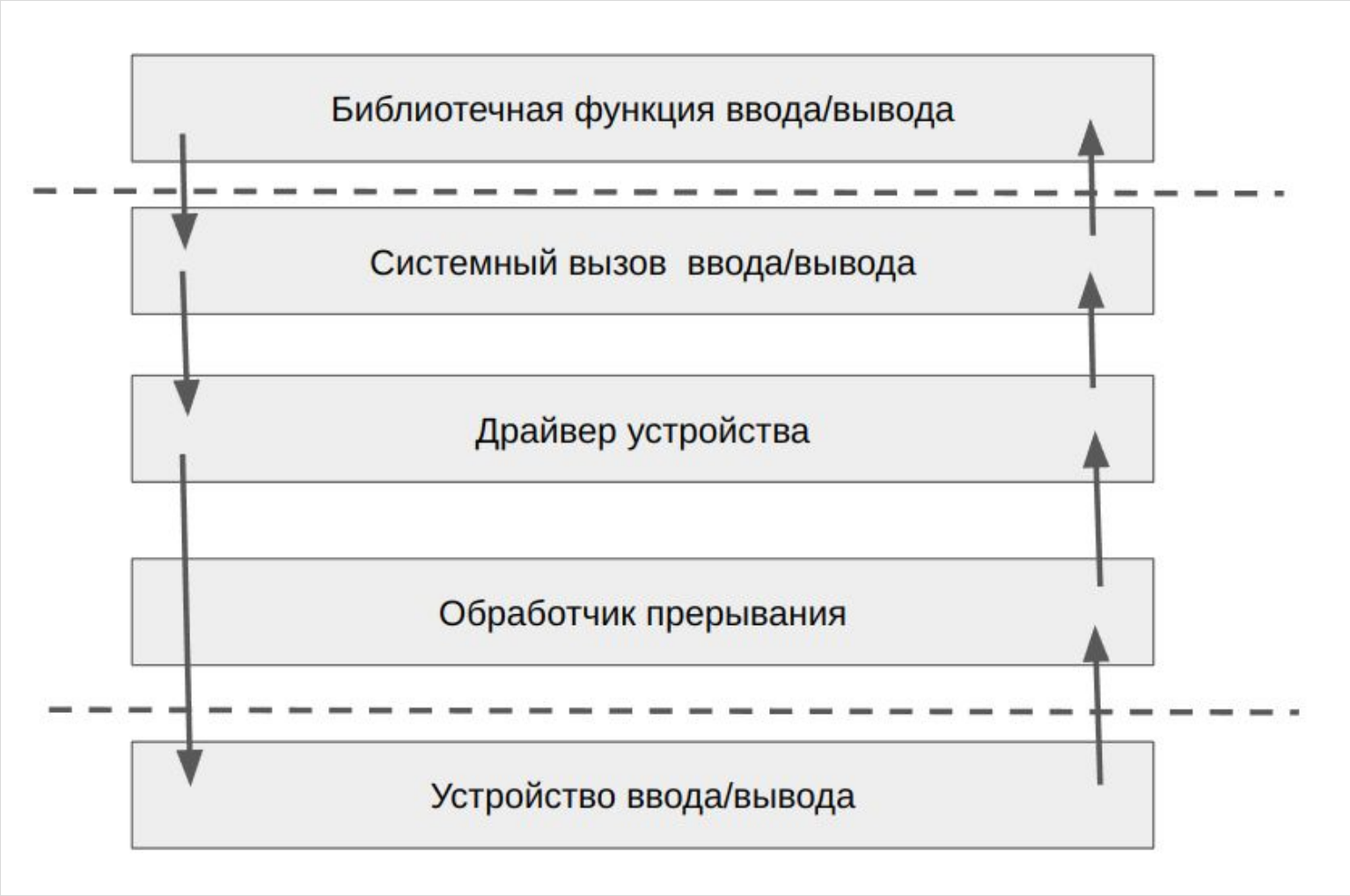
Подсистема ввода/вывода.

Внешние устройства, устройства ввода/вывода, компьютера снабжены контроллерами - микросхемами, управляющими их работой и связью с центральным процессором. У контроллера есть регистры и буфер. Физически контроллеры подключаются с помощью шины, обеспечивающей обмен данными с центральным процессором по определённому протоколу. Программно взаимодействие с операционной системой осуществляют *драйверы* - аппаратно зависимые модули ядра. Драйверы, в соответствие с запросом системного вызова инициализируют регистры контроллера, передавая контроллеру код команды и её параметры. После выполнения команды контроллер инициирует *прерывание* - выставляет сигнал на линии прерываний. Центральный процессор прерывает выполнение текущей задачи и выполняет *обработчик прерываний* - код ядра, входной адрес которого указан в таблице векторов прерываний.


ЦП
АЛУ
УУ
A2
0011 0101 1110 0111
Системная шина
ОЗУ
00
01
XY
A0
35
A1
E7
XY
FF
К1
К2
К3
К4
Кn
Контроллеры
CD/DVD дисковод
HDD
Модем
Монитор
Внешние устройства


Процессор
Шина процессора
(быстродействующая)
Внешний
кэш
ОЗУ
Контроллер
памяти/шины
Графический
адаптер
Сетевой
адаптер
Локальная шина PCI
Контроллер
SCSI
Слот
в/в
Интерфейс шины
расширения
Слот в/в
Шина ISA/EISA или MCA


ЦПУ
Контроллер ввода/вывода
Драйвер устройства инициирует I/O
Инициирование I/O
ЦПУ выполняет проверки на прерывания между инструкциями
ЦПУ, получив прерывание, передаёт контроль обработчику прерываний
Готовность к вводу, окончание вывода, или ошибка генерируют сигнал прерывания
Обработчик прерывания обрабатывает данные, возвращается из прерывания
ЦПУ возобновляет выполнение прерванной задачи


Библиотечная функция ввода/вывода
Системный вызов ввода/вывода
Драйвер устройства
Обработчик прерывания
Устройство ввода/вывода

Примечание. Наряду с аппаратными прерываниями существуют внутренние прерывания - программные, инициированные в коде программ (инструкция INT <номер прерывания>), и исключения, порожденные нештатными событиями (например, попытка деления на ноль, некорректное обращение к памяти и т.д.).

СПАСИБО ЗА ВНИМАНИЕ!