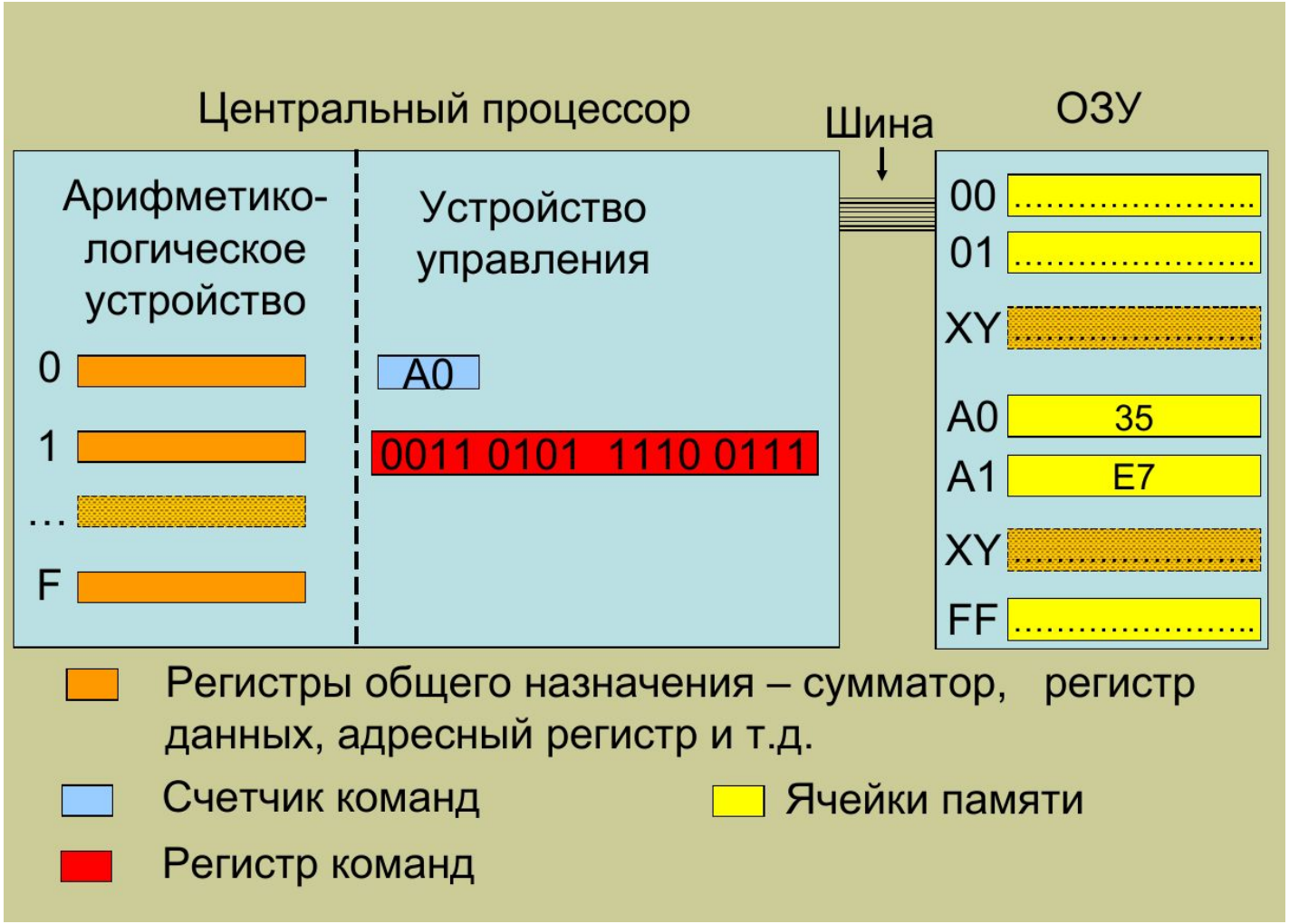
