User

Kernel

User level Software

Device Independent Software

Device DriverS

Polling
Intreruperi
Direct-memory access

Cum folosim hardware?

Intreruperi

Hardware

Dispozitive de Intrare/Iesire

SO Curs 10

```
int fd = open / socket / pipe / ...;
read(fd, buf, 10);
write(fd2, buf, 10);
close(fd);
```

Dispozitivele de I/O sunt foarte diverse

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Ce vor utilizatorii?

Usurinta in utilizare:

- cod care ruleaza pe oricate dispozitive de intrare-iesire
- sa fie usor sa numeasca dispozitivele
- tratarea erorilor sa se intample automat

Performanta

Operatii dorite:

- sincrone (majoritatea)
- asincrone (unele aplicatii)

Cum implementam?

User

Kernel

Dispozitive de intrare/ieșire

SO Curs 10

```
int fd = open / socket / pipe / ...;
read(fd, buf, 10);
write(fd2, buf, 10);
close(fd);
```

User level Software

Device Independent Software

Device DriverS

Polling
Intreruperi
Direct-memory access

Cum folosim hardware?

Intreruperi

Hardware

Hardware

Doua categorii de dispozitive: caracter si bloc

Fiecare dispozitiv are:

- un controller
- unul sau mai multe porturi
- conectat la o magistrala

Procesorul da comenzi de I/O controllerului

Controllerul gestioneaza dispozitivul

Hardware

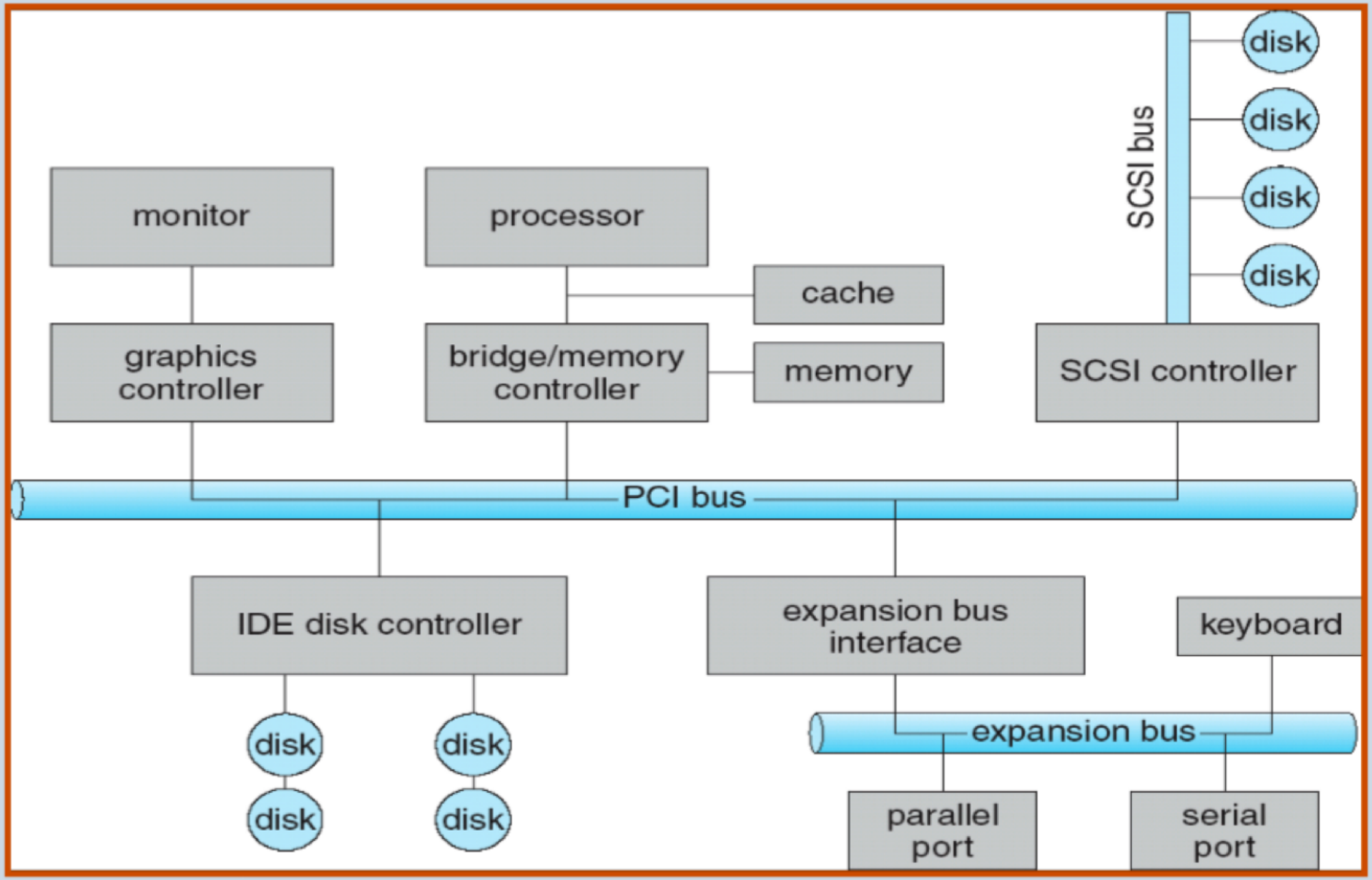
Doua categorii de dispozitive: caracter si bloc

