Sisteme de Operare

Structură, componente, servicii

Cristian Vidrașcu

http://www.info.uaic.ro/~vidrascu

Cuprins

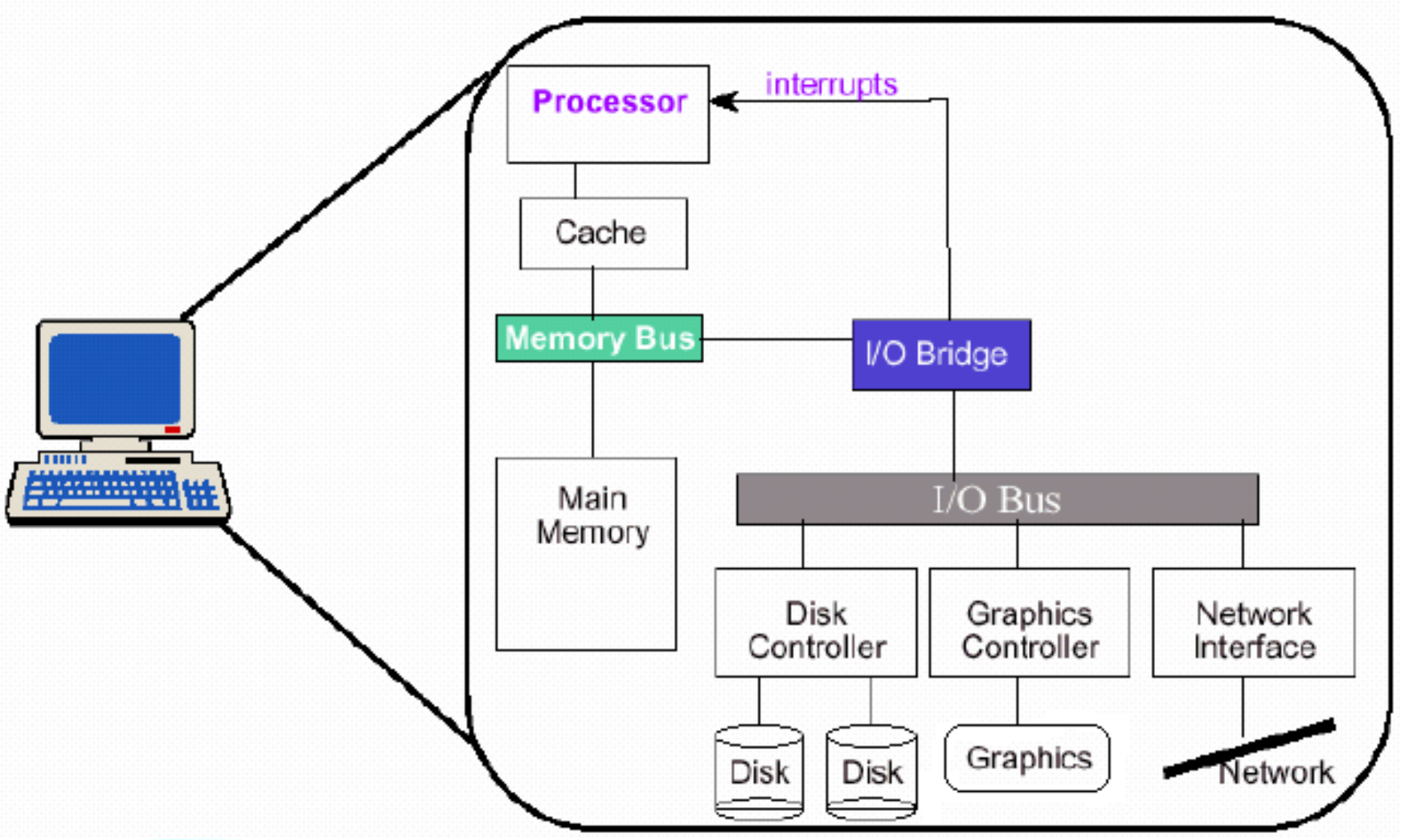
- Organizarea unui sistem de calcul
- Structura unui S.O.
- Servicii oferite de S.O.
- Abstractizări și API-uri S.O.
- Programe de sistem
- Nucleul S.O.

Preliminarii

Pentru ca operațiile CPU și I/E să se suprapună în timp (i.e., să se execute în paralel), trebuie să existe un mecanism, oferit de hardware-ul sistemului de calcul, care să permită sincronizarea operațiilor:

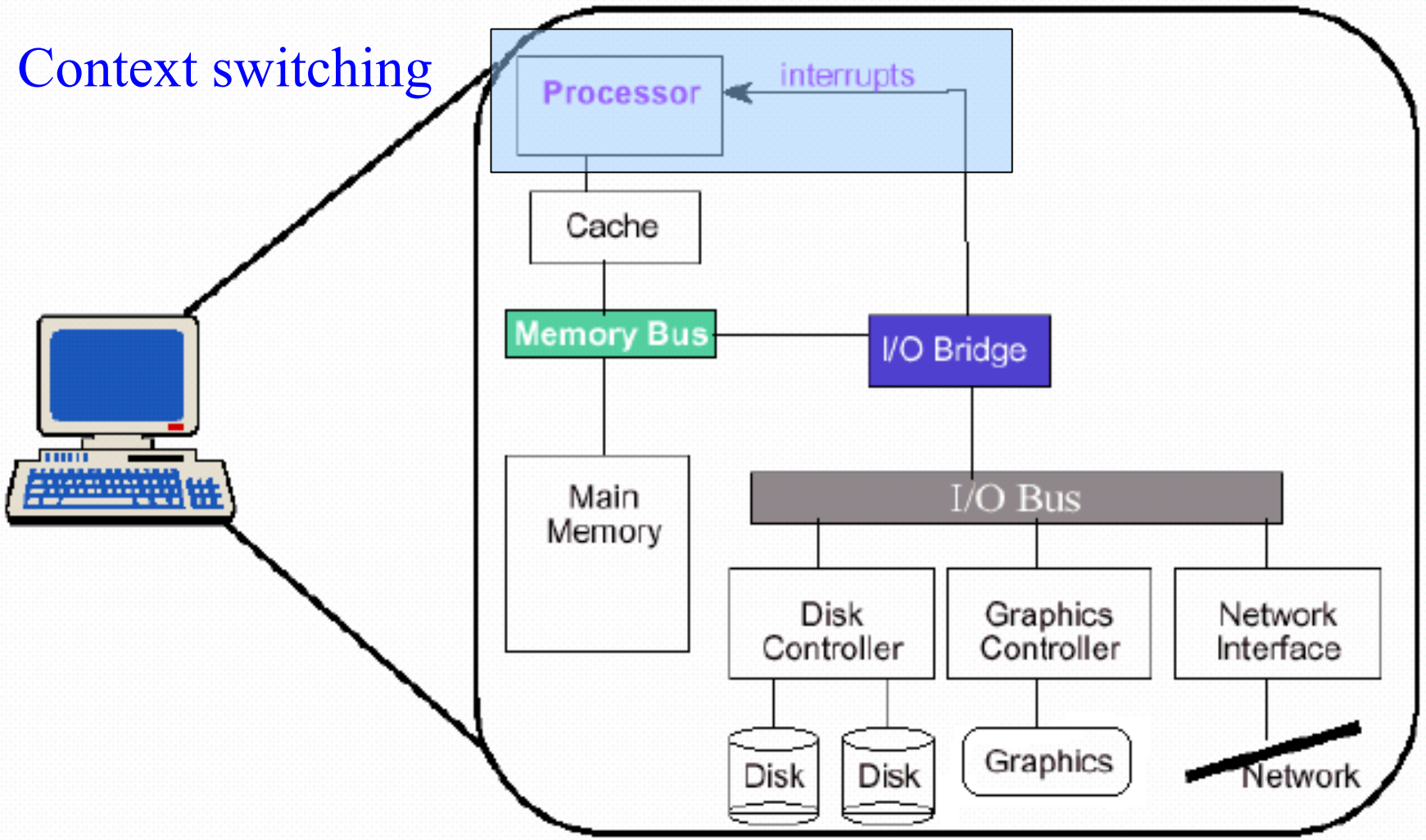
- transfer de date controlat prin întreruperi
- transfer de date prin DMA (direct memory access)