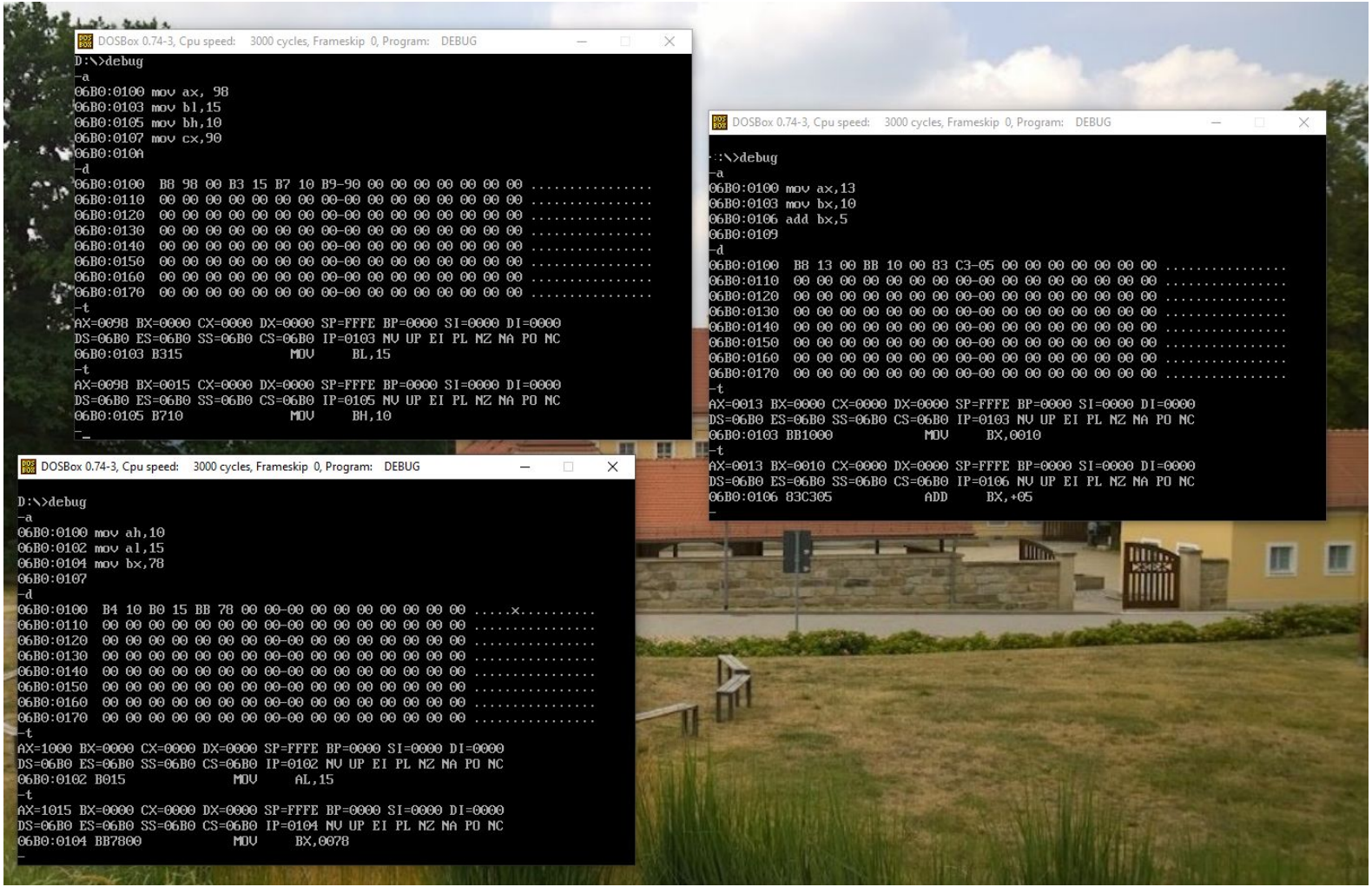
Лекция 2