Fiecare dispozitiv are:

- un controller
- unul sau mai multe porturi
- conectat la o magistrala

Procesorul da comenzi de I/O controllerului

Controllerul gestioneaza dispozitivul


monitor
processor
cache
graphics controller
bridge/memory controller
memory
SCSI bus
disk
disk
disk
disk
SCSI controller
PCI bus
IDE disk controller
expansion bus interface
keyboard
disk
disk
disk
disk
expansion bus
parallel port
serial port

Comunicatia cu dispozitivele de I/E

Fiecare controller are:

- registre de stare, control, intrare, iesire
- (optional) buffer pentru date

Cum adresam un dispozitiv de I/O?

Port-mapped I/O

Spatiu de adresa separat pentru dispozitivele de I/O
O adresa de I/O se numeste port
Un port adreseaza un registru al controllerului
Instructiuni specializate:
- IN R0, 0x2F2
- OUT REG, 0x2FA

Memory mapped I/O

Registrele I/O sunt mapate in spatiul de memorie
Avantaje:
- nu necesita instructiuni specializate
- protectia de la memoria virtuala
- instructiunile pot referi memorie sau registre

Dezavantaje:

Port-mapped I/O

Spatiu de adresa separat pentru dispozitivele de I/O

O adresa de I/O se numeste port

Un port adreseaza un registru al controllerului

Instructiuni specializate:

- IN REG, 0x2F8
- OUT REG, 0x2FA

Memory mapped I/O

Registrele I/O sunt mapate in spatiul de memorie

Avantaje:

- nu necesita instructiuni specializate
- protectia de la memoria virtuala
- instructiunile pot referi memorie sau registre

Dezavantaje:

- trebuie inhibat cache-ul la nivel de pagina
- bridge-uri intre magistrale-> mai lent decat port-mapped I/O

User

Kernel

Dispozitive de Intrare/Ieșire

SO Curs 10

```
int fd = open / socket / pipe / ...;
read(fd, buf, 10);
write(fd2, buf, 10);
close(fd);
```

User level Software

Device Independent Software

Device DriverS

Polling

Intreruperi

Direct-memory access

Cum folosim hardware?

