Sisteme de Operare

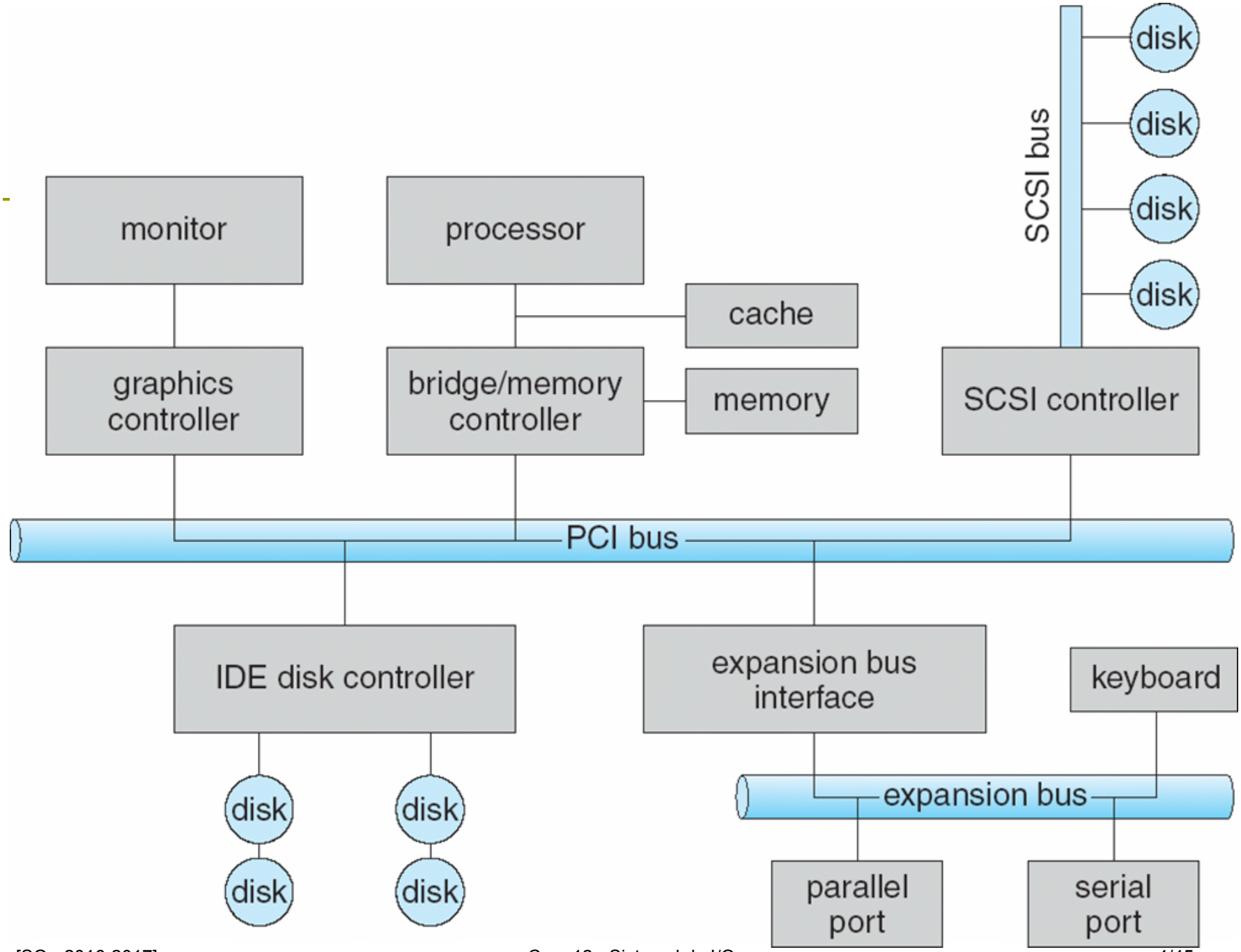
- Sistemul de I/O
- Accesul direct la memorie
- Buffer-ele sistemului de I/O

Sistemul de I/O

- generarea de comenzi către dispozitive
- tratarea întreruperilor
- tratarea erorilor posibile
- furnizarea unei interfeţe utilizator cât mai uşor de utilizat şi cu un grad cât mai ridicat de standardizare.
- este destul de dificil de realizat o generalizare din cauza multitudinii de dispozitive periferice
- dispozitivele se clasifică în funcţie de sensul în care se vehiculează informaţia :
  - dispozitive de intrare (tastatura, mouse)
  - dispozitive de ieşire (display, imprimanta).
- Sunt dispozitive care pot vehicula informaţie în ambele sensuri (discurile şi benzile magnetice, adaptoarele de reţea).

Sistemul de I/O caracteristicile perifericelor

- viteza de acces
  - variază cu câteva ordine de mărime de la un dispozitiv la altul;
- unitatea de transfer
  - poate fi caracter, octet, cuvânt, bloc sau înregistrare, în funcţie de natura dispozitivului periferic;
- reprezentarea datelor
  - datele pot fi codificate în diverse moduri, depinzând de diferite medii de intrare/ieşire;
- operaţiile posibile cu un anumit dispozitiv I/O
  - sunt determinate în principiu de sarcina pe care dispozitivul o îndeplineşte în cadrul sistemului.
  - Operaţiile de citire/scriere aplicabile în conjuncţie cu anumite dispozitive nu au sens pentru altele.
- condiţiile de eroare
  - au diferite cauze (de la lipsa hârtiei la eroarea codului de control pentru un transfer de date) şi implicit modalităţile de tratare sunt diferite.


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

Obiectivele proiectarii unui sistem de I/O

- independenţa faţă de codul de caractere
  - sistemul de I/O trebuie să recunoască diversele coduri de caractere utilizate de dispozitivele periferice şi să prezinte datele într-un format standard.
- independenţa faţă de dispozitivele periferice
  - posibilitatea scrierii programelor, astfel încât să nu necesite modificări ale codului atunci când se modifică tipul dispozitivului periferic faţă de cel prevăzut iniţial, lucru ce presupune furnizarea unor operaţii a căror sintaxă şi semantică să fie cât mai asemănătoare pentru o clasă cât mai mare de dispozitive periferice.
  - Aici apare şi denumirea uniformă a dispozitivelor periferice din cazul UNIX şi Windows, unde fiecare dispozitiv are asociat un fişier, dispozitivele fiind denumite prin intermediul numelui fişierului asociat.
- eficienţa operaţiilor
  - dispozitivele periferice pot introduce penalizări sub aspectul timpului de acces, datorate atât diferenţei mari dintre viteza de calcul a unităţii centrale şi cea de transfer a datelor precum şi dintre viteza de transfer a datelor şi viteza ansamblurilor mecanice mobile ce intră în componenţa multora dintre dispozitivele periferice.
  - au apărut mecanisme care să conducă la creşterea eficienţei (DMA, spooling etc).

Evoluția sistemului de I/O

- Procesorul controla direct dispozitivul periferic
- controller-ele de I/O:
  - procesorul utilizează mecanismul programmed I/O fără întreruperi
  - procesorul trebuie să gestioneze detaliile lucrului cu dispozitivul de I/O
- controller-ele sau modulele de I/O cu întreruperi:
  - procesorul nu mai pierde timp aşteptând terminarea operaţiei de I/O
- apariţia DMA (Direct Memory Access)
  - blocurile de date sunt mutate direct în memorie fără implicarea procesorului
  - procesorul este implicat numai la începutul şi sfârşitul operaţiei
- modulul de I/O este un procesor separat
  - apar canalele de I/O
  - este posibil accesul la memoria principală pentru instrucţiuni
- procesoarele de I/O cu propria memorie
  - este un computer în adevăratul sens al cuvântului
  - pot fi controlate un set mare de dispozitive de I/O
  - o utilizare frecventă este controlul comunicaţiei cu terminalele interactive

Structura sistemului de I/O

- Structura ierarhică
  - nivelurile sunt caracterizate de nivelul de complexitate, timp de acces, nivelul de abstractizare
  - nivelurile inferioare pot lucra la intervale de timp de ordinul nanosecundelor
- Structura logică
  - dispozitive I/O logice – toate dispozitivele sunt privite ca resurse logice (permit operaţii open, read, write)
  - Dispozitivele I/O – operaţiile şi datele sunt convertite în secvenţe de instrucţiuni I/O
    - Se pot folosi buffer-e pentru a creşte viteza de lucru
  - Planificare şi control – creează şi gestionează cozile de aşteptare pentru operaţiile I/O şi realizează planificarea acestor operaţii
    - la acest nivel sunt gestionate întreruperile

Structura hardware a sistemului de I/O

- Dispozitivele de I/O se pot clasifica în două categorii:
  - dispozitive bloc(discul):
    - stochează informaţia sub forma unor blocuri de dimensiune fixă, fiecare având asociată o adresă cu ajutorul căreia poate fi accesat individual.
  - dispozitive caracter(imprimanta, mouse, terminale, adaptoare de reţea):
  - lucrează cu şiruri de caractere cărora nu le conferă o structură pe blocuri;
  - nu pot fi adresate individual şi nu pot constitui obiectul unor operaţii de căutare.