- Управление ресурсами вычислительной системы (мотивация).
- Режим ядра и пользовательский режим.
- Системные вызовы.
- Интерфейсы прикладного программирования.


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
D:\>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
-

```
.global main

main:
  movl $0x11, %eax
  mov $258, %rbx
  movb $9, %ch
ret
```

```
.global main

main:
  movl $0xaf, %eax
  mov $512, %rbx
  movb $9, %cl
  add $16,%rbx
ret
```

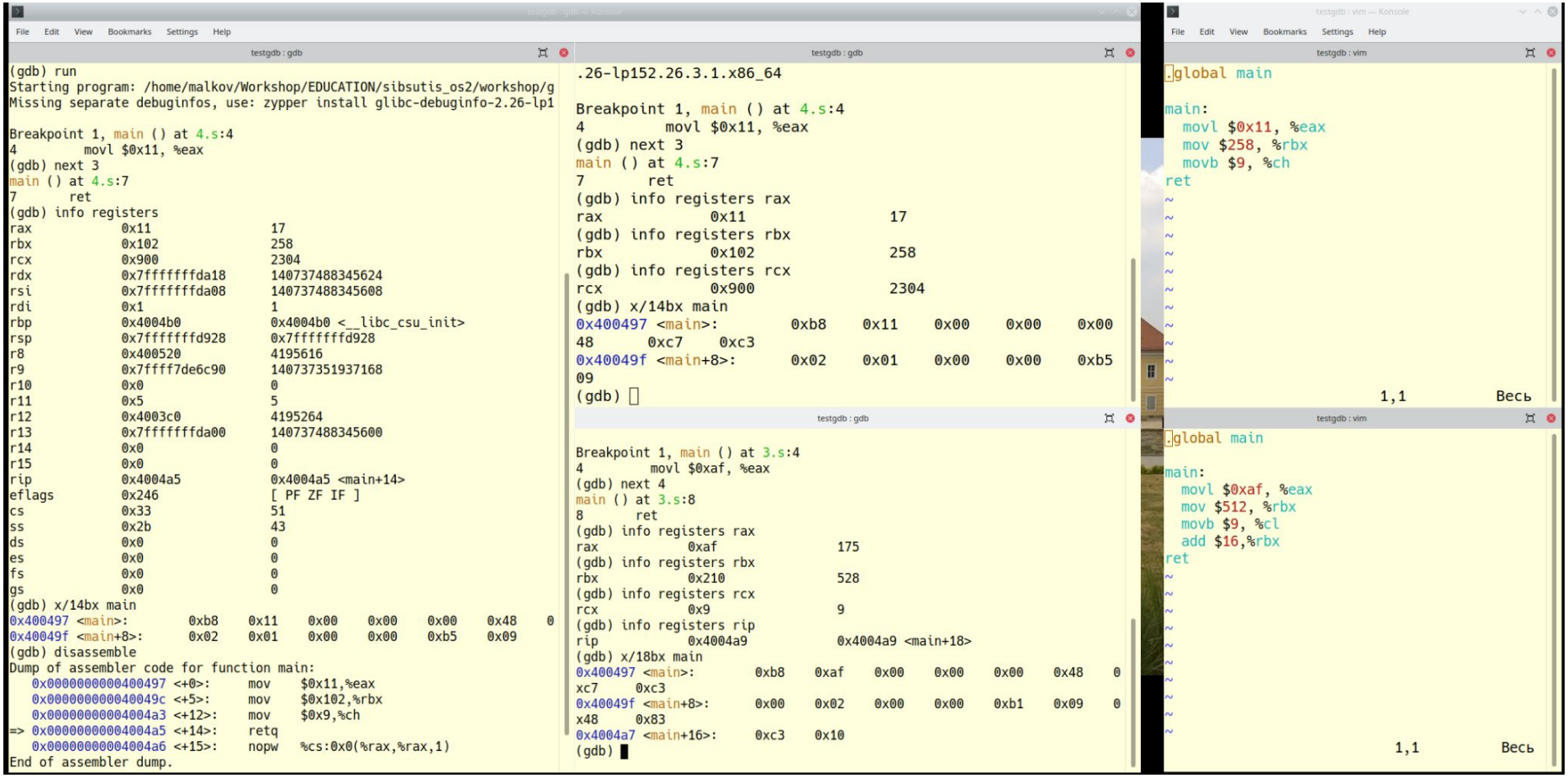
```
