

# User

#### Dispozitive de Intrare/Ieșire

SO Curs

## User level Software

## Device Independent Software

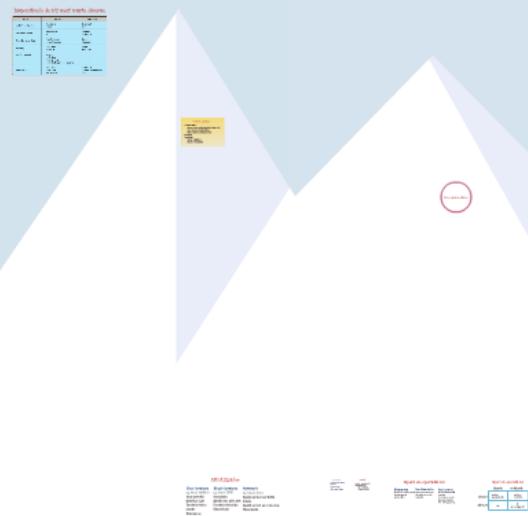
# Device Drivers

## Intreruperi

## Hardware

## Dispozitive de Intrare/IeSire

```
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 intampla automat

Performanta

Operatii dorite:

- sincrone (majoritatea)
- asincrone (unele aplicatii)

**Cum implementam?**



# 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



## Comunicarea cu dispozitivele de I/O

- Rețea controller -> dispozitive de I/O
- (optional) interfață pentru date

• Cum aducem un dispozitiv la I/O?

## Port-mapped I/O

- Rețea controller -> dispozitive de I/O
- (optional) interfață pentru date

• Cum aducem un dispozitiv la I/O?

## Memory mapped I/O

- Rețea controller -> dispozitive de I/O
- (optional) interfață pentru date

• Cum aducem un dispozitiv la I/O?

# Interfață

Doua categorii de dispozitive: caracter si bloc

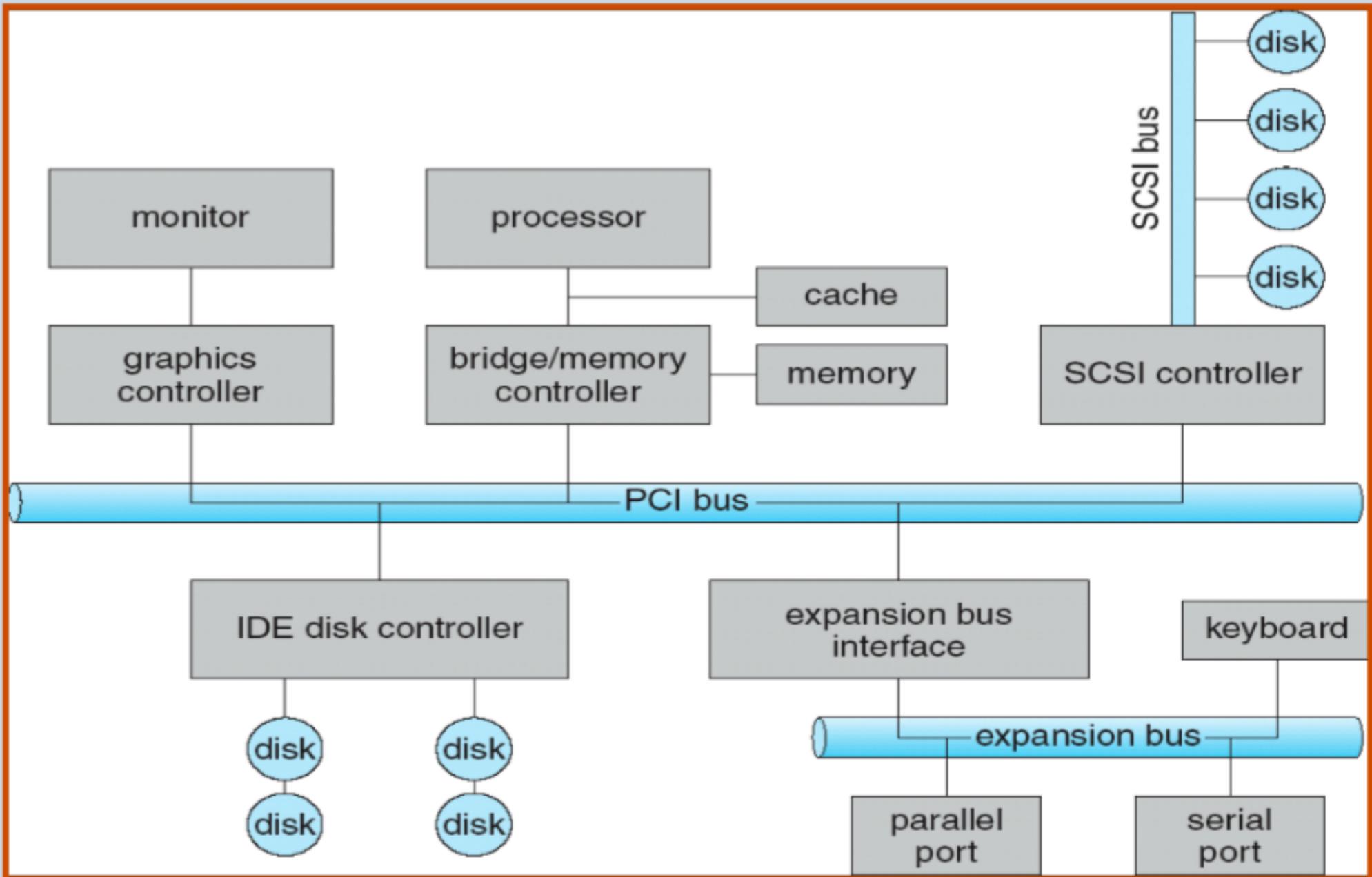
Fiecare dispozitiv are:

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

Procesorul da comenzi de I/O controllerului

Controllerul gestioneaza dispozitivul