Structura hardware a sistemului de I/O

- unităţile de bandă
  - nu se pot implementa în mod eficient operaţii de acces aleator, deşi sunt structurate bloc şi permit operaţii de căutare.
- ceasul de timp real
  - nu poate fi încadrat în nici una din categoriile de mai sus
  - are sarcina de a genera întreruperi la intervale de timp bine stabilite.

Controller

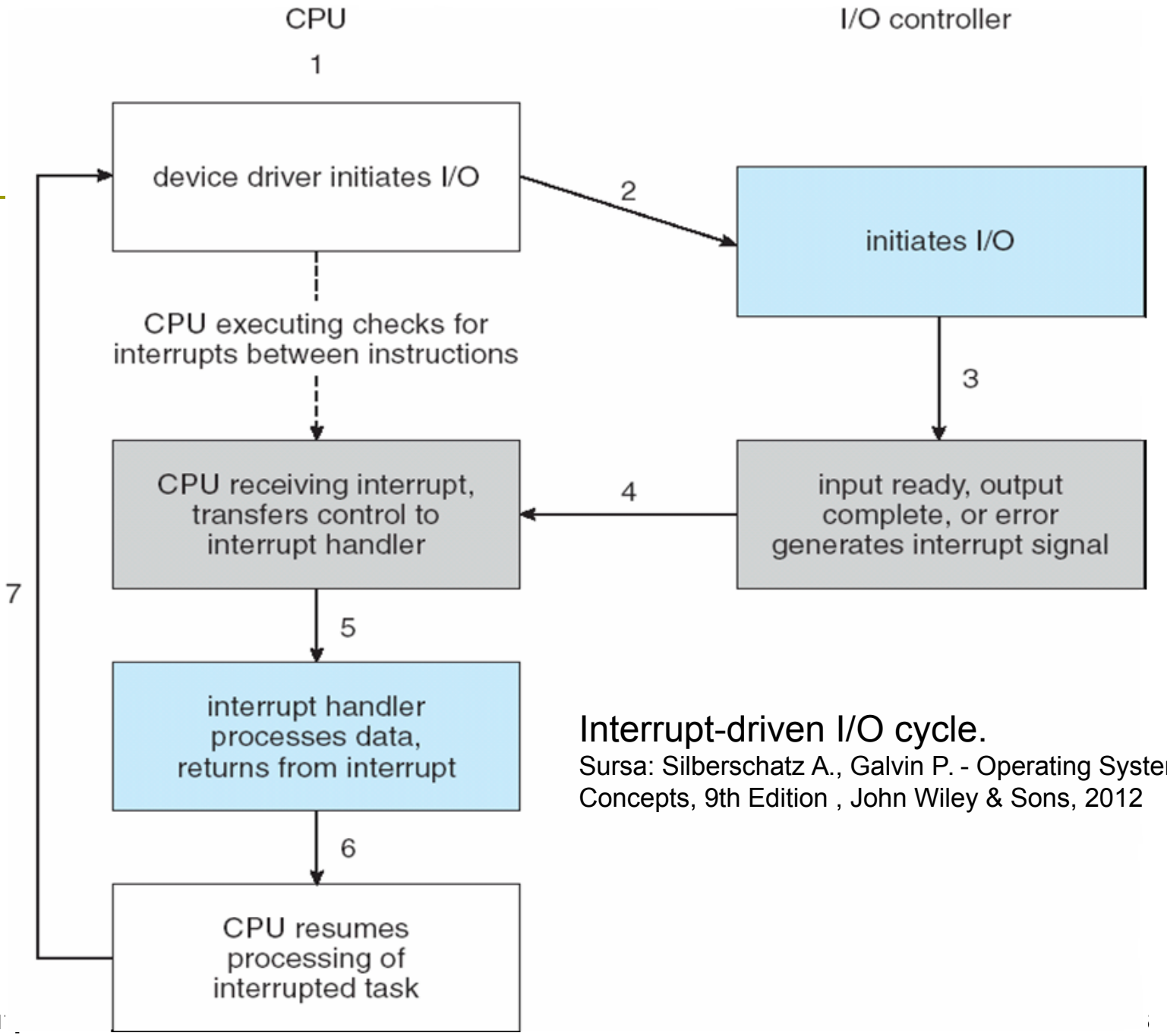
- Dispozitivele de I/O sunt formate dintr-o componentă mecanică şi una electronică numită controller.
- Un controller poate gestiona mai multe dispozitive identice.
- Distincţia între controller şi dispozitivul propriu-zis este necesară, deoarece sistemul de operare interacţionează cu controller-ul.

Controller

- operaţia de citire de pe un disc magnetic:
  - controller-ul este cel care poziţionează capetele şi citeşte de pe disc un şir de biţi care cuprinde:
    - un antet ce conţine informaţiile înscrise la momentul formatării discului: numărul cilindrului, al sectorului, dimensiunea unui sector;
    - biţi de informaţie stocaţi în sectorul respectiv;
    - un cod de corecţie a erorilor.
  - Controller-ul asamblează bit cu bit un bloc de date căruia îi calculează suma de control, ce trebuie să fie identic cu codul citit de pe disc şi abia după aceea blocul de date respectiv este trecut în memoria principală.

Comunicația dintre controller şi unitatea centrală

- se realizează prin intermediul unor registre, care de cele mai multe ori fac parte din spaţiul de adrese de memorie (sunt mapate în memorie – memory mapped I/O).
  - registrele mapate în memorie
    - se accesează la fel ca orice locaţie de memorie, singura diferenţă fiind timpul de acces mai redus.
    - sunt utilizate de sistemul de operare pentru a înscrie parametri şi comenzi şi pentru a citi starea dispozitivului respectiv şi codurile de eroare.
- O comandă odată acceptată de controller, SO lasă dispozitivul să o execute în timp ce el planifică alte task-uri.
- După terminarea operaţiei, controller-ul generează o întrerupere, care permite SO să preia controlul şi să analizeze rezultatul.



Interrupt-driven I/O cycle.

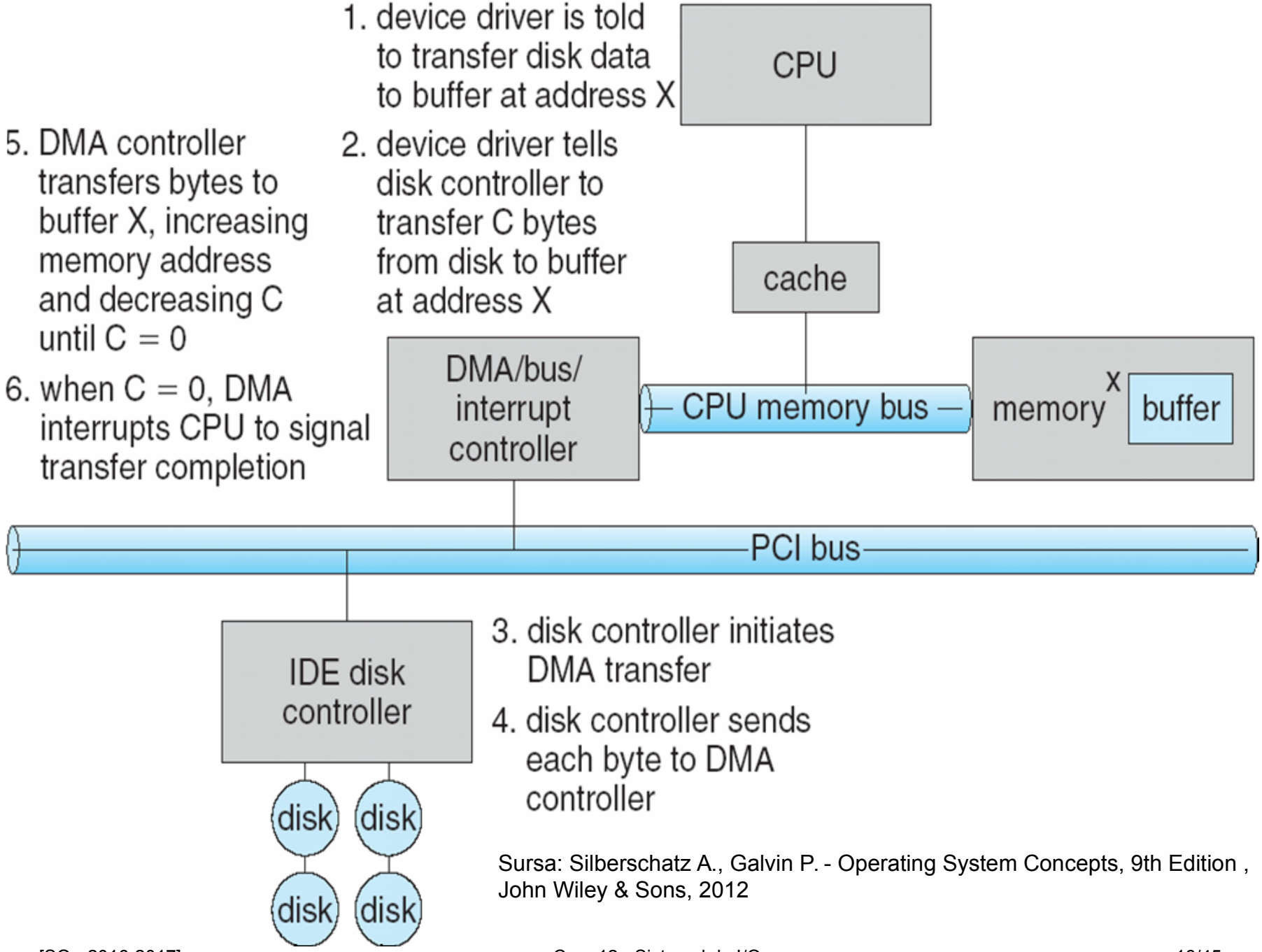
Sursa: Silberschatz A., Galvin P. - Operating System Concepts, 9th Edition , John Wiley & Sons, 2012