> gcc 1.s  -g -o 1
```

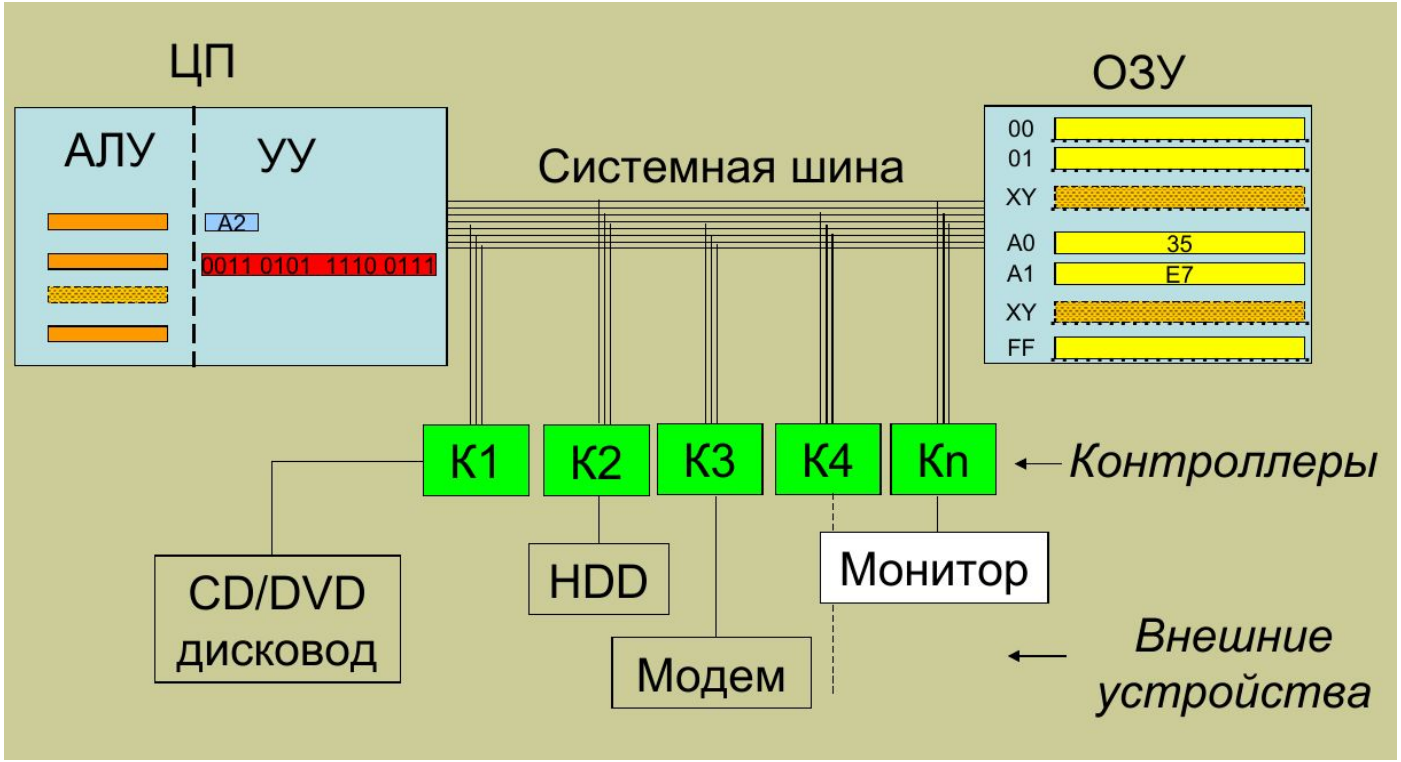
```
> gdb 1
```

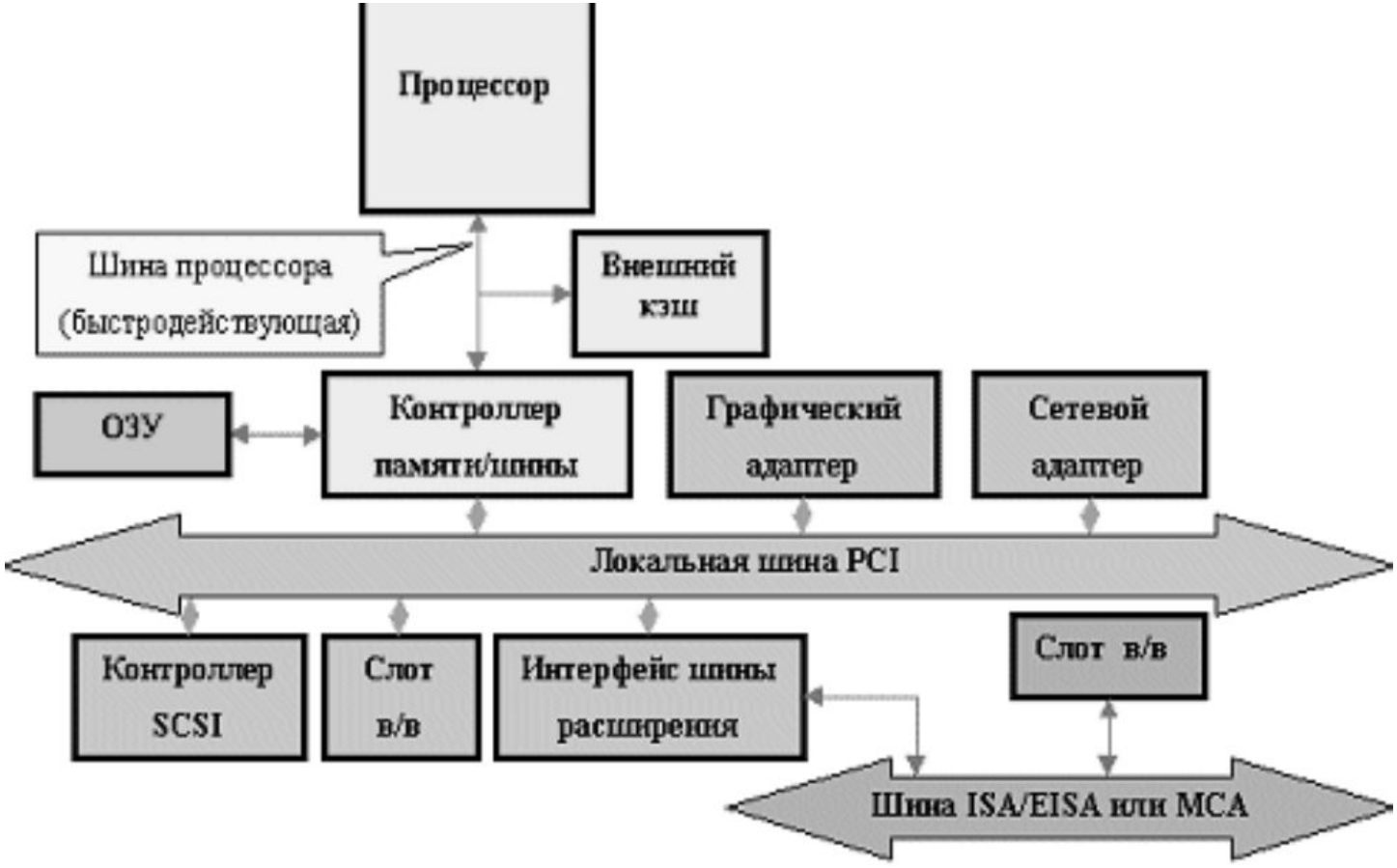
```
> (gdb) break 1
```

```
> (gdb) run
```

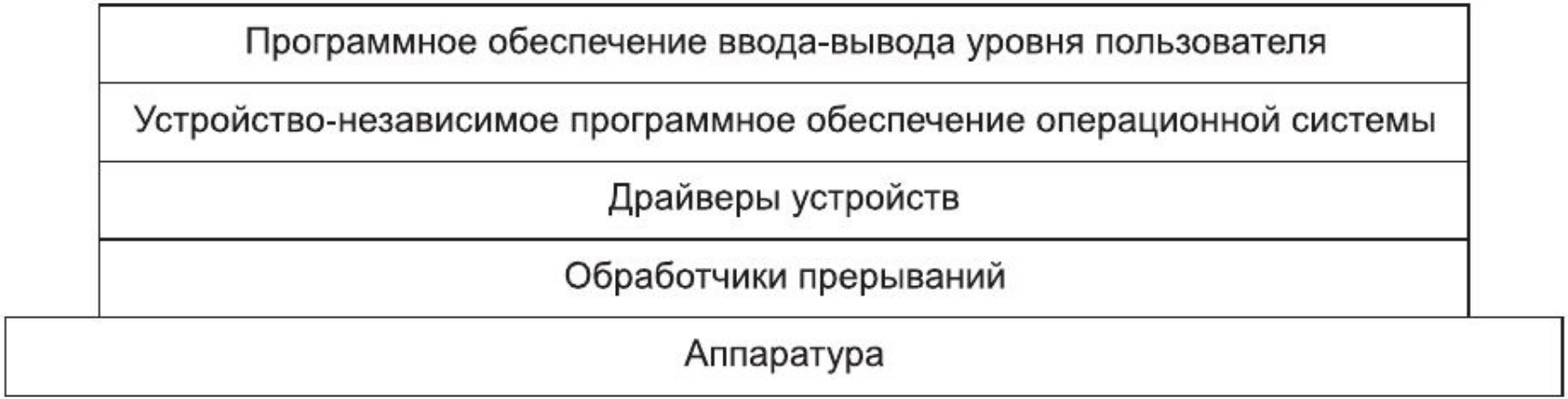
https://www.opennet.ru/base/dev/from_c_to_asm.txt.html