Intreruperi

Hardware

Prezi

verS

Polling

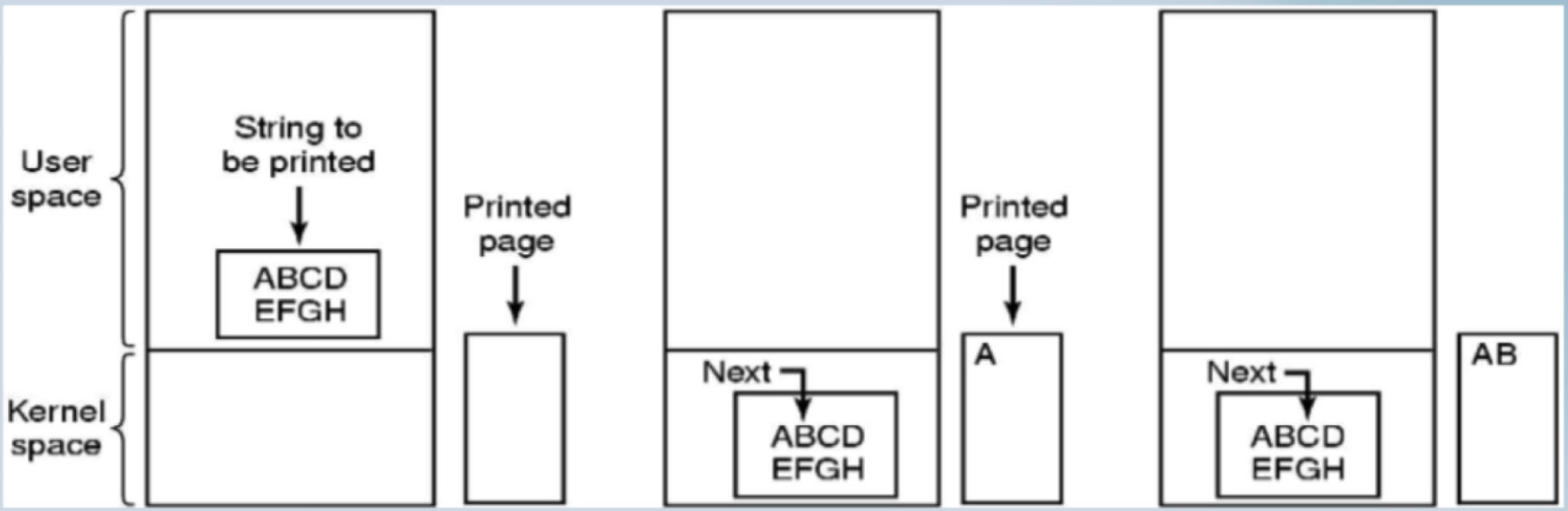
Intreruperi

Direct-memory access

Direct Memory Access

DMA: dispozitiv separat specializat pentru copieri din memorie in memorie
CPU comanda DMA sa efectueze transferuri date ca (sursa,destinatie,octeti)
CPU nu poate accesa magistrala daca este folosita de DMA

Cum folosim hardware?