PC – adresele porturilor I/O (lista partiala)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

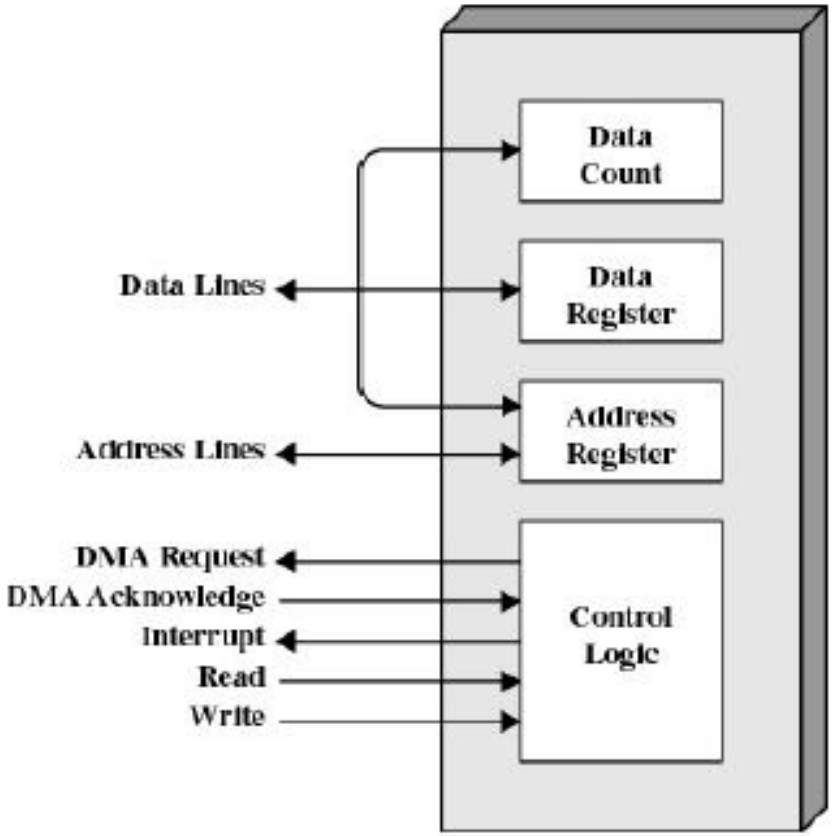
Accesul direct la memorie (Direct Memory Access – DMA)

- operaţia de copiere în memorie este efectuată de către controller şi nu de către unitatea centrală.
  - În acest mod se obţine o utilizare mai eficientă a acesteia.
  - Utilitatea acestui mecanism este justificată de necesitatea transferului unui volum mare de date.
- Fără DMA:
  - Dacă un controller de disc primeşte comanda de citire de pe disc a unui bloc (corespunzător unuia sau mai multor sectoare) precizat prin adresă, converteşte adresa într-un număr de cilindru-sector-cap.
  - Conţinutul blocului este citit bit-cu-bit în buffer-ul intern al controller-ului şi verificat dacă nu are erori, după care controller-ul semnalizează printr-o întrerupere terminarea operaţiei, urmând ca sistemul de operare să transfere cuvânt-cu-cuvânt conţinutul buffer-ului intern în memoria principală lucru ce duce la o utilizare ineficientă a tipului de lucru a unităţii centrale.
    - Acest mecanism se mai numeşte şi programmed I/O.



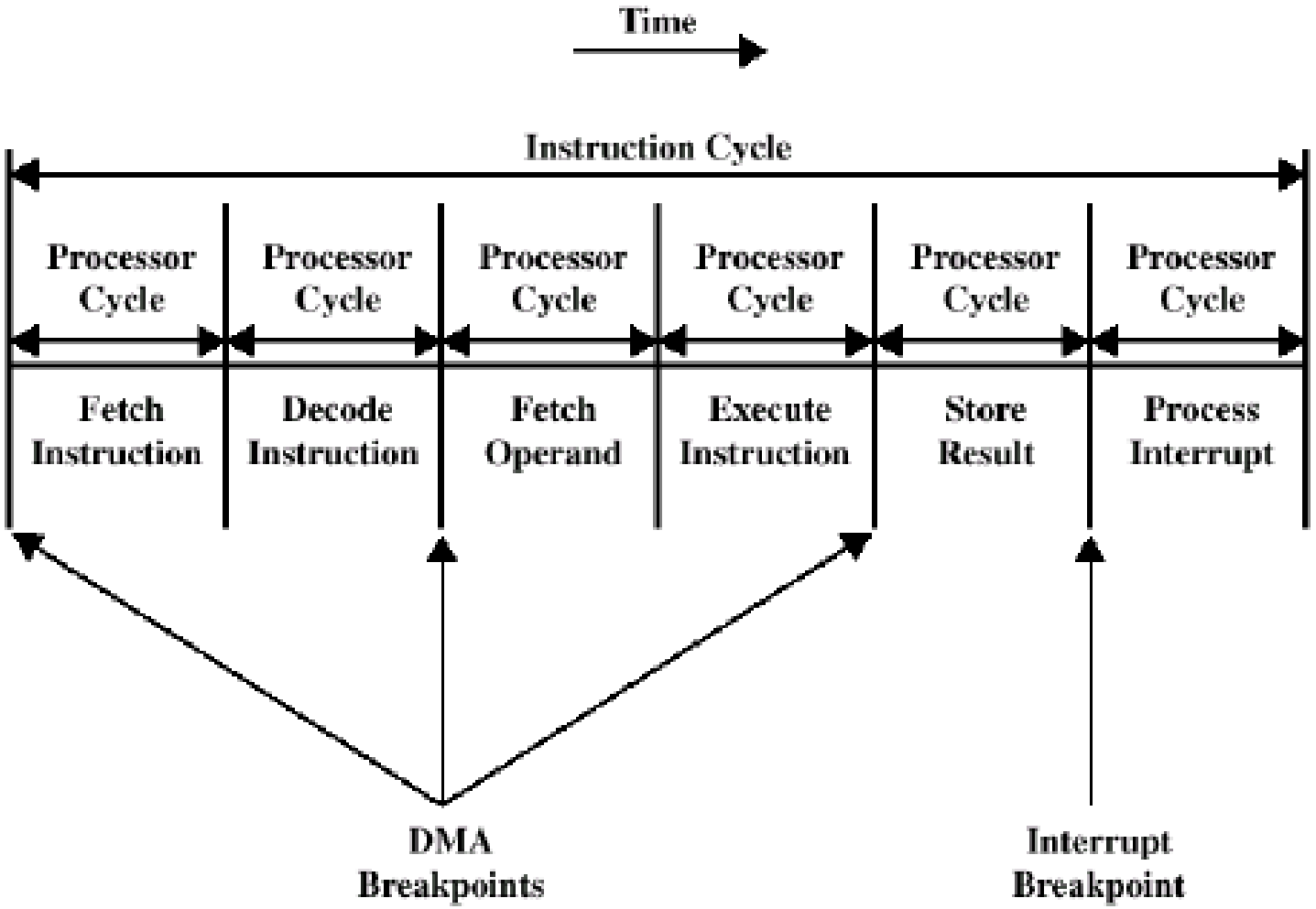
Sursa: Silberschatz A., Galvin P. - Operating System Concepts, 9th Edition , John Wiley & Sons, 2012

DMA – schema bloc



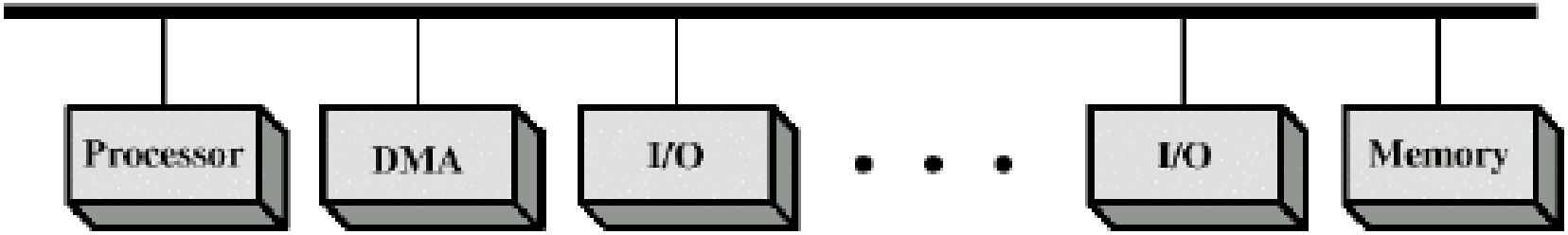
- Conţinutul blocului de date este stocat în buffer-ul intern şi verificat, controller-ul aşteaptă eliberarea magistralei sistemului şi transferă întregul bloc în memorie.