Comunica  
Fiecare controller:  
• registre  
• (optional)  
Cum ac  
Port-mapped



# 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 regisztrul 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

Dificultati:

# 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 în spațiul de memorie

Avantaje:

- nu necesita instrucțiuni specializate
- protecția de la memoria virtuală
- instrucțiunile pot referi memorie sau registre

Dezavantaje:

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

# User

# Kernel

Dispozitive de Intrare/Iesire

SO Curs 9

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

USER level Software

Device Independent Software

Device Drivers

Polling

Intreruperi

Direct-memory access

Cum folosim hardware?

Intreruperi

Hardware

Introducere în dispozitivele caracteristice

Fisiere cheie pentru:

- un controller
- un sau mai multe porturi
- conectivitatea magazinului

Procesorul de comunicare cu dispozitivul

Controlul gestionarea dispozitivelor

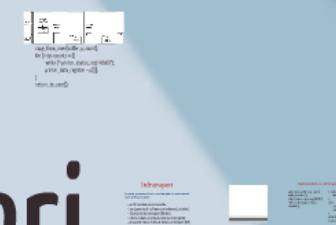


# vers

## Polling Intreruperi

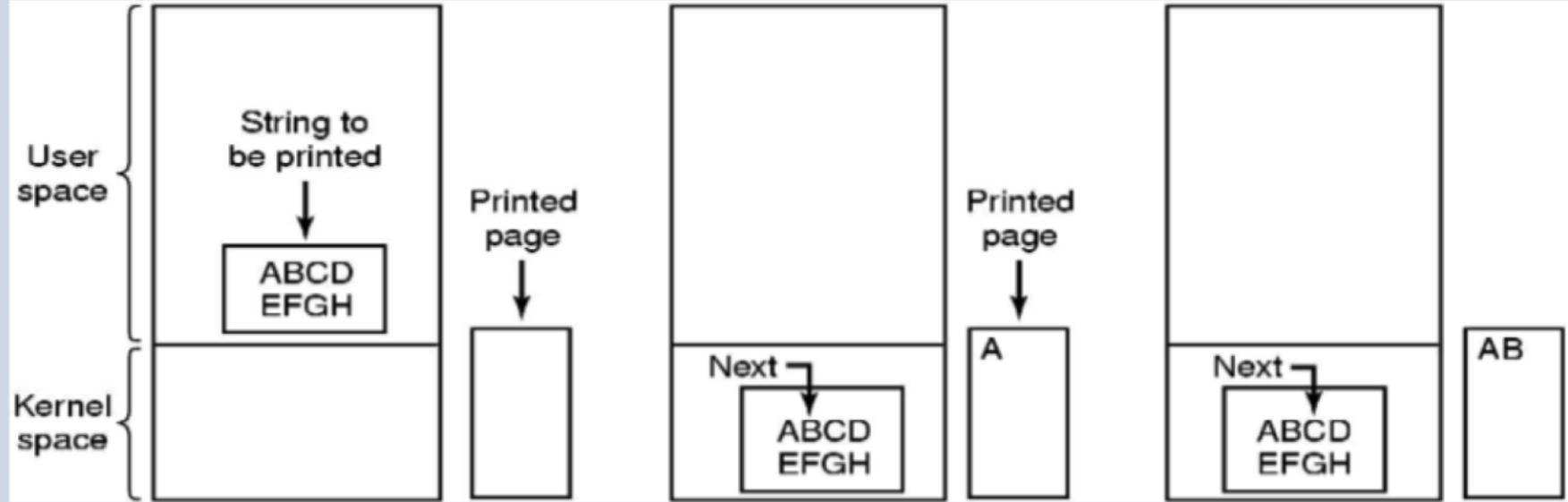
## Direct-memory access