File Edit View Bookmarks Settings Help
testgdb : gdb
(gdb) run
Starting program: /home/malkov/Workshop/EDUCATION/sibsutis_os2/workshop/g
Missing separate debuginfos, use: zypper install glibc-debuginfo-2.26-lp1
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 0x11 17
rbx 0x102 258
rcx 0x900 2304
rdx 0x7fffffffda18 140737488345624
rsi 0x7fffffffda08 140737488345608
rdi 0x1 1
rbp 0x4004b0 0x4004b0 <__libc_csu_init>
rsp 0x7fffffffd928 0x7fffffffd928
r8 0x400520 4195616
r9 0x7ffff7de6c90 140737351937168
r10 0x0 0
r11 0x5 5
r12 0x4003c0 4195264
r13 0x7fffffffda00 140737488345600
r14 0x0 0
r15 0x0 0
rip 0x4004a5 0x4004a5 <main+14>
eflags 0x246 [PF ZF IF]
cs 0x33 51
ss 0x2b 43
ds 0x0 0
es 0x0 0
fs 0x0 0
gs 0x0 0
(gdb) x/14bx main
0x400497 <main>: 0xb8 0x11 0x00 0x00 0x00 0x48 0
0x40049f <main+8>: 0x02 0x01 0x00 0x00 0xb5 0x09
(gdb) disassemble
Dump of assembler code for function main:
0x0000000000400497 <+0>: mov $0x11,%eax
0x000000000040049c <+5>: mov $0x102,%rbx
0x00000000004004a3 <+12>: mov $0x9,%ch
=> 0x00000000004004a5 <+14>: retq
0x00000000004004a6 <+15>: nopw %cs:0x0(%rax,%rax,1)
End of assembler dump.
testgdb : gdb
.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
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)
testgdb : vim — Konsole
File Edit View Bookmarks Settings Help
testgdb : vim
.global main
main:
movl $0x11, %eax
mov $258, %rbx
movb $9, %ch
ret
~
1,1 Весь
testgdb : vim
.global main
main:
movl $0xaf, %eax
mov $512, %rbx
movb $9, %cl
add $16,%rbx
ret
~
1,1 Весь


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



Архитектура компьютера с шиной PCI


Программное обеспечение ввода-вывода уровня пользователя
Устройство-независимое программное обеспечение операционной системы
Драйверы устройств
Обработчики прерываний
Аппаратура


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

Взаимодействие прикладных программ и ОС

Режим ядра (режим супервизора, привилегированный режим):