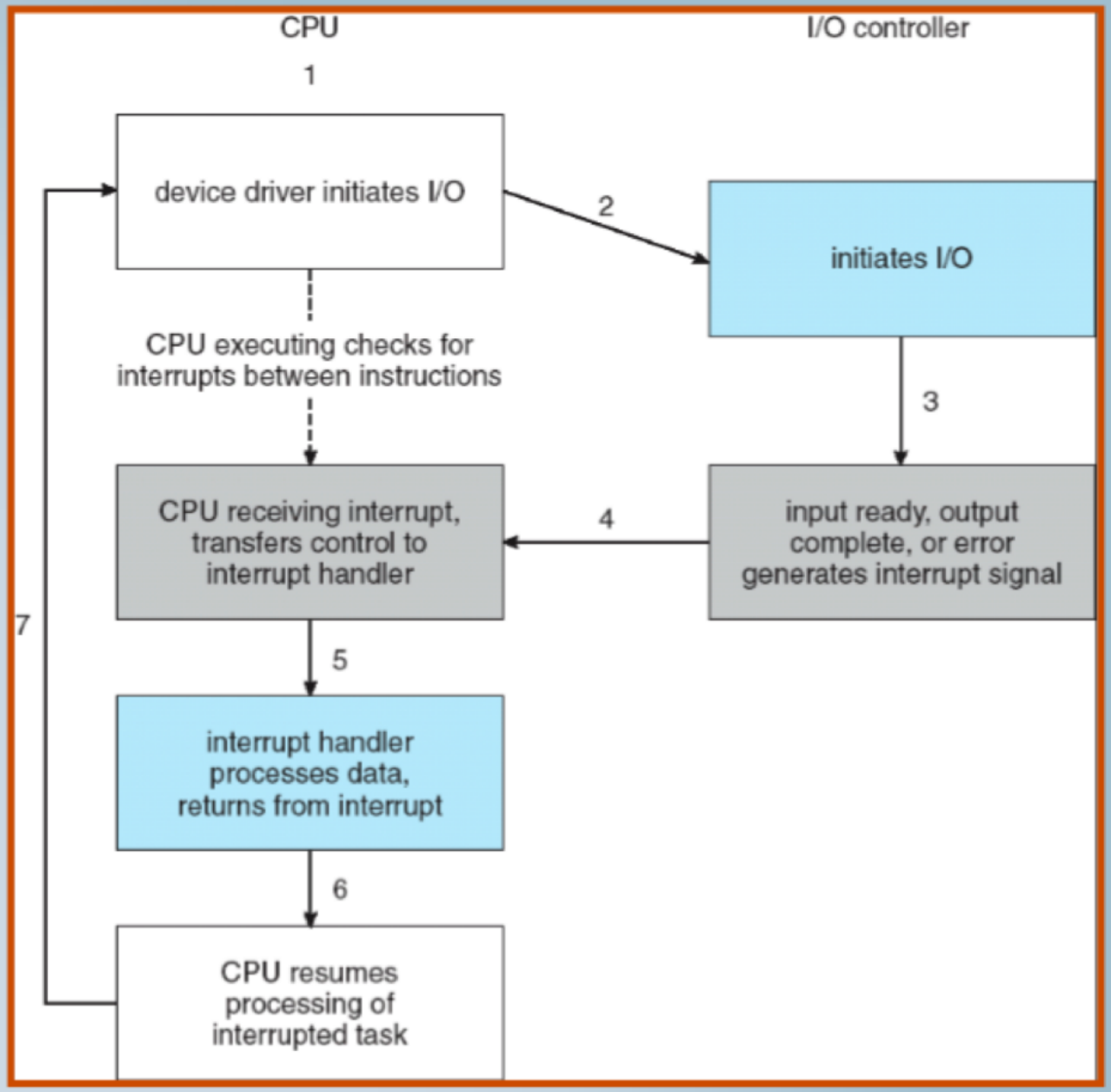



```
copy_from_user(buffer, p, count);
for (i=0;i<count;i++){
        while (*printer_status_req!=READY);
        printer_data_register = p[i];
}
return_to_user();
```

Intreruperi

Anunta procesorul ca s-a intamplat un eveniment care trebuie tratat

- pot fi mascabile sau nemascabile
- sunt generate de software sau hardware (controller)
- folosesc linii de intrerupere (IRQ line)
- exista o tabela cu rutine de tratare a intreruperilor
- procesorul ruleaza rutina de tratare a intreruperii (ISR)


CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
7
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task

Implementare cu intreruperi

```
copy_from_user(buffer, p , count);
enable_interrupts();
while (*printer_status_reg!=READY);
*printer_data_register = p[0];
scheduler();
```