Punctele de oprire a execuției ciclului de instrucțiuni la apariția DMA şi a întreruperilor



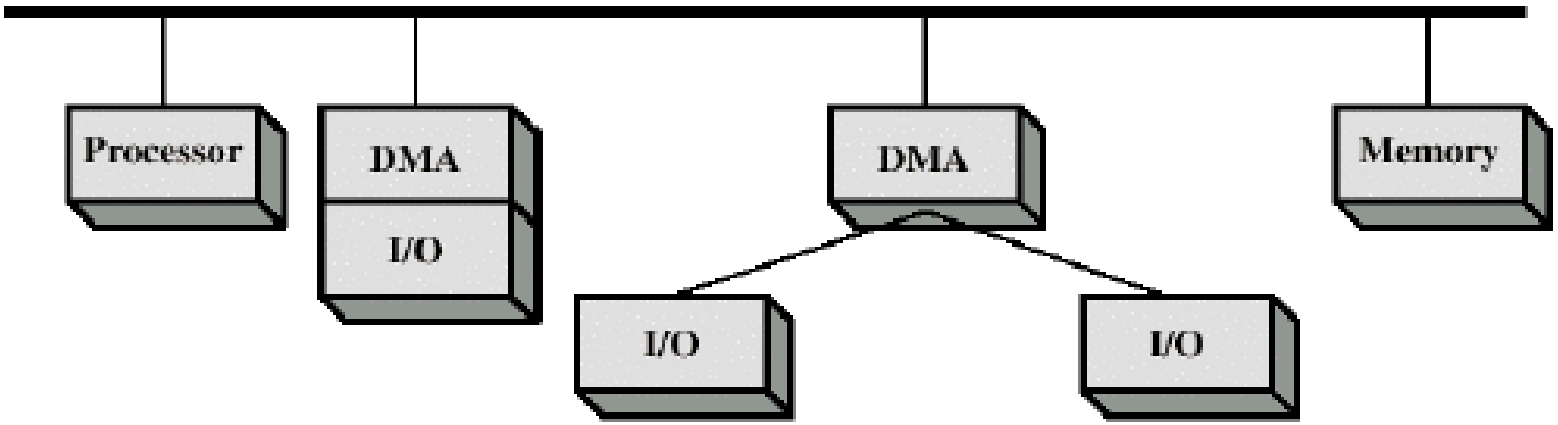
Variante de implementare ale DMA

- Single bus cu modulul DMA separat
  - toate modulele partajează aceeaşi magistrală
  - ieftină, dar ineficientă



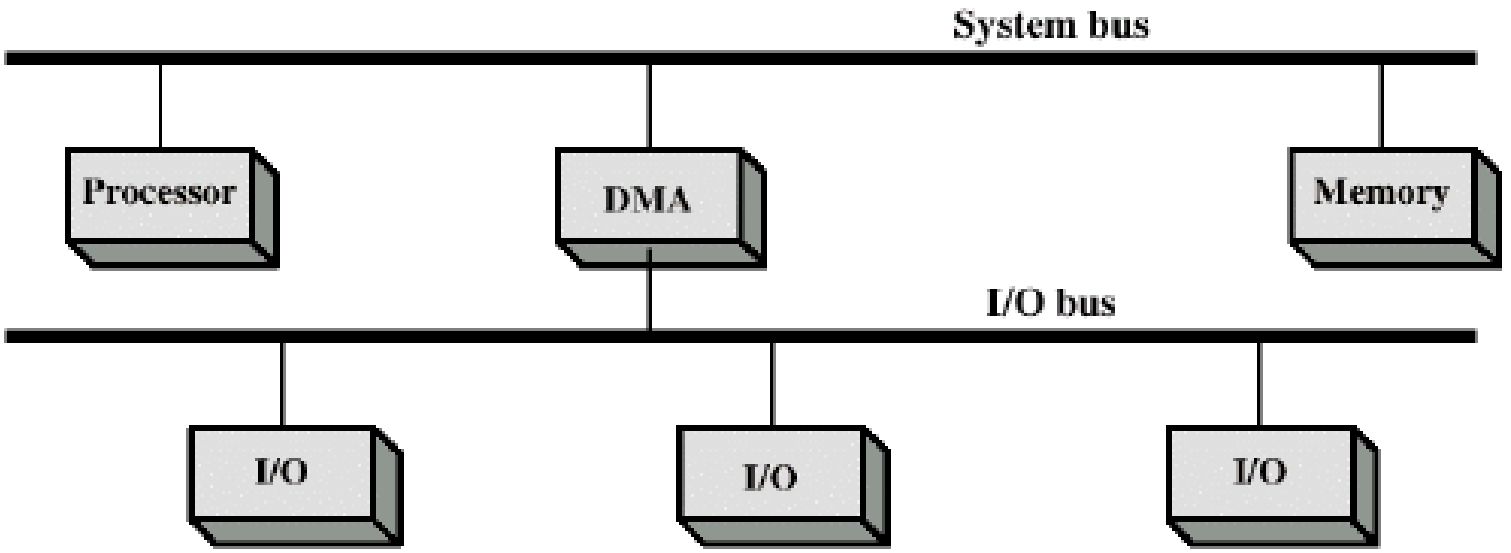
Variante de implementare ale DMA (2)

- Single bus cu modulele DMA-I/O integrate
  - există o cale de comunicaţie separată între modulul DMA şi modulele de I/O



Variante de implementare ale DMA (3)

- bus I/O separat
  - o singură interfaţă între modulele DMA şi I/O
  - uşurează expandarea configuraţiei



Utilizarea DMA

- are implicaţii şi asupra organizării informaţiei pe discul magnetic:
  - Datorită mişcării continue de rotaţie a discului, citirea unui sector se realizează numai în timpul cât el trece pe sub capul de citire/scriere.
  - După citire, controller-ul verifică datele şi le transferă în memoria principală, timp în care, în general, nu poate citi şi următorul sector care tocmai trece pe sub capul de citire/scriere (se presupune că se transferă o cantitate de date mai mare decât cea conţinută într-un sector).
  - La încheierea transferului, controller-ul trebuie să aştepte o perioadă de timp până când următorul sector ce trebuie citit va ajunge sub capul de citire/scriere.

Utilizarea DMA (2)

- Corelarea vitezei de transfer a informaţiei cu viteza de rotaţie a discului, în scopul minimizării acestei aşteptări, conduce la ideea de întreţesere (interleaving):
  - constă în plasarea a două sectoare adiacente din punct de vedere logic la o distanţă de câteva sectoare fizice, astfel încât următorul sector logic să se găsească sub capul de citire/scriere exact în momentul încheierii transferului sectorului logic anterior.
- Mecanismul DMA este aplicabil şi în cazul operaţiilor de scriere.
  - Adresa fizică şi lungimea zonei memorie furnizate ca parametri controller-ului, vor indica localizarea şi dimensiunea datelor ce trebuie preluate de controller şi scrise pe disc.

Buffer-ele sistemului de I/O

- Operaţiile de I/O din spaţiul de memorie al utilizatorului duc la apariţia următoarelor probleme:
  - paginile care păstrează data ce trebuie transferată trebuie să rămână în memorie
  - apar limitări la adresa acţiunilor sistemului de operare
  - procesele nu pot fi transferate complet în swap sau pot apare blocaje (deadlock):
    - procesele aşteaptă terminarea unei operaţii de I/O
    - sistemul de I/O aşteaptă ca procesul să fie adus din swap
- Există posibilitatea de a citi în avans unele date (read in advance), precum şi de a întârzia scrierea (se combină mai multe cereri de scriere pe disc atunci când se realizează scrierea).
- Sistemul de operare atribuie un singur buffer în memoria principală pentru operaţiile de I/O.

