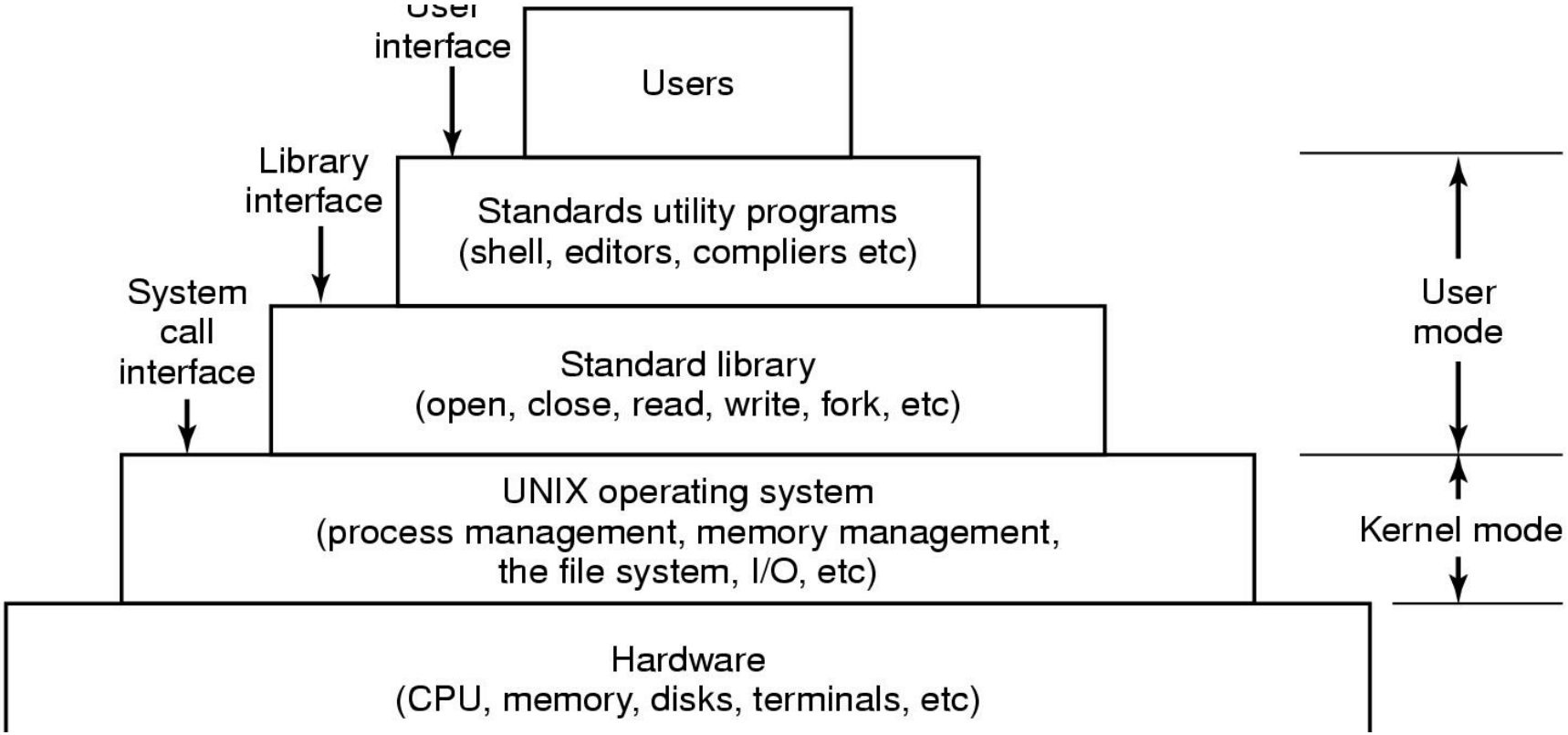
Mode noyau
Appels Systèmes
Interruptions