Organizarea sistemului /1a

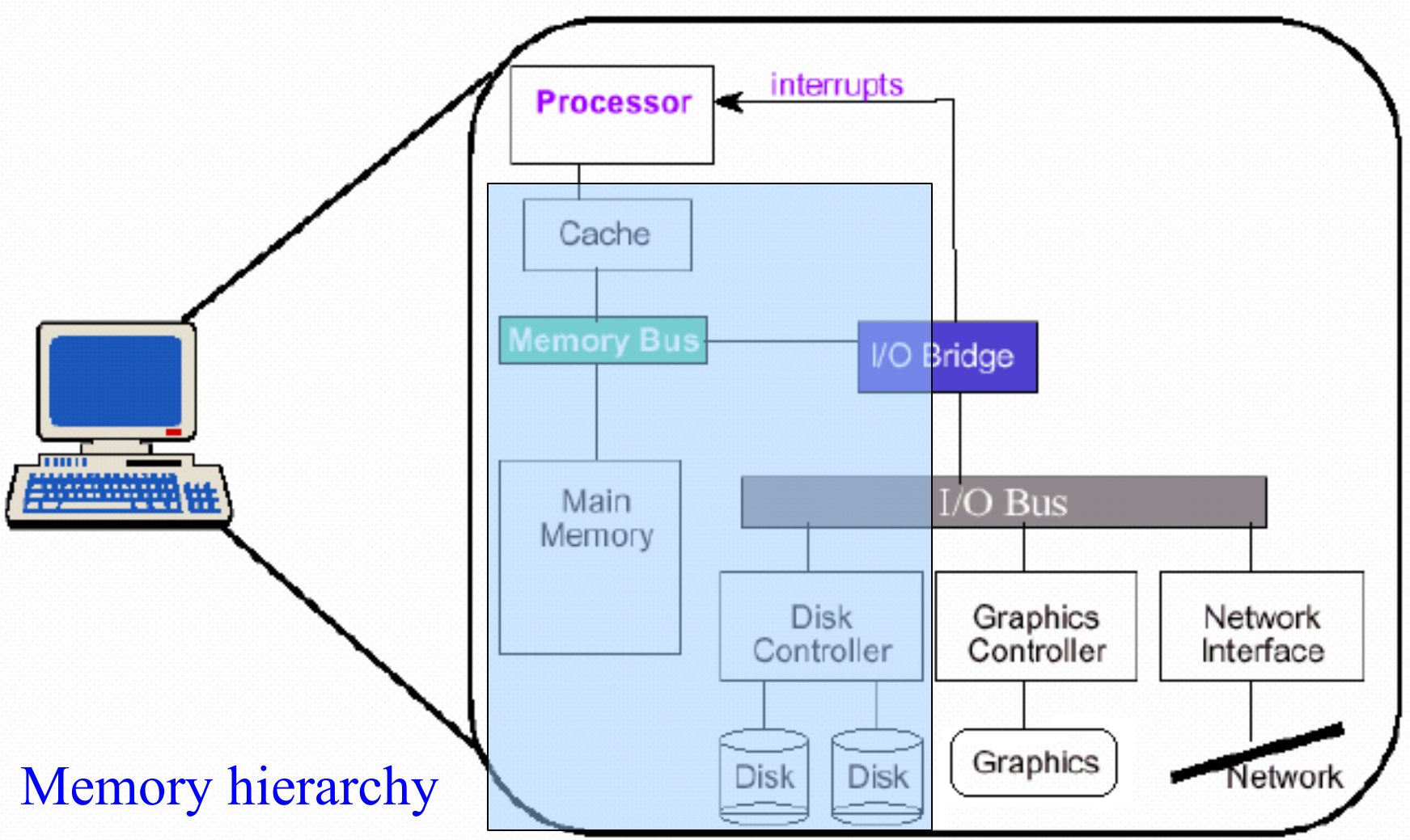


Organizarea sistemului /1b

Context switching

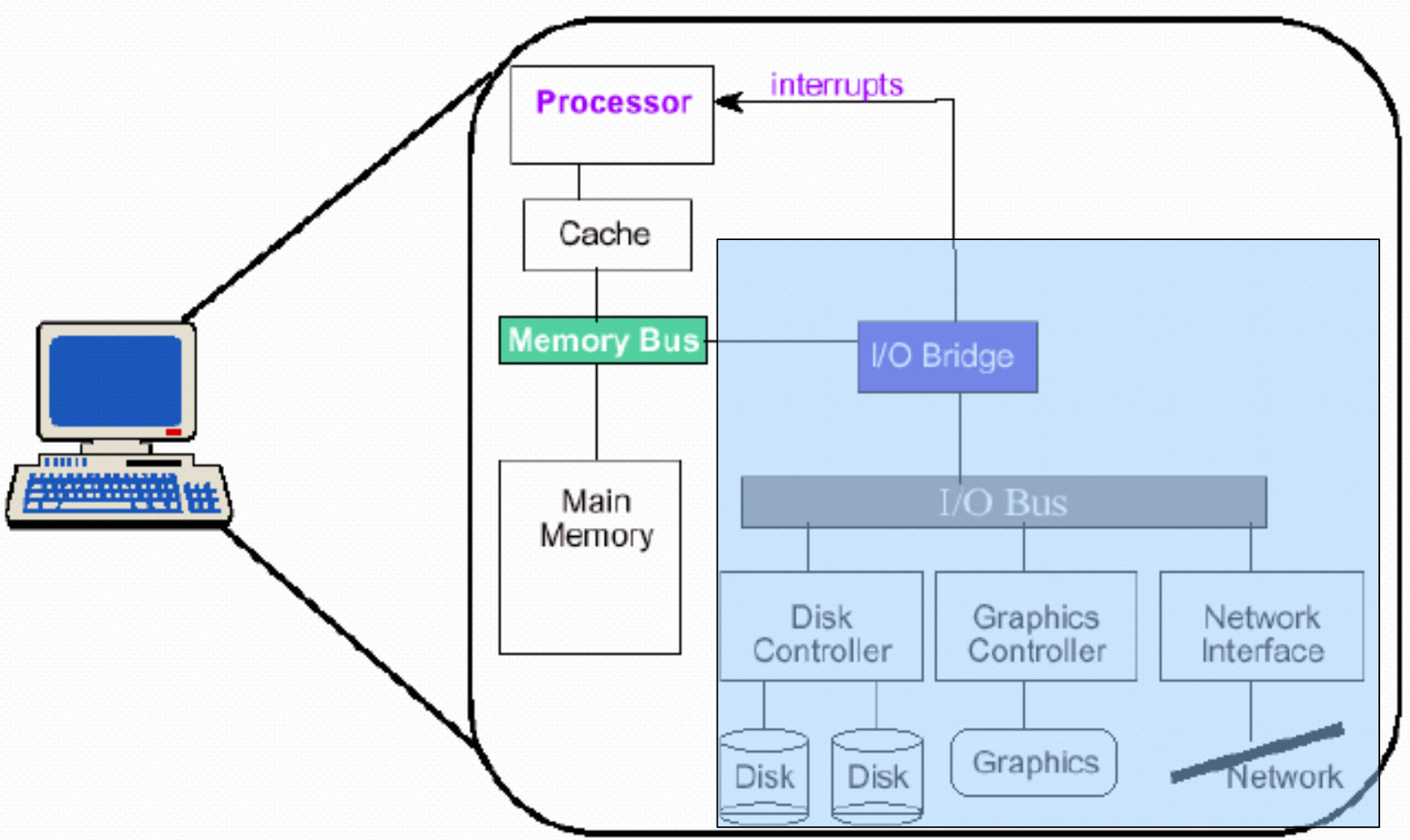


Organizarea sistemului /1c



Memory hierarchy

Organizarea sistemului /1d



DMA
I/O

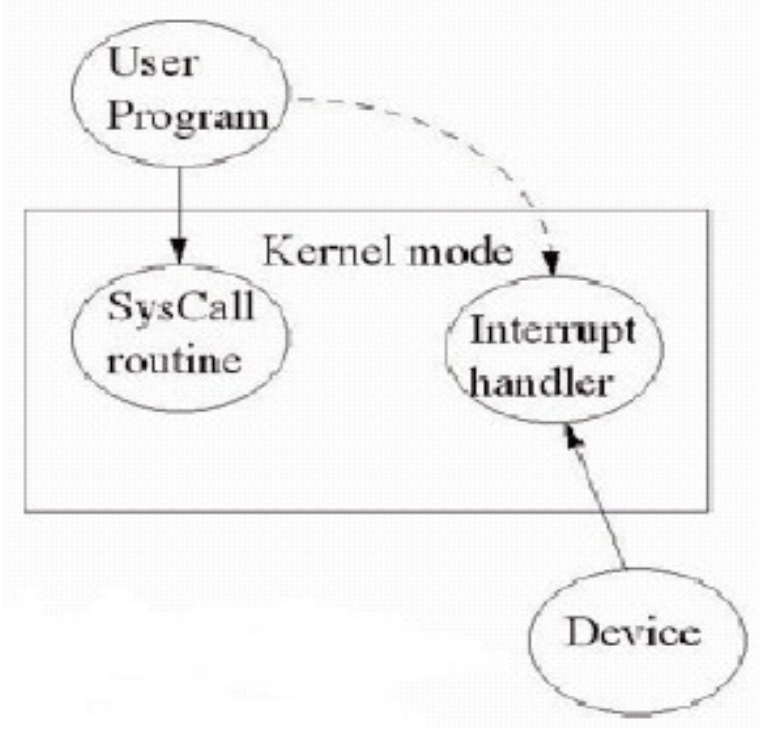
Organizarea sistemului /2

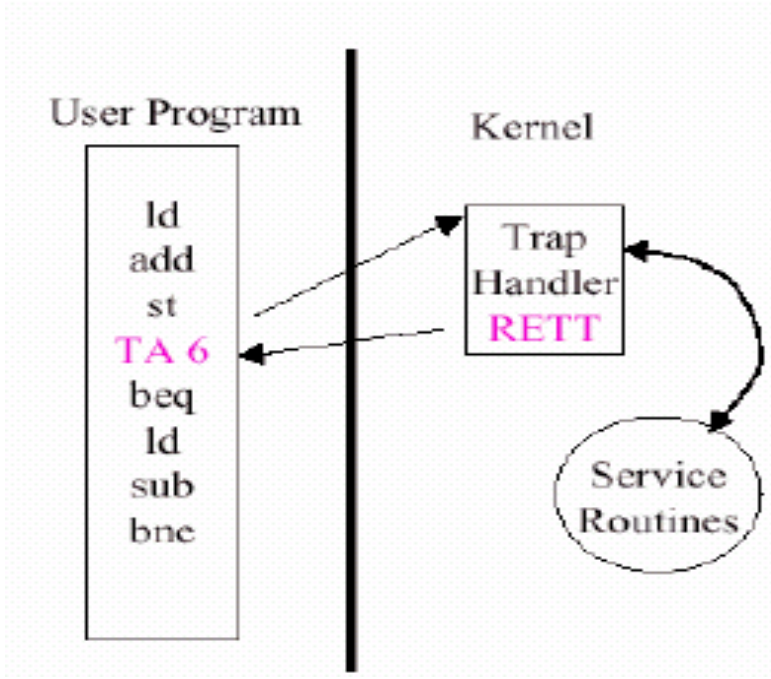
- Termeni I/E
  - **Trap** (excepție) – o întrerupere generată software, cauzată de o eroare sau de o cerere explicită dintr-un program utilizator
  - **Polling** – activitatea de determinare a tipului de întrerupere apărută

- Operarea în mod dual a CPU-ului
  - **User-mode** (modul utilizator): modul de lucru neprivilegiat
  - **Kernel-mode** (modul sistem, sau supervizor): modul de lucru privilegiat

Organizarea sistemului /3

Pentru ca un utilizator să poată face ceva "privilegiat", el trebuie să apeleze acea procedură a S.O. ce furnizează acel serviciu.
Un ***apel de sistem*** **(system call)**.





- O instrucțiune trap specială, care salvează contextul, schimbă modul de lucru (neprivilegiat – privilegiat) și transferă controlul la o rutină handler

Structura unui S.O.

Componentele sistemului de operare:

- Gestiunea proceselor
- Administrarea memoriei principale
- Administrarea memoriei secundare
- Gestiunea sistemului I/E