Buffer-ele sistemului de I/O

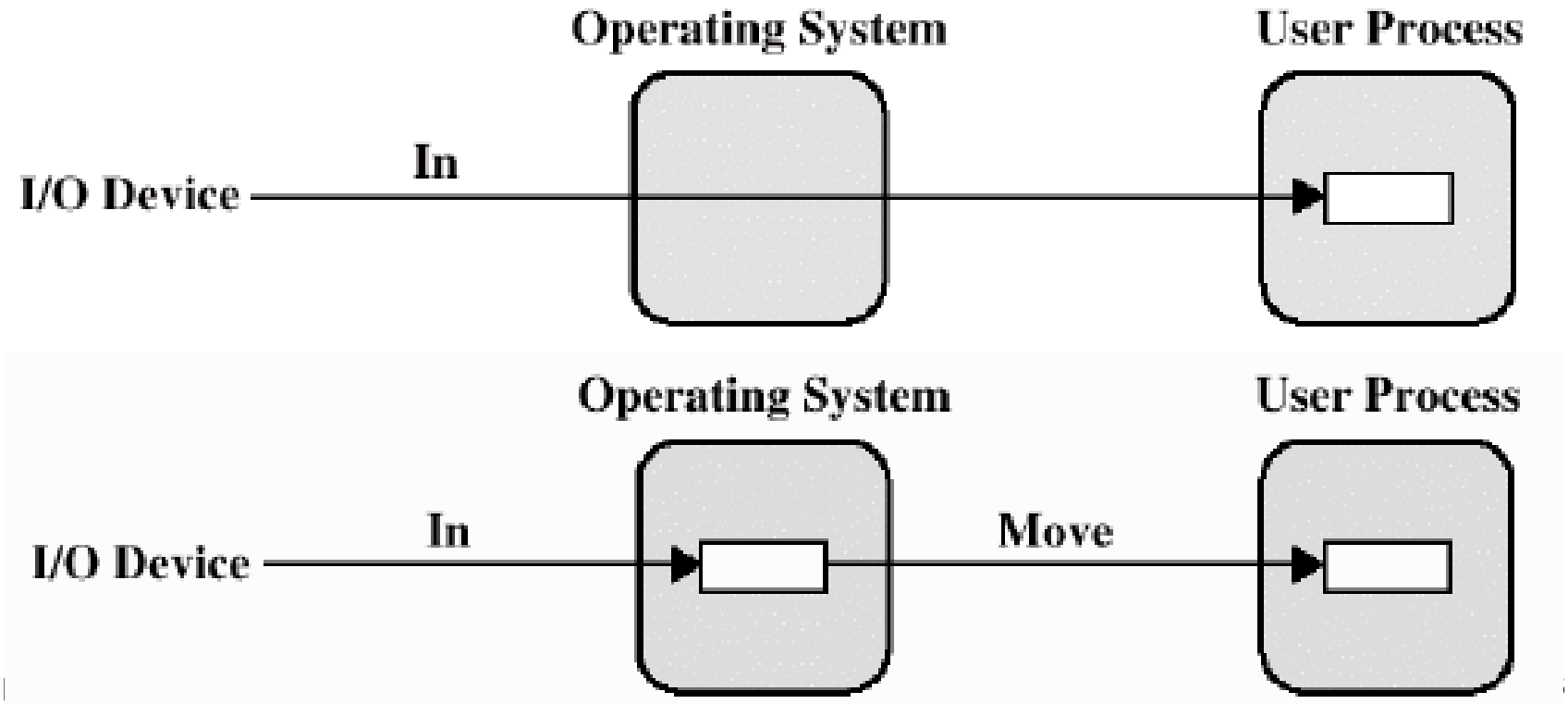
- Procesele utilizator pot procesa un bloc de date, în timp ce următorul bloc este citit.
- Datorită tehnicii de swapping, blocurile de date care trebuie să fie trecute în spaţiul de memorie al utilizatorului sunt trecute în memoria sistemului.
- Sistemul de operare păstrează informaţiile legate de atribuirea buffer-elor sistem proceselor utilizatorilor.
- Transferul de date către dispozitivele de I/O se face prin scrierea de către procese în buffer-e şi apoi are loc transferul efectiv al datelor.

Buffer-ele sistemului de I/O

- Buffer-e de tip bloc
  - transferul datelor se face în buffer,
  - blocurile sunt mutate în memorie atunci când este necesar;
  - un alt bloc este transferat în buffer.
- buffer-e de tip caracter
  - Exemplu: introducerea datelor de la terminal. Scrierea la terminal se face linie cu linie.

Implementarea buffer-elor sistemului de I/O

- Un singur buffer
  - Sistemul de operare atribuie un singur buffer în memoria principală pentru operaţiile de I/O.



Implementarea buffer-elor sistemului de I/O (2)

- Dispozitivul de I/O transferă datele în buffer-ele sistemului şi SO copie data în spaţiul de memorie al utilizatorului.
- Imediat ce un transfer este terminat se încearcă citirea în avans a următorului bloc.
- Procesele utilizator pot lucra cu un singur bloc de date, în timp ce următorul bloc este citit.
- Timpul de transfer al unui bloc:

$$\mathbf{max[C,T] + M}$$

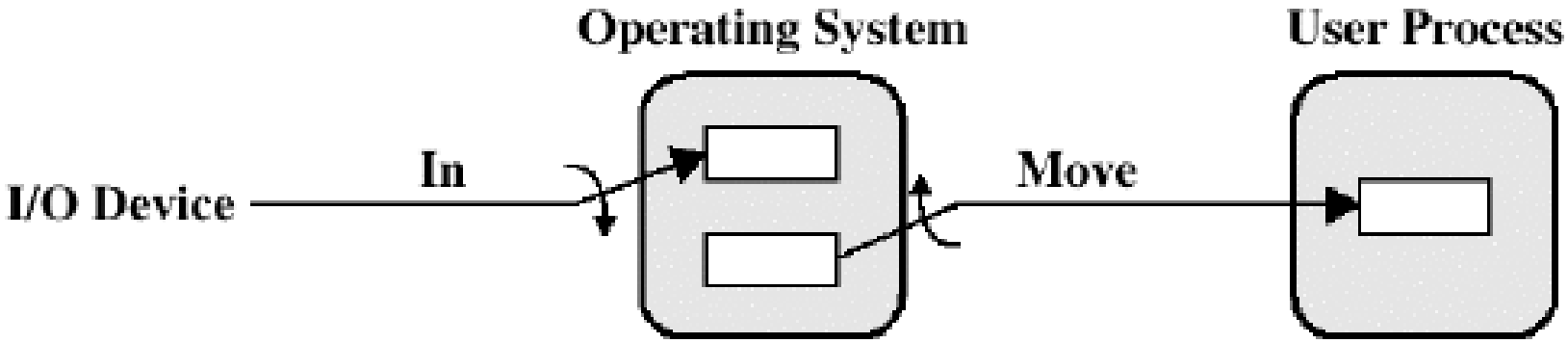
  - C = timp de calcul,
  - T = timp de realizare a operaţiei de I/O,
  - M = timpul de transfer cu buffer-ul
- Observaţie: fără buffer-e timpul este C+T

Implementarea buffer-elor sistemului de I/O (3)

- Sistemul de operare va urmări buffer-ele atribuite utilizatorilor;
- nu este de dorit trecerea în swap a unui proces care aşteaptă terminarea unei operaţii de I/O.
- Totuşi, sistemul de operare are posibilitatea de a transfera procesele în swap fără ca acest lucru să afecteze operaţiile de I/O.
- Pentru dispozitivele de tip caracter buffer-ele pot transfera biţi (bytes) sau linii.

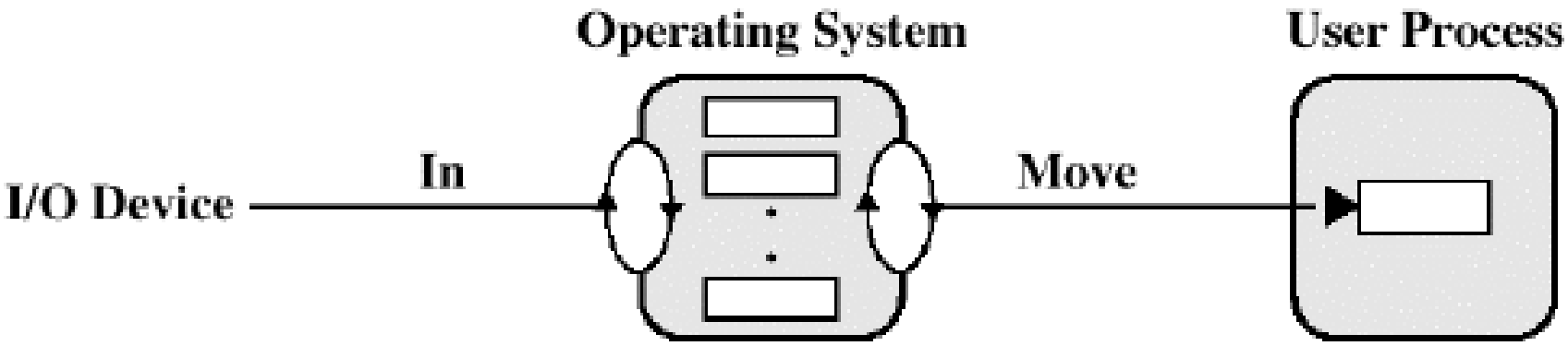
Implementarea buffer-elor sistemului de I/O (4)

- Buffer dublu:
  - Un proces poate transfera data dintr-un buffer, în timp ce sistemul de operare umple sau goleşte celălalt buffer.