Grégory Seront
Institut Paul Lambin
E-mail: gregory.seront@ipl.be

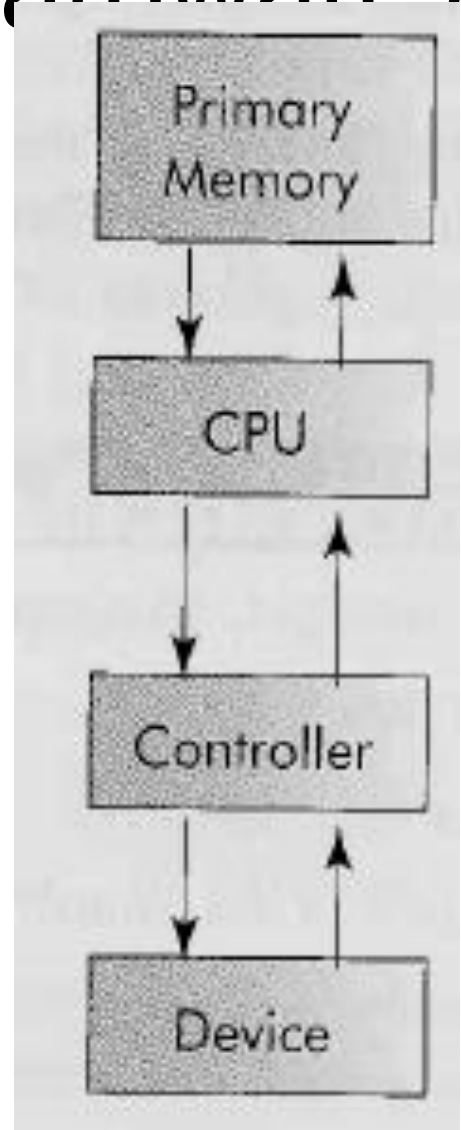
Rappel


interface
Users
Library
interface
Standards utility programs
(shell, editors, compliers etc)
System
call
interface
Standard library
(open, close, read, write, fork, etc)
UNIX operating system
(process management, memory management,
the file system, I/O, etc)
Hardware
(CPU, memory, disks, terminals, etc)
User
mode
Kernel mode

Comment accède-t-on au hardware?

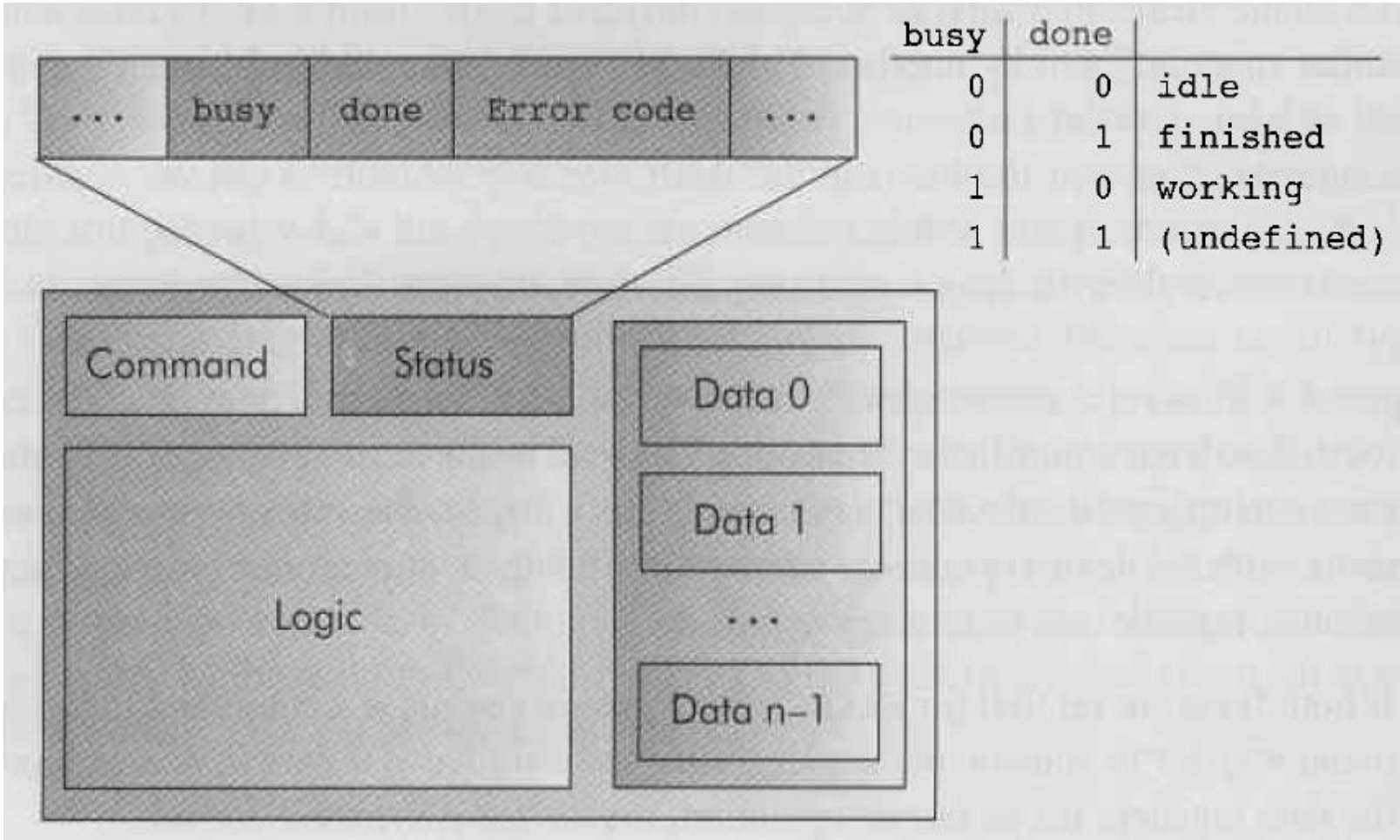
- Le contrôleur offre la possibilité de piloter un périphérique physique au moyen d‘instructions exécutées par le CPU.
  - Assure la transformation des signaux digitaux en signaux analogiques.
  - Ex: Interprète les instructions en ordre de positionnement pour une tête de lecture.