- Gestiunea fișierelor
- Sistemul de protecție a accesului la resurse
- Networking (gestiunea legăturii la rețea)
- Sistemul de interpretare a comenzilor

Gestiunea proceselor

- Proces
  - Un program în curs de execuție
  - O entitate activă (dinamică) a S.O.-ului
  - Unitatea de lucru într-un sistem de calcul

- Activități S.O.
  - Crearea și distrugerea proceselor user/sistem
  - Suspendarea și reluarea execuției proceselor
  - Mecanisme pentru sincronizarea proceselor
  - Mecanisme pentru comunicații între procese
  - Mecanisme pentru tratarea inter-blocajului

Gestiunea memoriei principale

- Memorie
  - Un vector f. mare de octeți (cuvinte de memorie), având fiecare propria sa adresă
  - Pentru a putea fi executat, un program trebuie să se afle în memoria principală
- Activități S.O.
  - Alocarea și dealocarea spațiilor de memorie la cerere
  - Gestiunea spațiilor de memorie libere și a celor ocupate, precum și a proceselor ce le ocupă
  - Decide ce procese vor fi încărcate în memorie atunci când un spațiu de memorie devine disponibil

Gestiunea memoriei secundare

- Memoria secundară (dispozitive de stocare)
  - O extensie a memoriei principale
  - Pentru stocarea programelor și datelor
  - Dispozitive de stocare: hard-discuri, benzi magnetice, CD-ROM-uri, memorii flash, ...