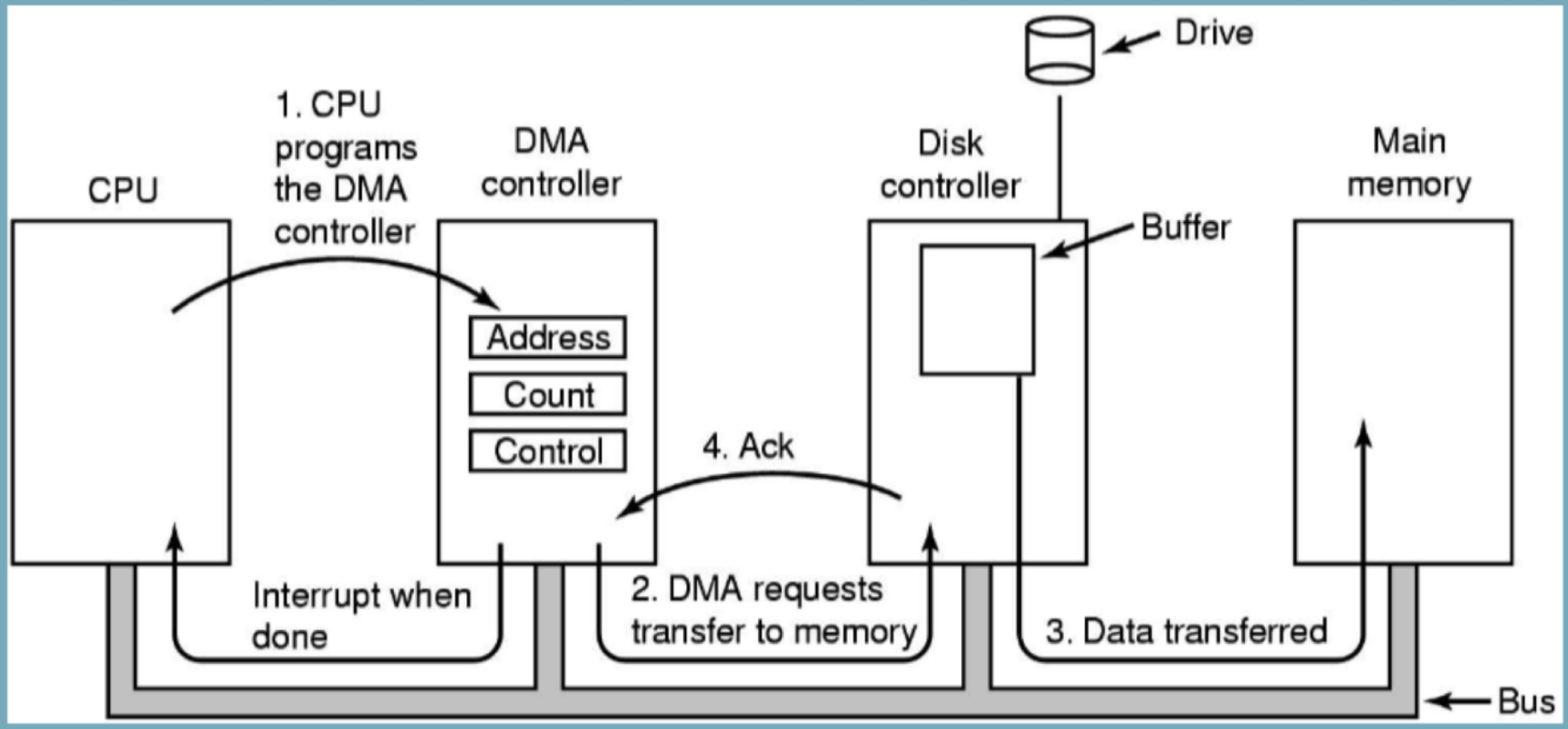
```
if (count==0) {
        unblock_user();
}
else {
        *printer_data_register = p[i];
        count --; i++;
}
acknowledge_interrupt();
return_from_interrupt;
```

Direct Memory Access

DMA: dispozitiv separat specializat pentru copieri din memorie in memorie

CPU comanda DMA sa efectueze transferuri date ca (sursa,destinatie,octeti)

CPU nu poate accesa magistrala daca este folosita de DMA


Drive
1. CPU programs the DMA controller
DMA controller
Disk controller
Main memory
CPU
Buffer
Address
Count
Control
4. Ack
Interrupt when done
2. DMA requests transfer to memory
3. Data transferred
Bus

Implementare cu DMA

```
copy_from_user(buffer,p,count);
set_up_DMA_controller();
scheduler();
```

```
acknowledge_interrupt();
unblock_user();
return_from_interrupt();
```

User-level software

Device Independent Software

Operatii independente de dispozitiv

Planificare pentru eficienta si echitate:
- sortare si comasare de cereri de I/O

Buffering
- Compensarea vitezei de transfer dintre dispozitive
- Reasamblarea datelor
- Semantica de copiere pentru utilizatori

Operatii independente de dispozitiv (2)