# Cum folosim hardware?



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



```

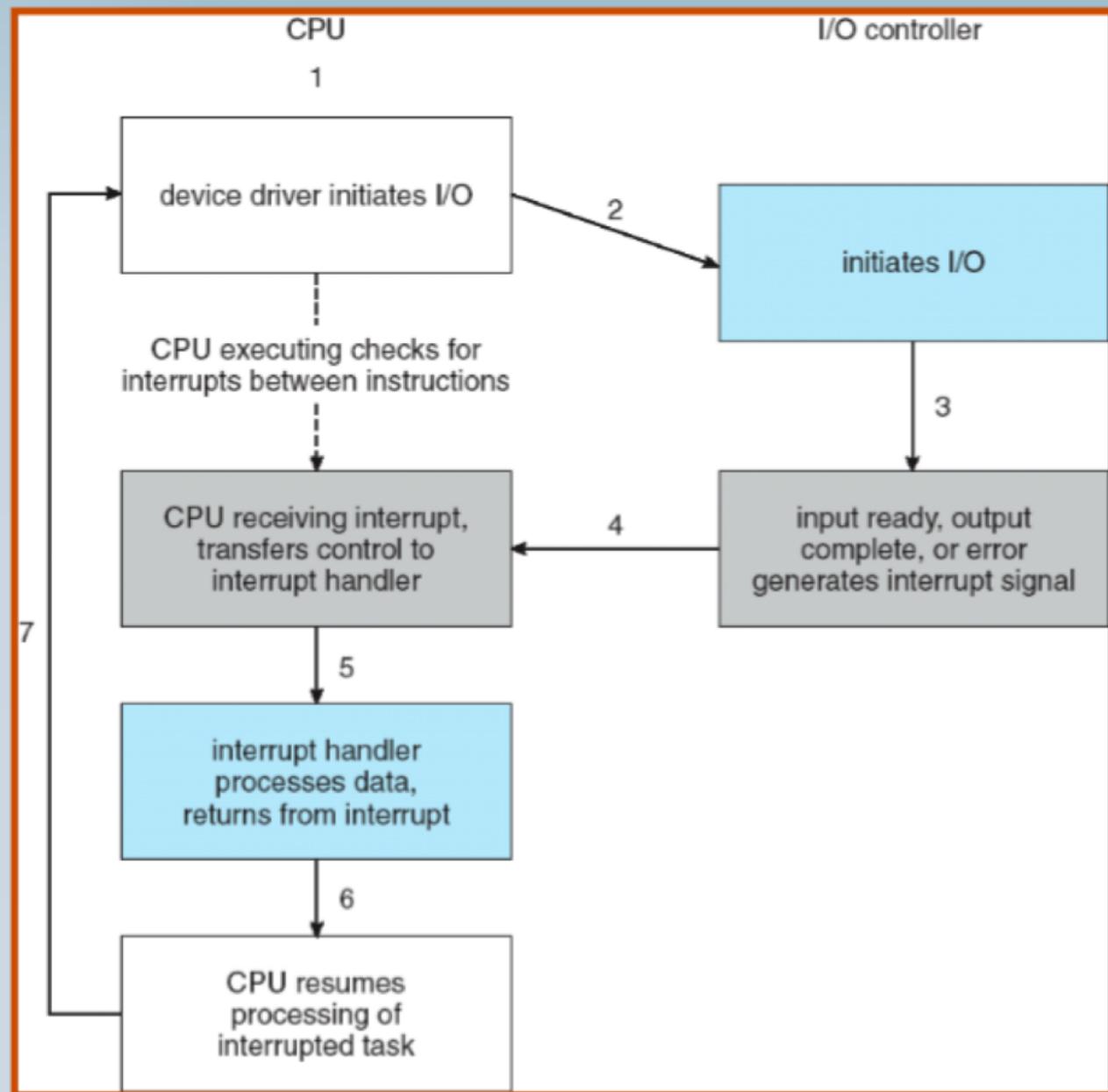
copy_from_user(buffer, p, count);
for (i=0;i<count;i++){
    while (*printer_status_req!=READY);
    printer_data_register = p[i];
}
return_to_user();

```

# 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 interrupere (IRQ line)
- exista o tabela cu rutine de tratare a interruperilor
- procesorul ruleaza rutina de tratare a interruperii (ISR)



## Implementare cu intreruperi

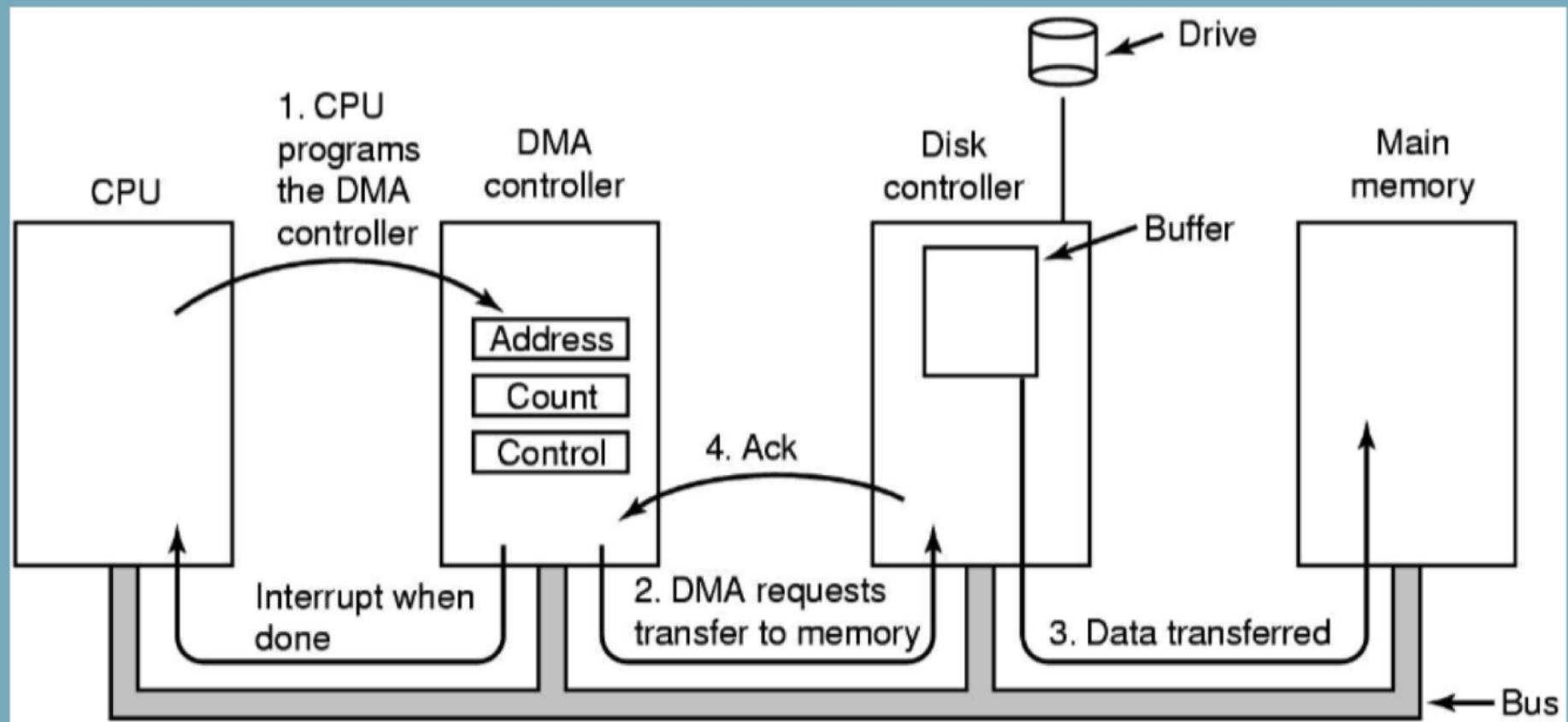
```
copy_from_user(buffer, p , count);
enable_interrupts();
while (*printer_status_reg!=READY);
*printer_data_register = p[0];
scheduler();
if (count==0) {
    unblock_user();
}
else {
    *printer_data_register = p[i];
    count --; i++;
}
acknowledge_interrupt();
return_from_interrupt;
```

# 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



## Implementare cu DMA

```
copy_from_user(buffer,p,count);    acknowledge_interrupt();  