- Activități S.O.
  - Gestiunea spațiului liber
  - Alocarea spațiului pentru stocare
  - Planificarea acceselor la discuri

Gestiunea sistemului I/E

- Sistemul I/E
  - Lucrează pe bază de zone tampon (*caching*)
  - Sistem de drivere pentru periferice generale
  - Drivere pentru diverse periferice hardware specifice

- Activități S.O.
  - Ascunderea detaliilor specifice diverselor periferice hardware, față de utilizatori/programe de aplicații
  - Rutine de tratare (*handlers*) a întreruperilor

Gestiunea fișierelor

- Fișier
  - O colecție de informații definită de creatorul ei
  - Unitatea logică de stocare a informațiilor
  - Fișierele sunt mapate pe perifericele fizice

- Activități S.O.
  - Crearea și ștergerea fișierelor/directoarelor
  - Suport pentru manipularea fișierelor/directoarelor
  - Maparea fișierelor pe dispozitivele de stocare
  - Backup-ul fișierelor pe medii de stocare nevolatile

Sistemul de protecție a accesului

- Protecție
  - Un mecanism pentru a controla accesul programelor / utilizatorilor la resursele sistemului de calcul

- Activități S.O.
  - Controlează utilizarea (autorizată vs. neautorizată) a resurselor sistemului de calcul

Networking

- Sistem de calcul distribuit
  - O colecție de procesoare care nu partajează o memorie comună sau un ceas intern comun
  - Fiecare procesor are propria sa memorie locală
  - Procesoarele (PCs, workstations, minicomputers, ...) sunt conectate printr-o **rețea de comunicație**
- Activități S.O.
  - Oferă acces la resursele partajate
  - "Ascunde" detaliile legate de topologia fizică a rețelei de comunicație

Sistemul de interpretare a comenzilor

- Interpretorul de comenzi
  - Un program folosit pe post de interfață cu utilizatorul
  - Inclus în nucleu sau privit ca un program special
  - Numit interpretor linie de comandă sau shell

- Instrucțiuni de comandă
  - Interfața utilizator text (e.g. DOS, UNIX)
  - Interfața utilizator grafică (e.g. Macintosh, Windows)

Servicii oferite de S.O.

- Execuția programelor
- Operații I/E
- Manipularea sistemului de fișiere
- Comunicații
- Detecția erorilor
- Alocarea resurselor
- *Accounting* (raportări)
- Protecția accesului

Abstractizări și API-uri S.O.

- Mediu de tip **Mașină Abstractă**
  - S.O.-ul definește un set de resurse logice (abstracte) și de operații asupra acelor resurse (i.e., o interfață de utilizare a acelor resurse)
- "Ascunde" partea de hardware
- Apel sistem (= o primitivă a S.O.-ului)
  - Interfața între un program în curs de execuție și S.O.
  - Apelează o rutină a nucleului (≠ funcție de bibliotecă)

Abstractizări și API-uri S.O./1

Categorii de apeluri sistem

- Controlul proceselor

  end, abort, load, execute, wait, wait event, signal event, allocate, free, get/set attributes
- Manipularea fișierelor

  create, delete, open, close, read, write, reposition, get/set attributes
- Manipularea dispozitivelor periferice

  request, release, read, write, reposition, attach, detach, get/set attributes
- Administrarea informațiilor

  get/set date, get/set system data, ...
- Comunicații

  create, delete, send, receive, write, reposition, get/set attributes

Abstractizări și API-uri S.O./2

Abstractizări UNIX (tradiționale)

- Procese
  fir de execuție (thread) cu context
- Fișiere
  un flux liniar, cu nume, de octeți de date
- Socket-uri
  capete ale canalului de comunicație dintre două procese neînrudite

Abstractizări și API-uri S.O./3

Modelul de proces în UNIX și API-ul POSIX

- Primitive simple, dar puternice, pentru crearea și inițializarea procesului
  - fork() creează un proces fiu (inițial) ca o clonă a procesului părinte
  - programul părintelui se execută în procesul fiu pentru a-l pregăti pentru exec()
  - fiul poate exit(), părintele poate wait() fiul înainte de a face ceva

Exemple din Windows API: CreateProcess(), WaitForSingleObject(), ExitProcess()

Abstractizări și API-uri S.O./4

Modelul de proces în UNIX (cont.)

- Facilități puternice pentru controlul proceselor prin **semnale** asincrone
  - notificări ale apariției unor evenimente interne și/sau externe pentru procese sau grupuri de procese
  - sunt asemenea și au puterea întreruperilor și excepțiilor
  - acțiuni de tratare implicite: stoparea procesului, omorârea procesului, *dump core*, nici un efect (ignorare)
  - rutine de tratare (*handlers*) definite de utilizator

Abstractizări și API-uri S.O./5

Fișiere UNIX și API-ul POSIX

- **Sesiune de lucru** - *descriptorii de fișiere* sunt numere întregi pozitive folosite ca “handles” (referințe) pentru a manipula toate obiectele din sistem care se aseamănă cu fișierele
- open() cu un nume de fișier returnează un descriptor
- read() si write() operează la poziția (*offset*) curentă din fișier
- lseek() repoziționează fișierul, close() termină sesiunea de lucru
- Pipe-urile sunt fluxuri I/E unidirecționale fără nume create de pipe(); pentru pipe-uri cu nume se folosește mkfifo()
- Dispozitivele periferice sunt fișiere speciale create de mknod(), cu parametrii specifici dispozitivului setați cu ioctl()
- Socket-urile oferă câte 3 forme de apel sendmsg() și recvmsg()

Exemple din Windows API: CreateFile(), ReadFile(), WriteFile(), CloseHandle()

Abstractizari si API-uri S.O./6

Abstractizări PalmOS

- Aplicații : programe single threaded, event-driven
- Elemente administrate de S.O.:
  - Obiecte din interfața utilizator: Windows, Forms, Menus
  - Obiecte din memorie: resurse, zone de memorie, databases
  - Facilități de comunicație: linie serială, IR (infra-roșu)
  - Facilități de sistem: alarme (*timers*), generator de sunete, funcții de administrare a puterii electrice consumate, acces la evenimentele de nivel scăzut generate de tastatură și *pen*

Programe de sistem

- Interpretoare de comenzi (shell-uri)
- Manipularea fișierelor
- Modificarea fișierelor
- Informații de stare
- Suport pentru limbajele de programare
- Incărcarea și execuția programelor
- Comunicații
- Programe de aplicații