Caching: memoria tine copii ale datelor

Spooling: se mentine un buffer cu datele transmise unui dispozitiv

Rezervarea dispozitivului

Fluxul unei operatii de I/O

Device Drivers

Polling

Intreruperi

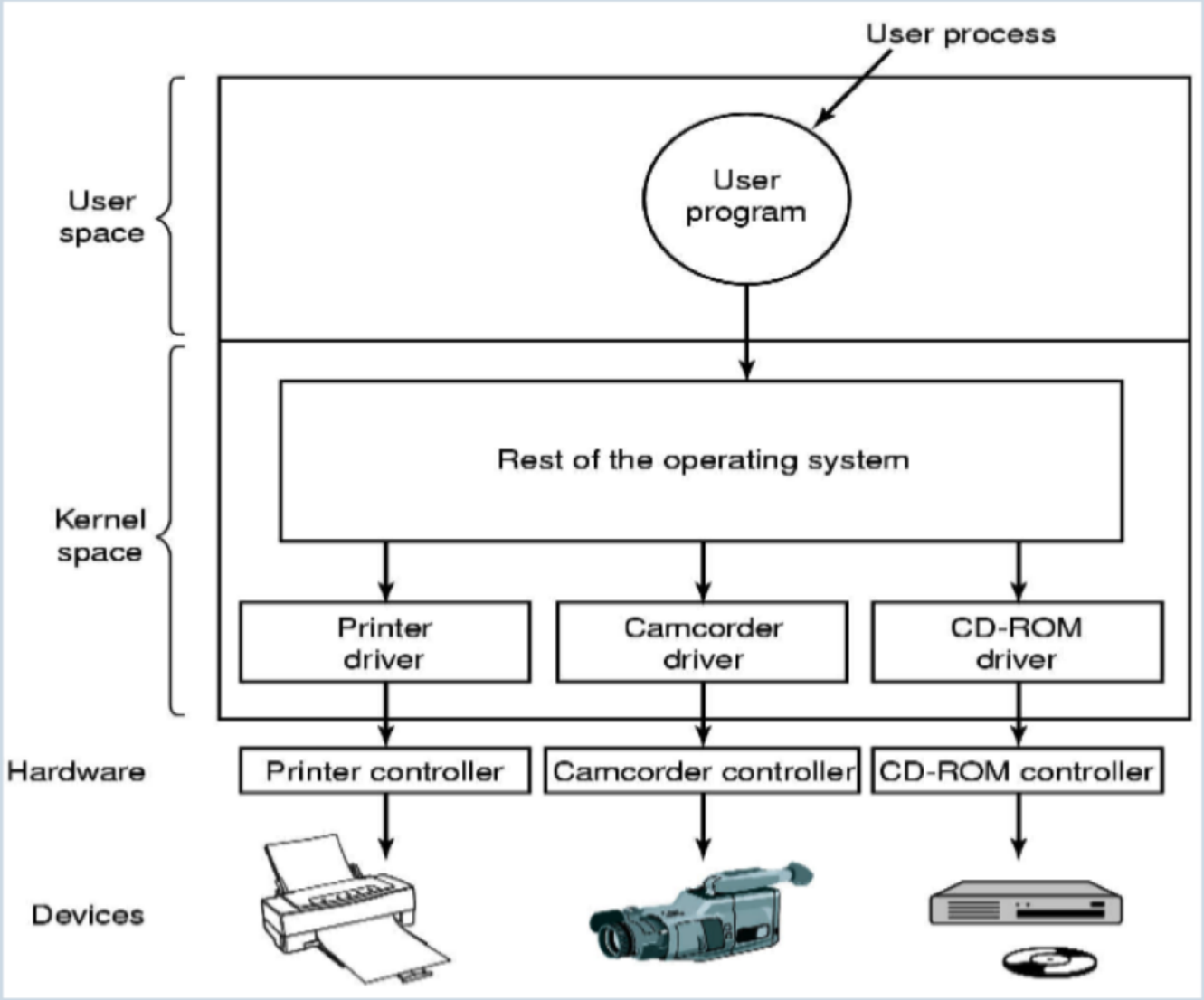
Direct-memory access

Cum folosim hardware?