Implementarea buffer-elor sistemului de I/O (5)

- Buffer circular:
  - Sunt folosite mai mult de două buffer-e, fiecare buffer constituind o unitate de buffer-e circulare.



Structura software a sistemului de I/O

- Software-ul destinat gestiunii dispozitivelor periferice este structurat pe patru niveluri:
  - rutinele de tratare a întreruperilor;
  - drivere-ele asociate dispozitivelor periferice;
  - programe sistem independente de dispozitive;
  - primitive de nivel utilizator.

Rutinele de tratare a întreruperilor

- Rolul rutinelor de tratare a întreruperilor
  - identificarea sursei întreruperii (adică dispozitivul care a generat-o)
  - de a reiniţializa linia de întrerupere respectivă
  - memorarea stării dispozitivului (în cazul în care aceasta va fi necesară ulterior)
  - ”trezirea” (printr-o operaţie signal) procesului care a iniţiat operaţia de I/O.

Driver-e

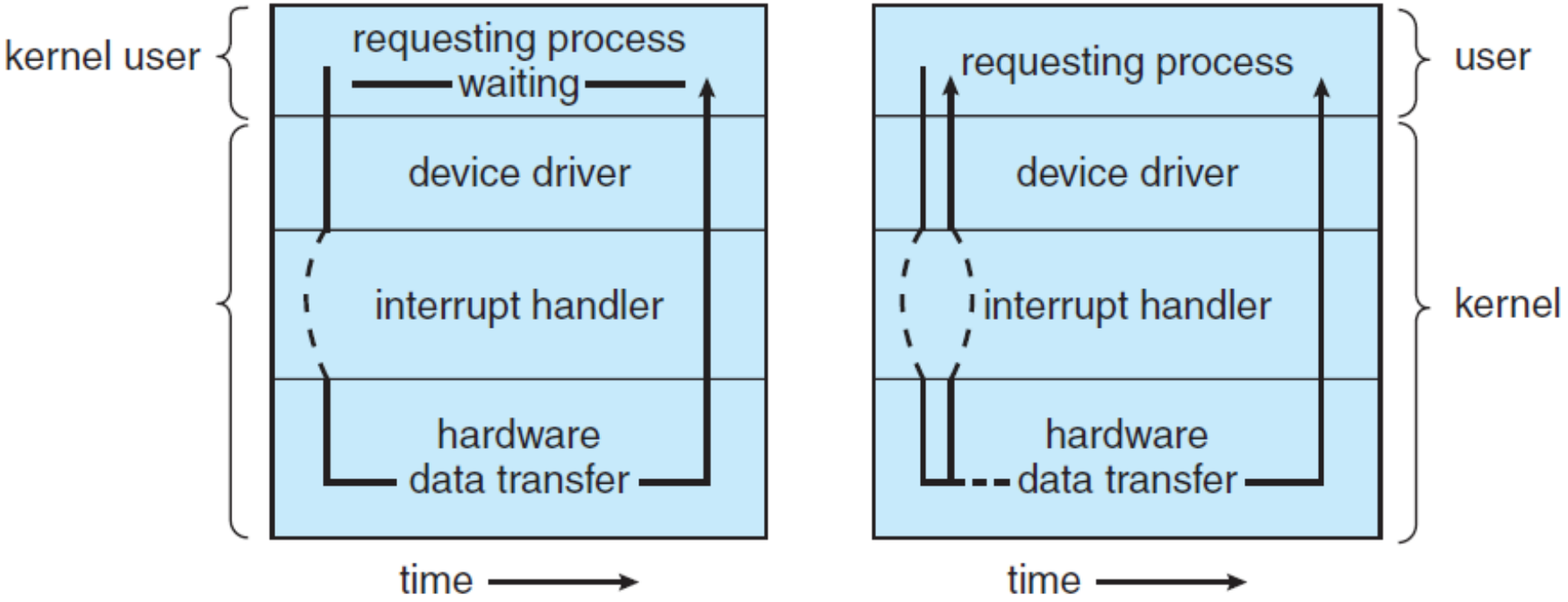
- Un driver acceptă cereri la nivelul software superior şi le transpune în comenzi pe care le transmite controller-ului, înscriind valori corespunzătoare în registrele acestuia din urmă.
- Driver-ele înglobează în totalitate acea parte a codului care este dependentă de dispozitivele periferice asociate, fiind totuşi capabile să gestioneze mai multe tipuri de dispozitive periferice înrudite.

Primitive de nivel utilizator

- au rolul de a transfera parametrii lor apelurilor sistem pe care le iniţiază, iar uneori oferă posibilitatea formatării datelor de intrare/ieşire
- pot lucra în mod sincron sau asincron
  - Sincron:
    - returnează parametrii numai după ce operaţia de I/O a fost realizată efectiv, fapt care o recomandă pentru a fi utilizată în cazul operaţiilor cu durată redusă sau care poate fi estimată.

Primitive de nivel utilizator

- Asincron :
  - are doar rolul de iniţiere a operaţiei, ea returnând imediat un cod de eroare în cazul în care operaţia nu poate fi efectuată sau 0 în caz de succes.
  - un proces îşi poate continua execuţia în paralel cu efectuarea operaţiei, testând periodic starea de evoluţie a acesteia.
  - Momentul în care operaţia se încheie este marcat printr-o notificare (procedură definită de utilizator şi asociată unui eveniment) pe care sistemul de operare o lansează automat.
  - O problemă care apare la folosirea acestui tip de primitive este disponibilitatea buffer-ului ce defineşte zona de memorie în care se află datele ce urmează a fi transferate: procesul iniţiator trebuie să evite citirea/scrierea conţinutului acestei zone atât timp cât operaţia nu a fost terminată, sarcină ce revine, în general, programatorului.
  - Acest tip de primitive sunt folosite în cazul operaţiilor cu o durată mare sau greu de estimat.



Sincron

Asincron

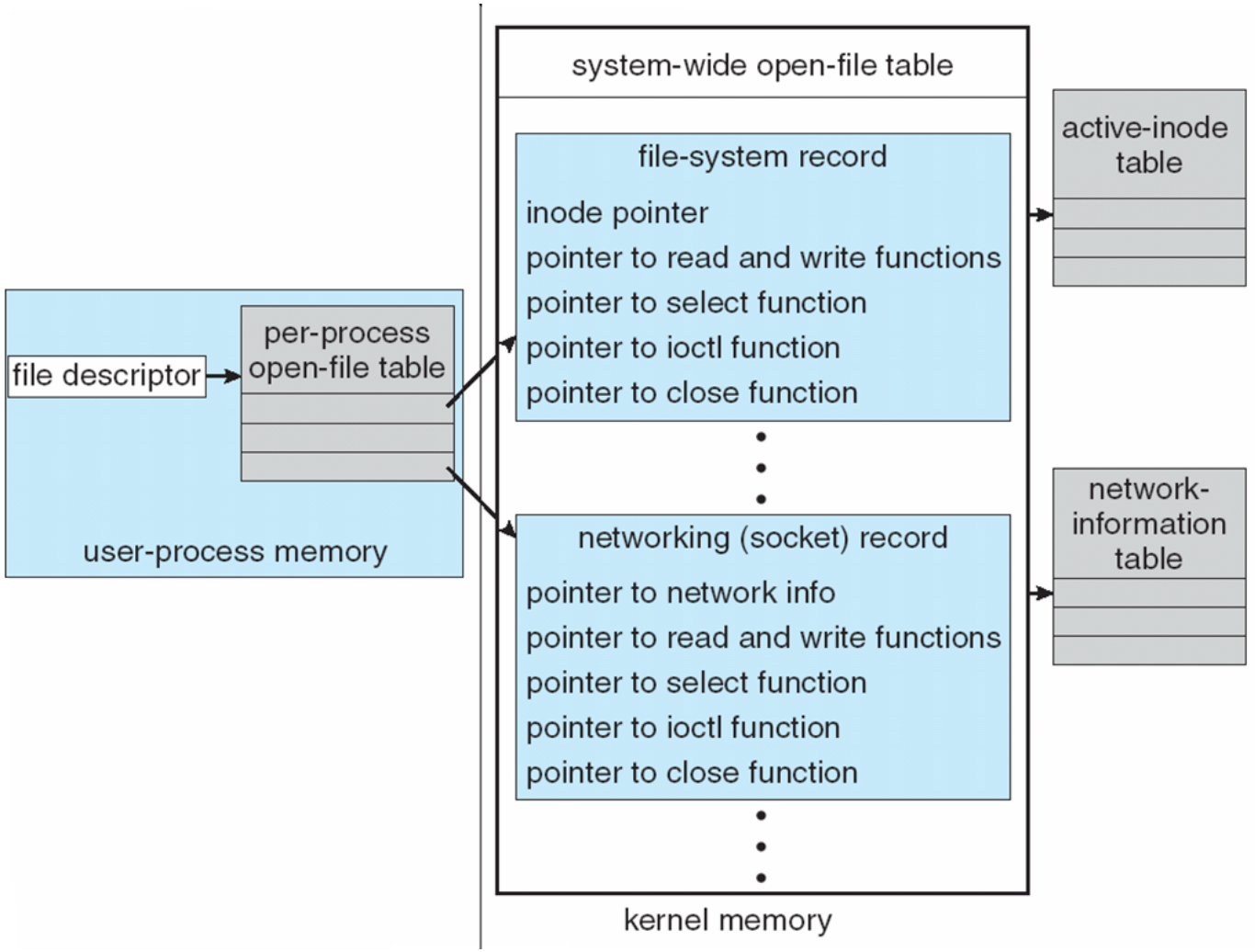
Spooling
(Simoultaneous Peripheral Operation On-Line)