Nucleul /1

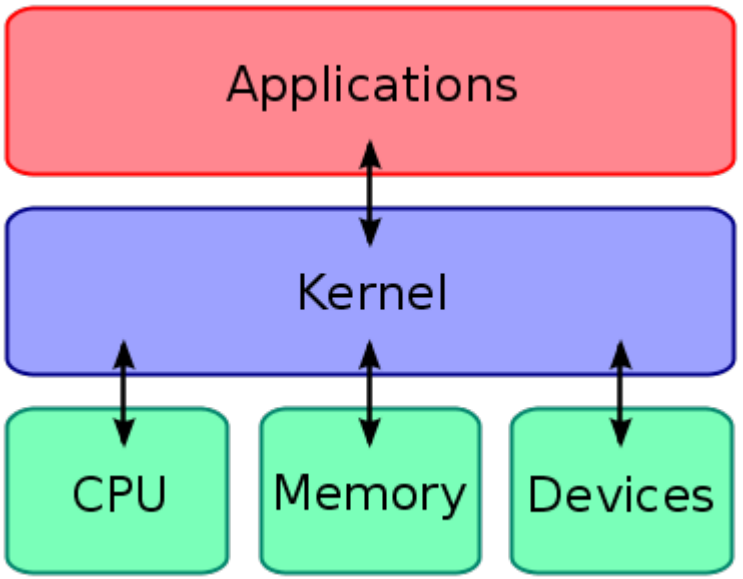
- Este cea mai importantă componentă a S.O.-ului
- Oferă servicii pentru procesele utilizator prin intermediul **primitivelor** (**apelurilor de sistem**)
- Activități importante desfășurate de către nucleu:
  - Gestiunea proceselor
  - Gestiunea memoriei
  - Gestiunea sistemelor de fișiere
  - Gestiunea perifericelor I/E

Nucleul /2

Plasarea serviciilor oferite de S.O.:

– Incluse în spațiul proceselor utilizator:
  funcțiile de bibliotecă
  (cu legare statică vs. legare dinamică)
– În nucleu: apelurile de sistem
– În *userspace* ca procese separate: procese server

Nucleul /3



- Tipuri de nucleu:
  - Nucleu monolitic
  - Micro-nucleu
  - Nucleu hibrid
  - Exo-nucleu

Nucleul /4

Nucleu monolitic:

– toate componentele S.O.-ului rulează în *kernelspace* (i.e. în mod privilegiat) și oferă servicii aplicațiilor prin intermediul apelurilor de sistem

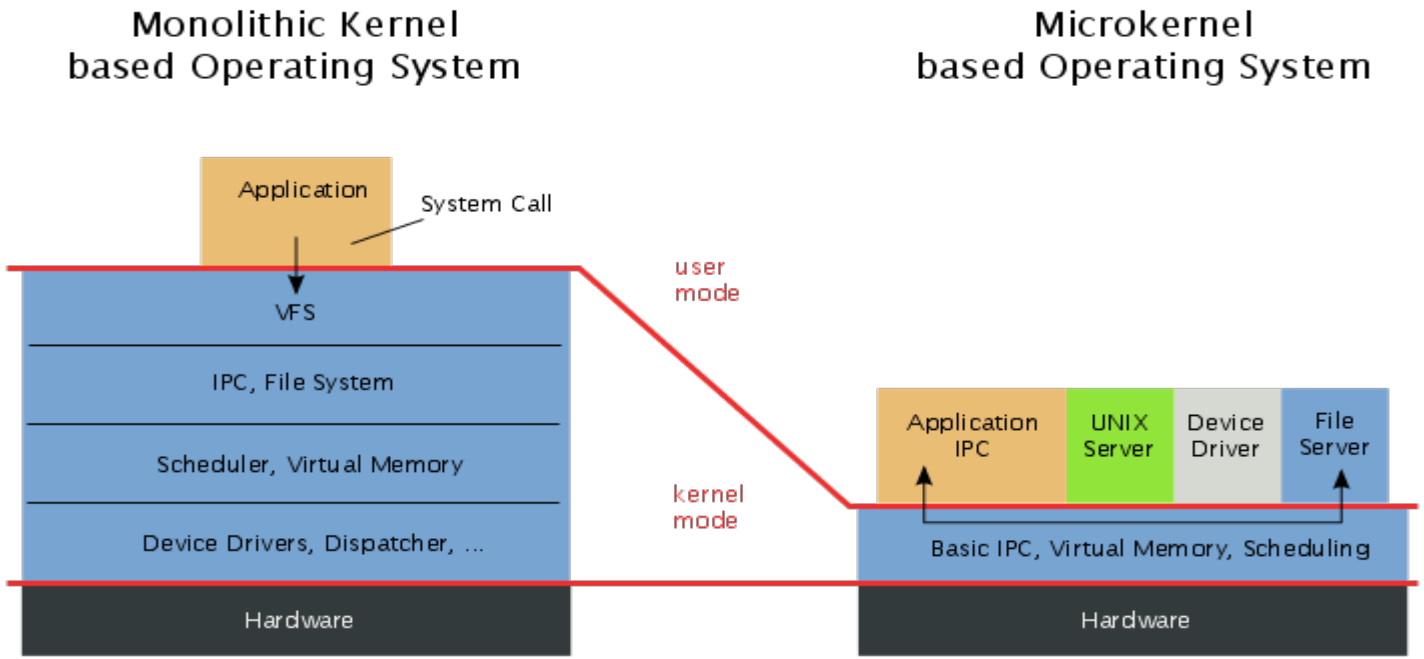
– exemple: UNIX, VMS, DOS, Windows 3.x/9x