Intreruperi

Prezi

Device Drivers



- controleaza un dispozitiv sau clasa de dispozitive
- ruleaza in kernel
- SO ofera interfata comuna pentru drivere

User level Software

Device Independent Software

Operatii independente de dispozitiv

Planificare pentru eficienta si echitate:
- sortare si comasare de cereri de I/O

Buffering
- Compensarea vitezei de transfer dintre dispozitive
- Reasamblarea datelor
- Semantica de copiere pentru utilizatori

Operatii independente de dispozitiv (2)

Caching: memoria tine copii ale datelor

Spooling: se mentine un buffer cu datele transmise unui dispozitiv

Rezervarea dispozitivului

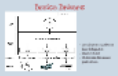
Fluxul unei operatii de I/O

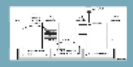
Device Drivers

Polling

Intreruperi

Direct-memory access





Cum folosim hardware?

Intreruperi

Operatii independente de dispozitiv

Planificare pentru eficienta si echitate:

- sortare si comasare de cereri de I/O

Buffering

- Compensarea vitezei de transfer dintre dispozitive
- Reasamblarea datelor
- Semantica de copiere pentru utilizatori

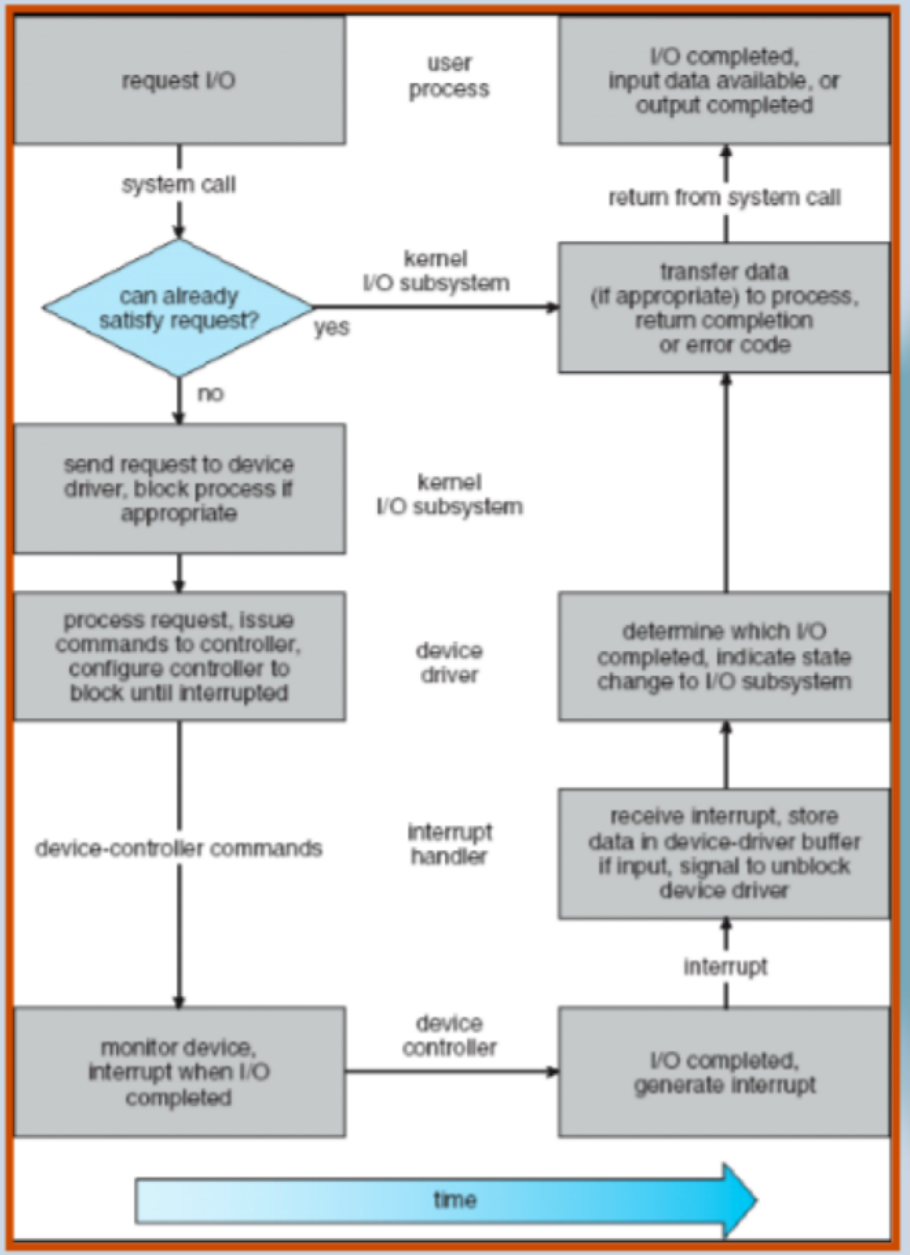
Operatii independente de dispozitiv (2)