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

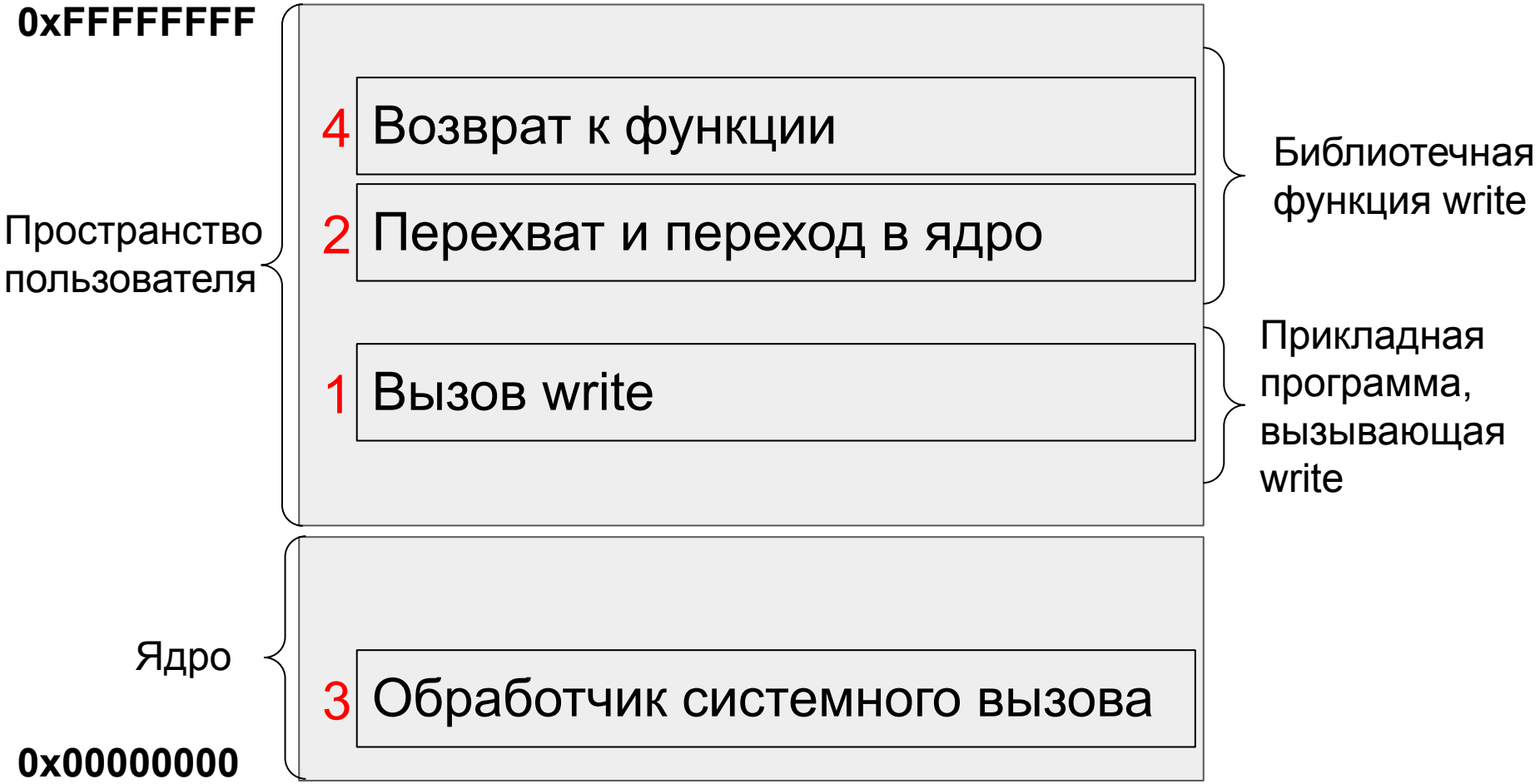
Пользовательский режим:

- 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.

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	-	-

> strace ./ts1 > /dev/null

```
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
……...
mmap(0x7f96a376b000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b1000) =
0x7f96a376b000
mmap(0x7f96a3771000, 14200, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f96a3771000
close(3)                                = 0
……..
mprotect(0x7f96a399a000, 4096, PROT_READ) = 0
munmap(0x7f96a396c000, 184780)          = 0
…….
brk(0x1cdb000)                          = 0x1cdb000
write(1, "Hello world\n", 12)           = 12
exit_group(0)                           = ?
```

```
> uname -a
Linux 192.168.0.6 5.3.18-lp152.57-default #1 SMP Fri Dec 4
07:27:58 UTC 2020 (7be5551) x86_64 x86_64 x86_64 GNU/Linux
```

```
#include <stdio.h>
#include <sys/utsname.h>

int main(){
 struct utsname  u;
 uname(&u);
 printf("%s release  %s  (version  %s) on %s\n", u.sysname, u.release, u.version,
u.machine);
 return 0;
}
```

```
> ./un
Linux release  5.3.18-lp152.57-default  (version  #1 SMP Fri
Dec 4 07:27:58 UTC 2020 (7be5551)) on x86_64
```