– dezavantaje: greu de scris, inflexibilitate

– Nuclee monolitice modulare: au module executabile ce se pot încărca/descărca la cerere în/din memorie la *runtime*
Exemple: Linux, Solaris, FreeBSD și alte variante

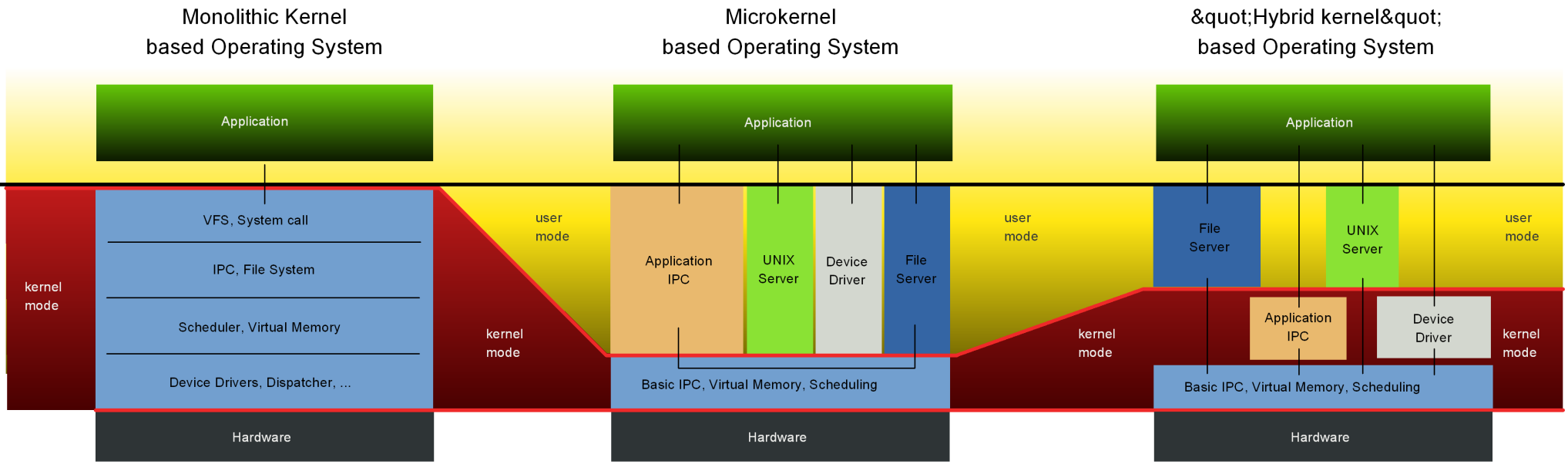
Micro-nucleu:

- principiul minimalității: doar componentele esențiale rulează în *kernelspace,* majoritatea serviciilor rulează în *userspace* (i.e. în mod neprivilegiat) sub formă de procese server
- modularitate, scalabilitate, adaptabilitate, dimensiune mică
- exemple: Mach (@CMU), L4 microkernel, MINIX, QNX