set_up_DMA_controller();          unblock_user();  
scheduler();                      return_from_interrupt();
```

# Device Independent Software

## Operatii independente de dispozitiv

### Planificare pentru eficiență și echitate:

- sortare și 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 ține copii ale datelor

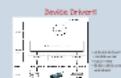
### Spooling: se menține un buffer cu datele transmise unui dispozitiv

### Reservarea dispozitivului



Fluxul unei operatii de I/O

# Device Drivers



Polling  
Intreruperi

Direct-memory access



Direct Memory Access

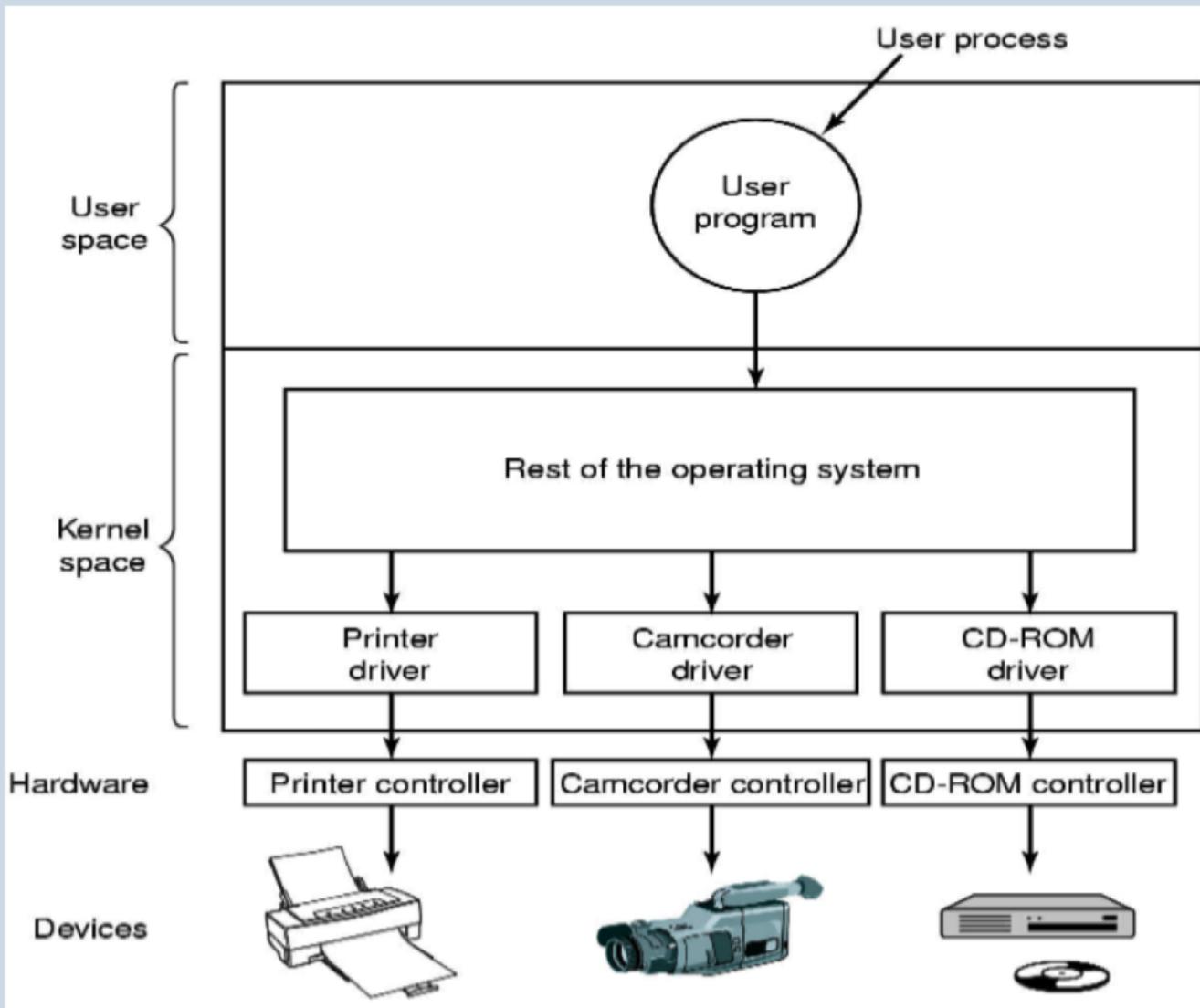
Depanarea cu DMA

(copy per buffer); acknowledge interrupt; add to DMA controller; update loc pointer)

Cum folosim hardware?

# Intreruperi

# Device Drivers



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

# User level Software

## Device Independent Software

### Operatii independente de dispozitiv

Planificare pentru eficiență și echitate:

- sortare și 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 ține copii ale datelor

Spooling: se menține un buffer cu datele transmise

unui dispozitiv

Reservarea dispozitivului

Fluxul unei  
operatii de  
I/O



## Device Drivers

Polling  
Intreruperi  
Direct-memory access



Direct Memory Access

Întrările sunt preluate și trimise la memorie.

OS trimite DMA să verifice în trunchiul date și să le transmită la aplicație.

OS nu este nevoie să aștepte datele să fie lăudate de DMA.



Împreună cu DMA

admeteți (read),

scrie (write),

stergere (erase).

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

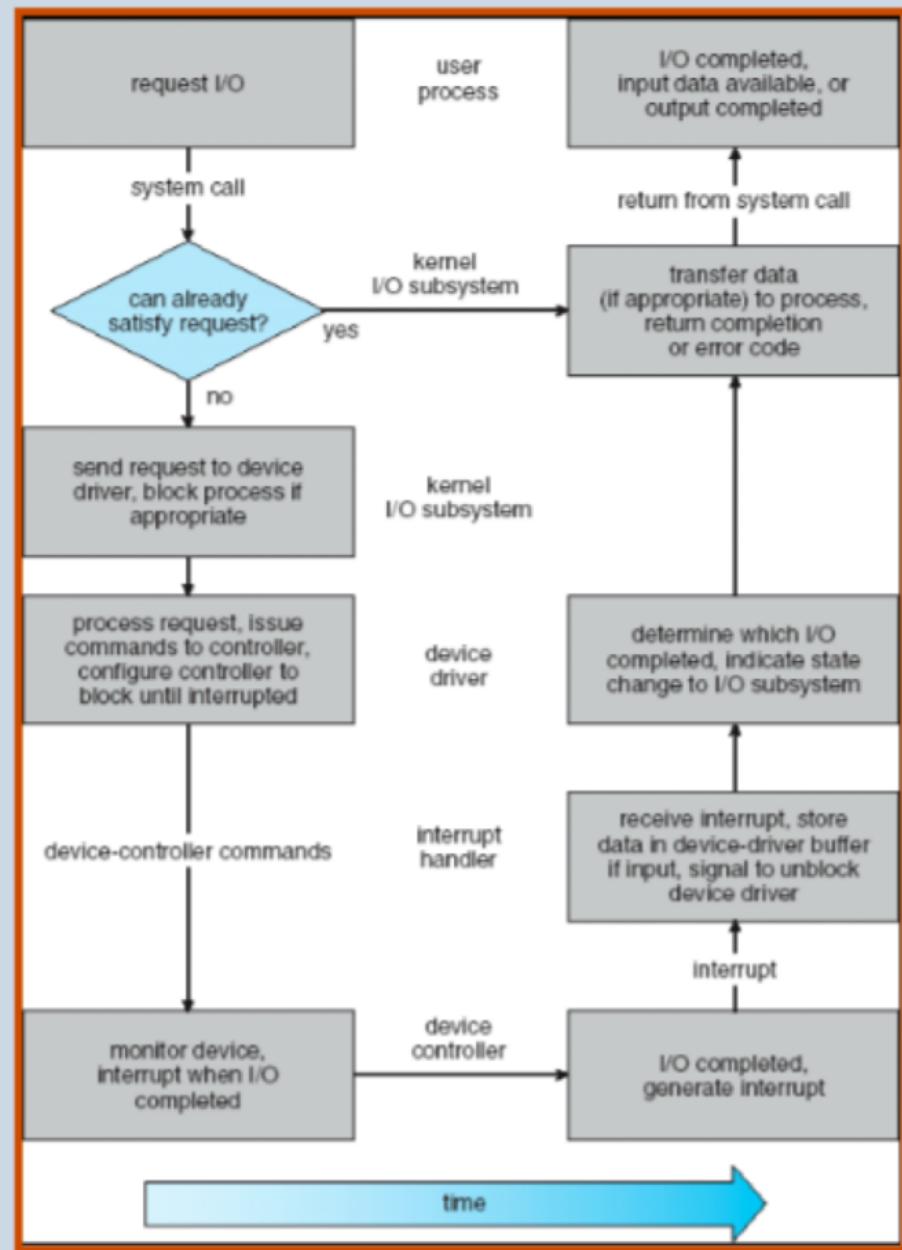
# Operatii independente de dispozitiv (2)

Caching: memoria tine copii ale datelor

Spooling: se mentine un buffer cu datele transmise unui dispozitiv

Rezervarea dispozitivului

# Fluxul unei operatii de I/O



# User level Software

## Device Independent Software

### Operatii independente de dispozitiv

- Priorizare pentru eficiență și securitate