- constă în stocarea informaţiei destinate transferului pe un mediu intermediar (de regulă, pe disc) şi deblocarea procesului iniţiator.
- Dispozitivele periferice dedicate au dus la apariţia unor procese specializate în gestionarea cererilor pe care alte procese din sistem le formulează către dispozitivul respectiv şi a unei zone (pe disc) reprezentând mediul intermediar de stocare a datelor care fac obiectul transferului.

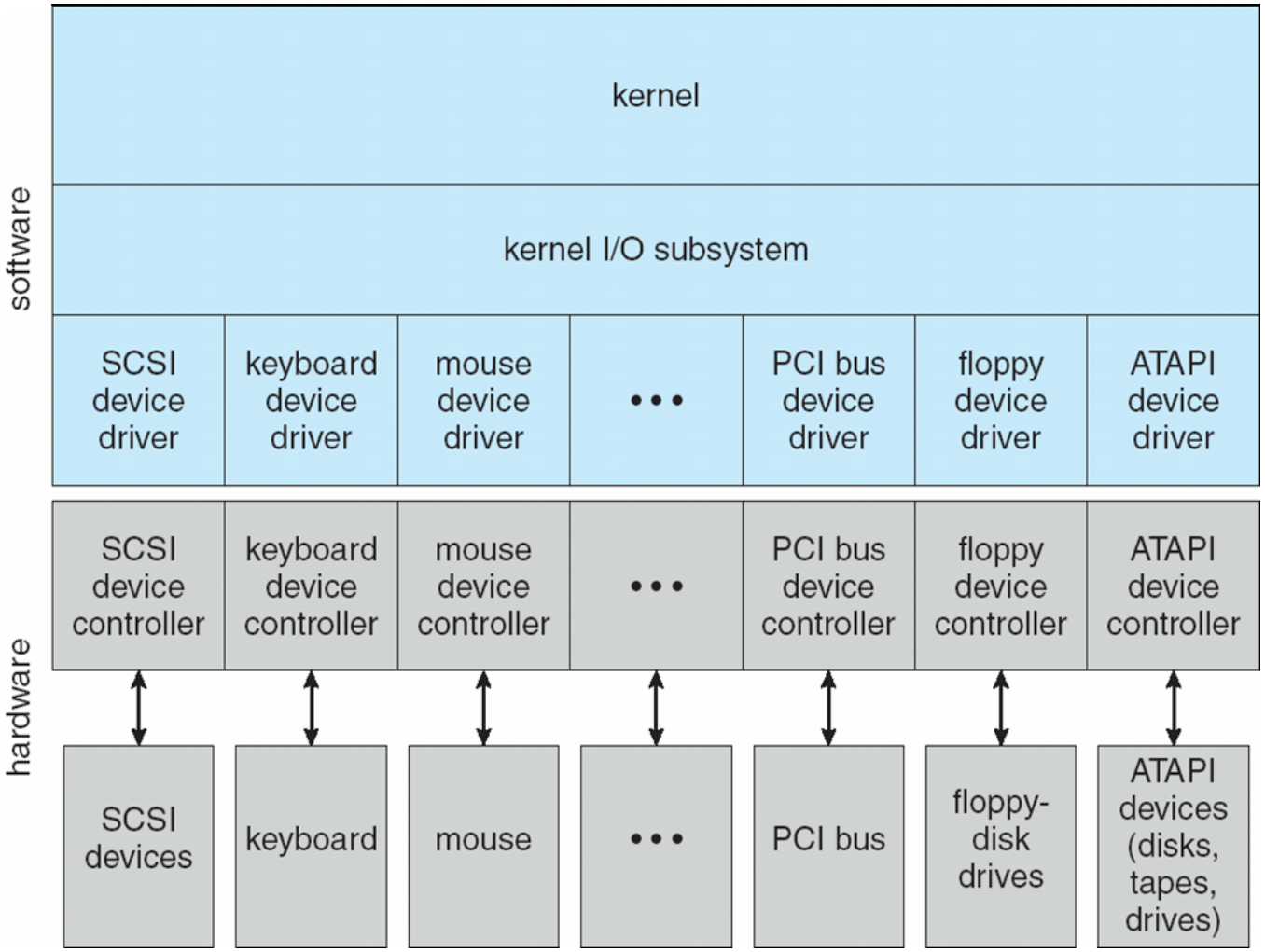
Spooling
(Simoultaneous Peripheral Operation On-Line)

- Exemplu - Imprimanta
  - se realizează prin stocarea într-un fişier a textului pe care procesul doreşte să-l tipărească, urmată de deblocare procesului.
  - Un alt proces, destinat strict gestionării cererilor către imprimantă (printer daemon), va iniţia tipărirea efectivă a textului în momentul în care imprimanta va deveni liberă.