Caching: memoria tine copii ale datelor

Spooling: se mentine un buffer cu datele transmise unui dispozitiv

Rezervarea dispozitivului

Fluxul unei operatii de I/O



close(fd);

User level Software

Device Independent Software

Operatii independente de dispozitiv

Planificare pentru eficienta si echitate:
- sortare si comasare de cereri de I/O

Buffering
- Compensarea vitezei de transfer dintre dispozitive
- Reasamblarea datelor
- Semantica de copiere pentru utilizatori

Operatii independente de dispozitiv (2)

Caching: memoria tine copii ale datelor

Spooling: se mentine un buffer cu datele transmise unui dispozitiv

Rezervarea dispozitivului

Fluxul unei operatii de I/O

Device DriverS

Polling
Intreruperi
Direct-memory access

Cum folosim hardware?

Intreruperi

Hardware

Doua categorii de dispozitive: caracter si bloc
Fiecare dispozitiv are:
- un controller
- unul sau mai multe porturi
- conectat la o magistrala

Procesorul da comenzi de I/O controllerului
Controllerul gestioneaza dispozitivul

API utilizator

Char Devices

e.g. mouse, tastatura

Acces secvential

Operatii: put, get

Transfer la nivel de caracter

Viteza redusa

Block Devices

e.g. discuri, CDROM

Acces aleator

Operatii: read, write, seek

Transfer la nivel de bloc

Viteza ridicata

Network

e.g. 802.11, 802.5

Separare protocol de interfata

Sockets

Operatii: connect, send, recv, close

Viteza ridicata

Initializare

Dispozitive block/caracter

open, close

Dispozitive retea

socket, connect, shutdown

Control

Controleaza dispozitivul de I/O

Depinde de dispozitiv!

- ioctl / DeviceIOControl
- setsockopt, getsockopt

Tipuri de operatii I/O

Blocante

Procesul este suspendat pana la incheierea operatiei
Simplu de folosit

Ne-blocante

Operatia se intoarce imediat
Procesul primeste datele disponibile

Asincrone

Procesul ruleaza in paralel cu operatia
Este notificat atunci cand operatia este finalizata
Event-driven programming

Tipuri de operatii I/O

	blocante	ne-blocante
sincrone	read/write ReadFile, WriteFile	read/write O_NONBLOCK
asincrone	select	AIO aio_*, overlapped I/O

ASynchronous I/O

Unix	Window
aio_read	ReadFile
aio_write	WriteFile
aio_suspend	OVERLAPPED I/O

User

Kernel

Dispozitive de Intrare/Iesire

SO Curs 10

```
int fd = open / socket / pipe / ...;
read(fd, buf, 10);
write(fd2, buf, 10);
close(fd);
```

User level Software

Device Independent Software

Device Drivers

Polling

Intreruperi

Direct-memory access

Cum folosim hardware?

Intreruperi

Hardware

Doua categorii de dispozitive: caracter si bloc

Fiecare dispozitiv are:

- un controller
- unul sau mai multe porturi
- conectat la o magistrala

Procesorul da comenzi de I/O controllerului

Controllerul gestioneaza dispozitivul