- surse și consumate de curenți de I/O
- Buffering:
  - Compensarea vitezelor de transfer dintre dispozitive
  - Reservare de căile
  - Semantica de copiere pentru utilizatori

### Operatii independente de dispozitiv (2)

- Caching: memorie fizică copie de date
- Spooling: se menține un buffer cu datele transmise unui dispozitiv
- Reservarea dispozitivelor

Rezervare  
dispozitiv



## Device Drivers

Polling

Intreruperi

Direct-memory access



Implementation

Cum folosim hardware?

## Intreruperi

## Hardware

Două categorii de dispozitive: caracter și bloc

Fiecare dispozitiv are:

- un controller
- unul sau mai multe porturi
- conectată la o magistrală

Procesorul da comenzi de I/O controllerului

Controllerul gestionează dispozitivul

# API Utilizator

## Char Devices

e.g. mouse, tastatura

Acces sequential

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/character  
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**

aio\_read

aio\_write

aio\_suspend

**Window**

ReadFile

WriteFile

OVERLAPPED I/O

# User



Dispozitive de Intrare/Iesire  
SO Curs 9

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

USER level Software

Device Independent Software

Opereu independent de dispozitive

- Multe sau puține de device și adăposte
- cu baza cărora se crează de OS
- Rezolvă:

  - Comunicația directă de la software la hardware
  - Rezolvă problema de adresa
  - Semnalele și datele sunt preluate și transformate într-un format standard

Opereu independent de dispozitive

Cablu interne la rețea și de date

- Spune că suntem interfață cu datele de rețea și de date
- Rezolvă problema de adresa

Interfață operativă de OS

Device Drivers

Polling  
Intreruperi  
Direct-memory access

Cum folosim hardware?

Intreruperi

Hardware

Două categorii de dispozitive: caracteristică bloc

Reține dispozitivele:

- un controller
- un suport fizic multe porturi
- conectare la imaginea de bază

Procesorul și componenta de I/O controlărență

Controllorul gestionărea dispozitivelor



# Kernel