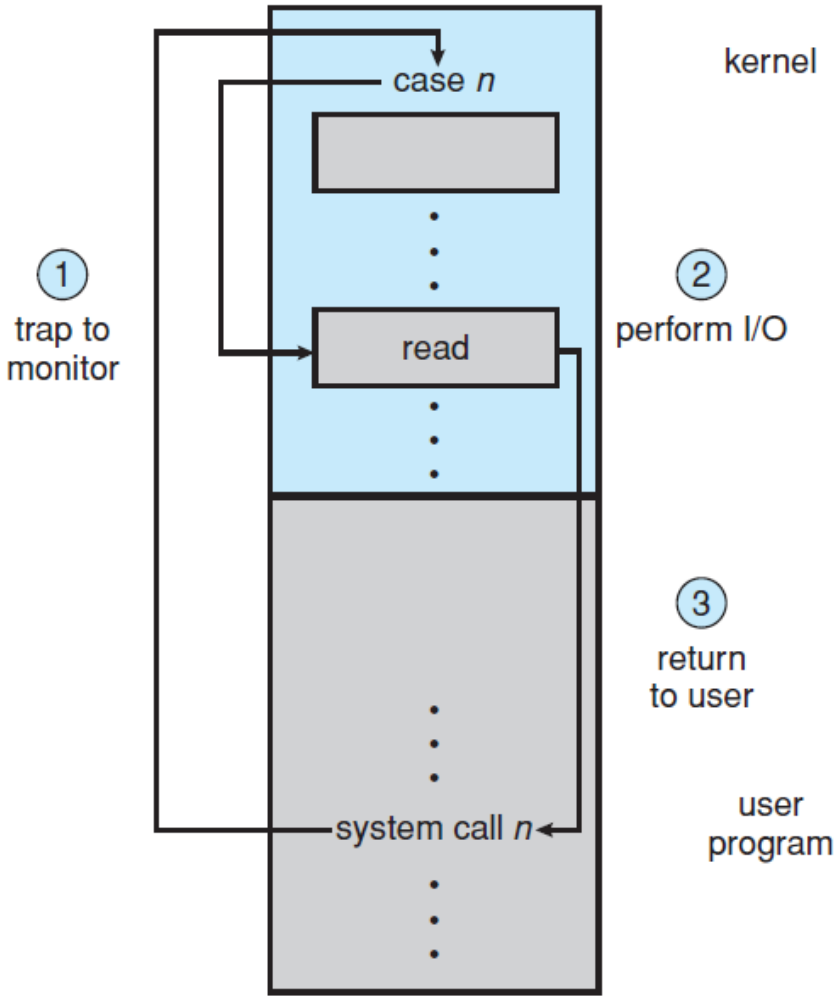
Studiu de caz
UNIX I/O Kernel subsystem

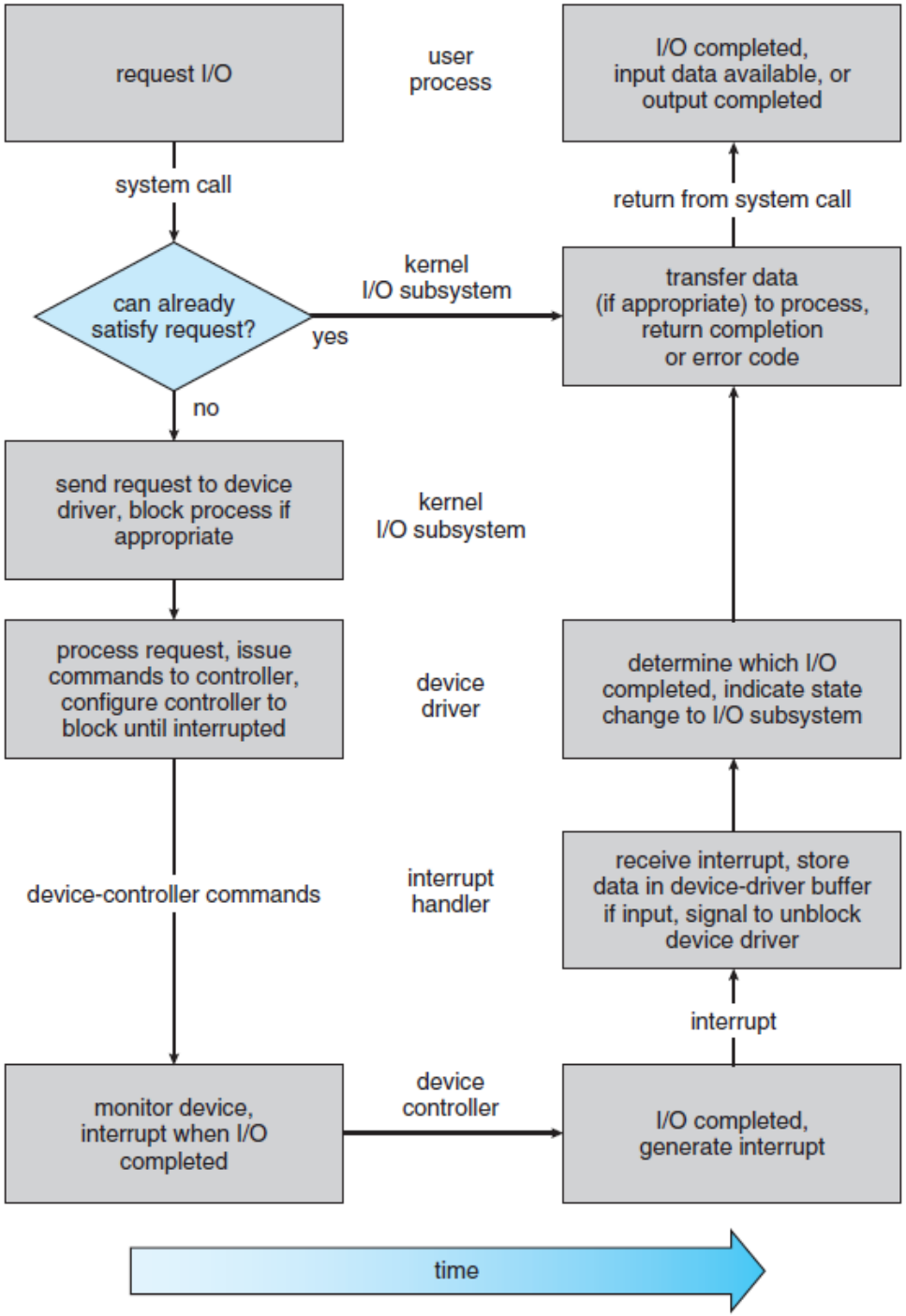


Studiu de caz
UNIX I/O Kernel subsystem



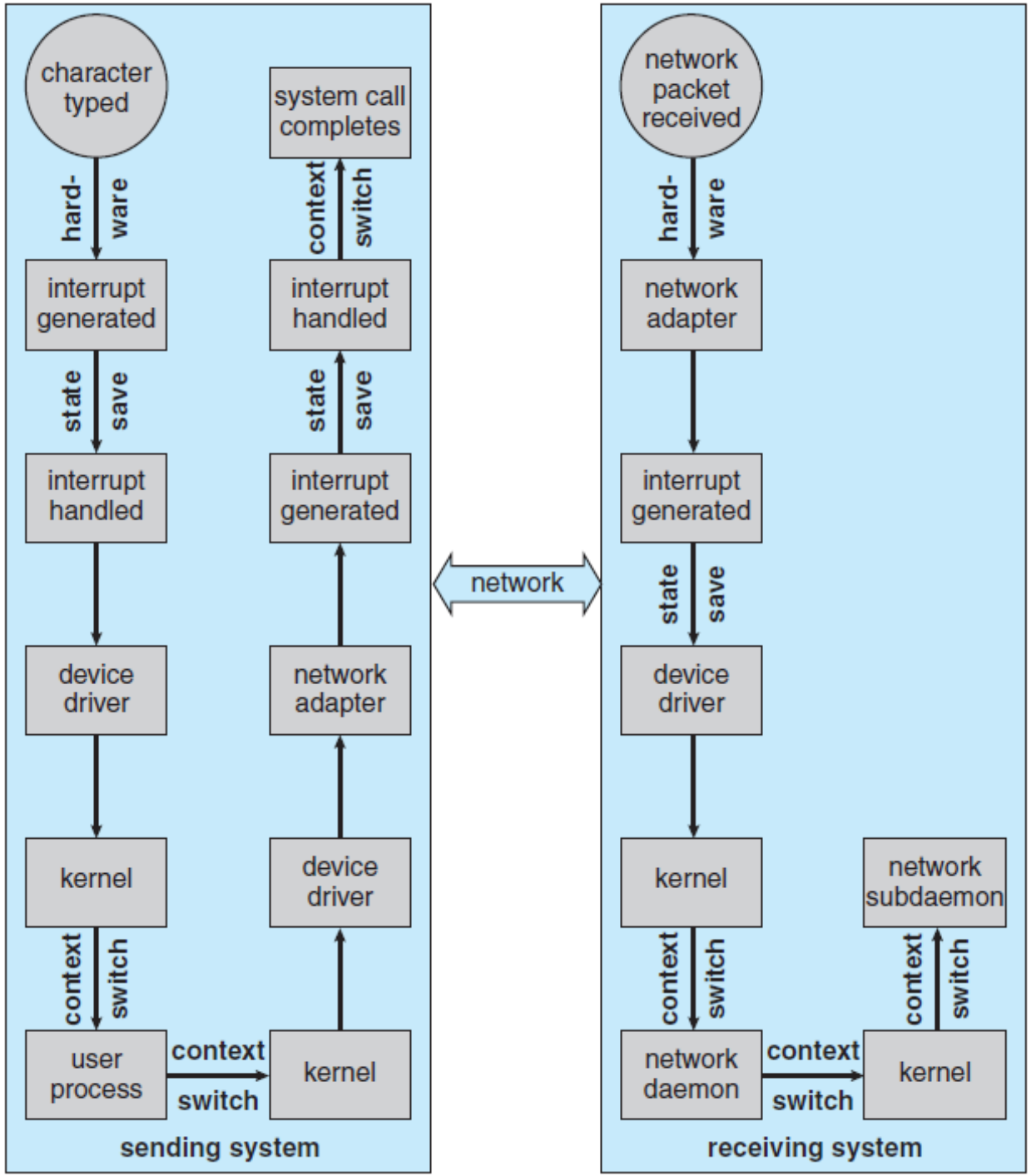
Utilizarea apelurilor sistem pentru I/O





The life cycle of an I/O request.

Sursa: Silberschatz A., Galvin P. - Operating System Concepts, 9th Edition , John Wiley & Sons, 2012



Intercomputer communications.

Sursa: Silberschatz A., Galvin P. - Operating System Concepts, 9th Edition , John Wiley & Sons, 2012

Performanța sistemului de I/O

- Sistemul de I/O are un rol important in performanta sistemului
  - generează cereri către CPU să execute codul driverelor si să planifice procesele eficient pe măsură ce acestea se blochează şi se deblochează.
  - Schimbările de context foarte dese introduc încărcare suplimentară asupra CPU si a cache-ului.
  - Sistemul de I/O dezvăluie problemele din mecanismele de tratare a întreruperilor din kernel.
  - Sistemul de I/O incarcă magistrala de date la care este conectată memoria atât în timpul transferului de date dintre controller si memoria fizică cât şi în timpul copierii informaţiilor din bufferele nucleului si spaţiul de adrese al aplicaţiilor.
  - Desi sistemele moderne pot gestiona un numar foarte mare de intrruperi pe secundă, gestiunea întreruperilor este un task costisitor. Fiecare întrerupere generează o schimbare a stării sistemului - tratarea întreruperii şi revenirea la starea anterioară.