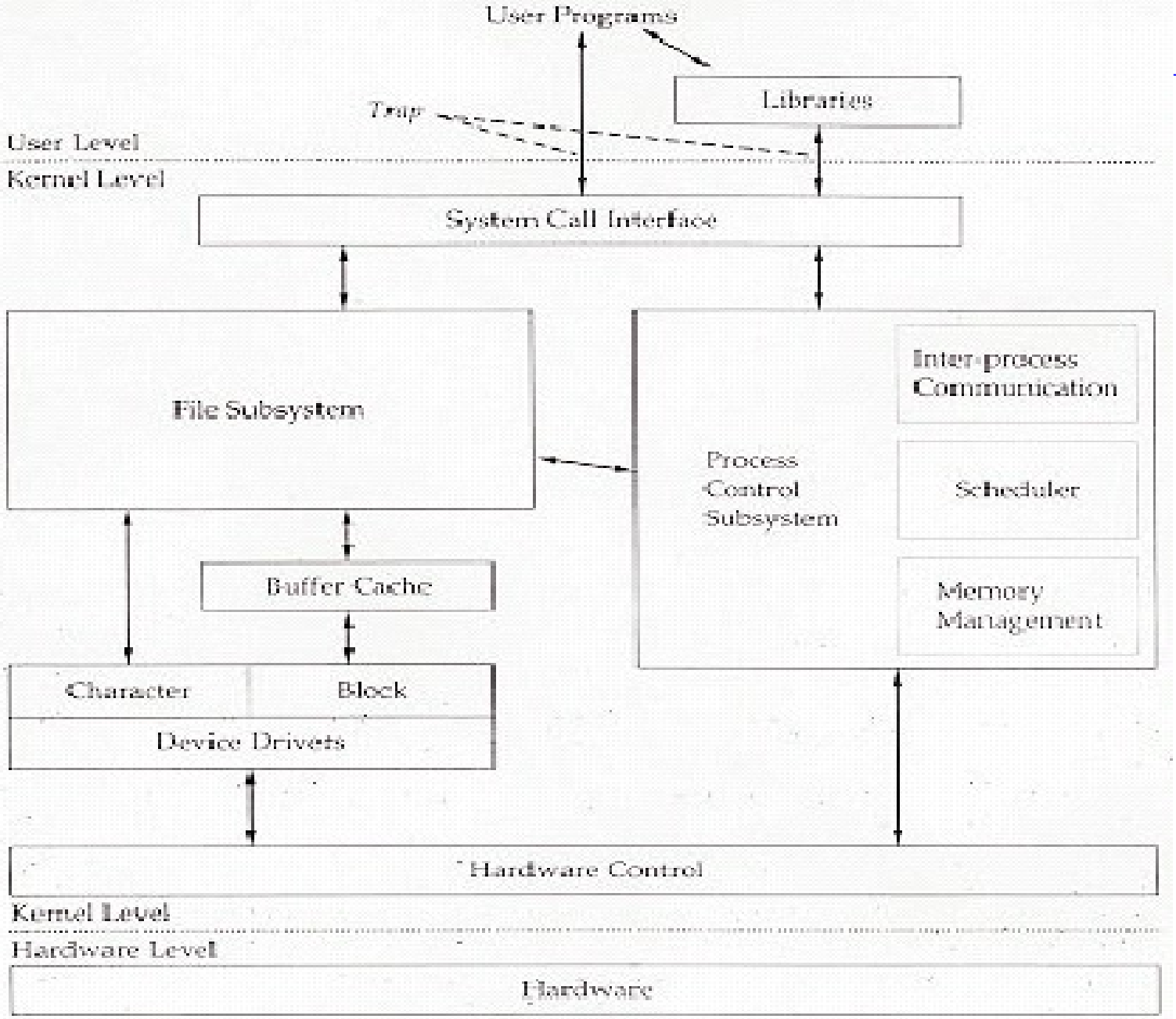


Nucleu hibrid:

- o combinație a arhitecturilor monolitică și micro
- exemple: NT kernel (familia Windows NT4/2000/XP/2003/Vista/2008/7), Plan 9 (@Bell Labs), ReactOS

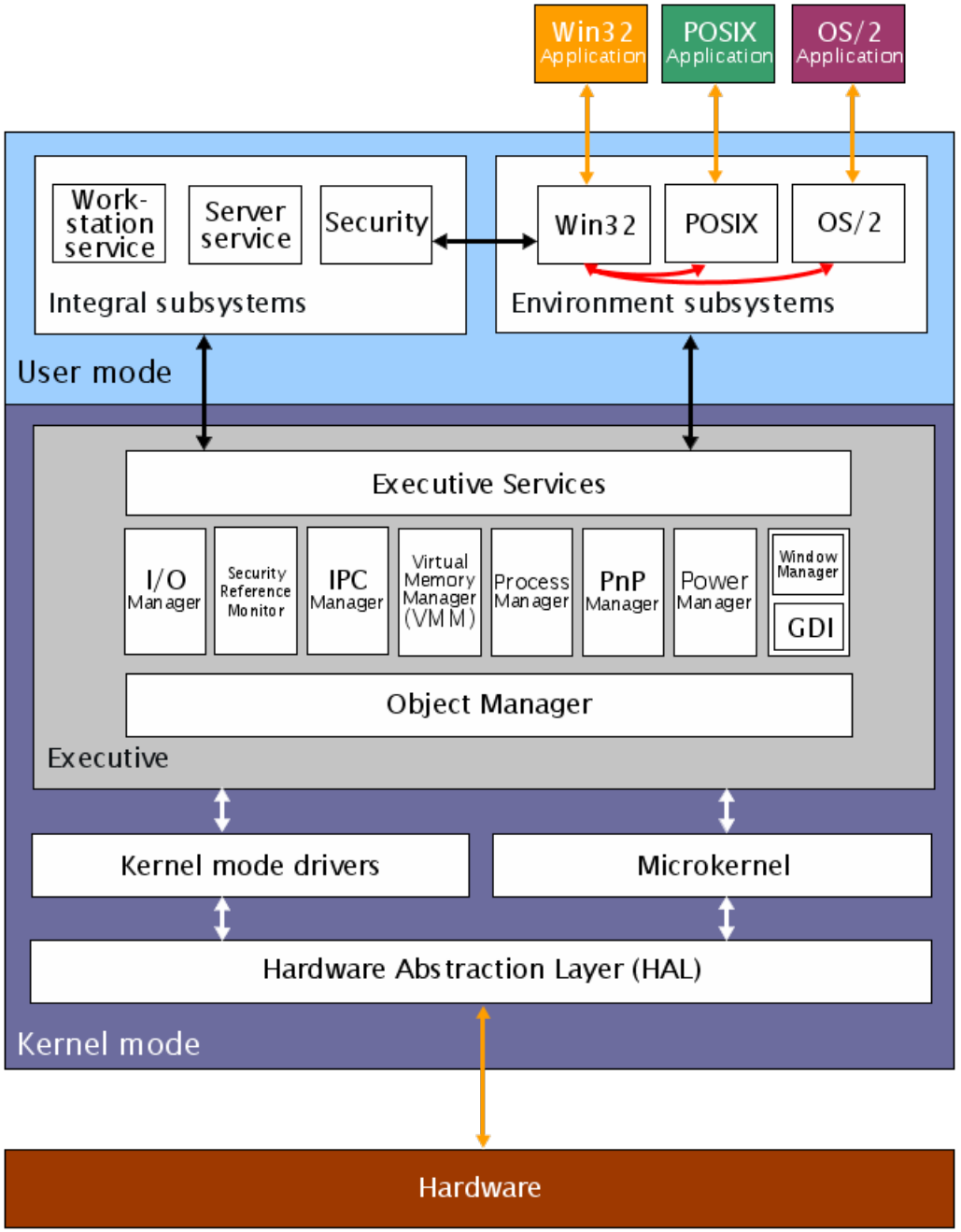


Nucleul /7



Nucleul
UNIX

Nucleul /8



Nucleul
WindowsNT

Bibliografie

- **Bibliografie obligatorie**

  capitolele introductive din

  – Silberschatz : "*Operating System Concepts*"
  (cap.2 din [OSCE8])

  sau

  – Tanenbaum : "*Modern Operating Systems*"
  (a doua parte a cap.1 din [MOS3])

Sumar

- Organizarea unui sistem de calcul
- Structura unui S.O.
- Servicii oferite de S.O.
- Abstractizări și API-uri S.O.
- Programe de sistem
- Nucleul S.O.

Întrebări ?