Comment accède-t-on au hardware?



Registres de Contrôle

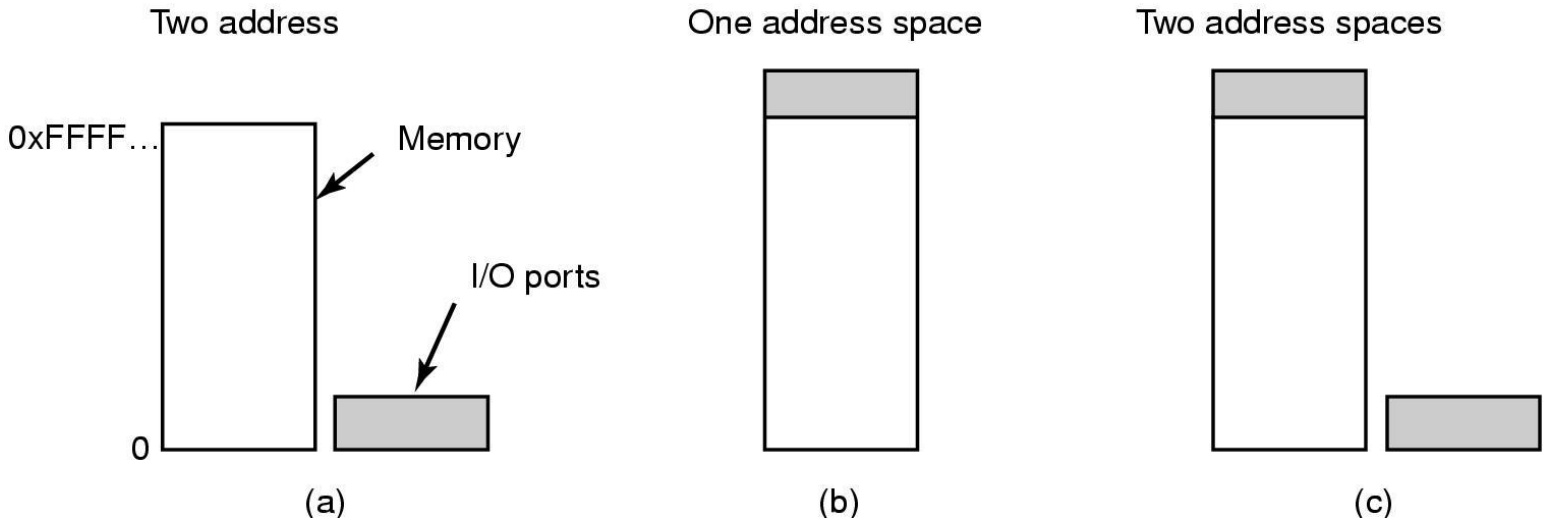
- **Le contrôleur est accédé via des registres de contrôle**



busy	done	
0	0	idle
0	1	finished
1	0	working
1	1	(undefined)

Accès registres contrôle

- Accès via des ports
  - In Reg, Port
  - Out Port, Reg
- Mappés en mémoire
  - Mov Reg, adresse



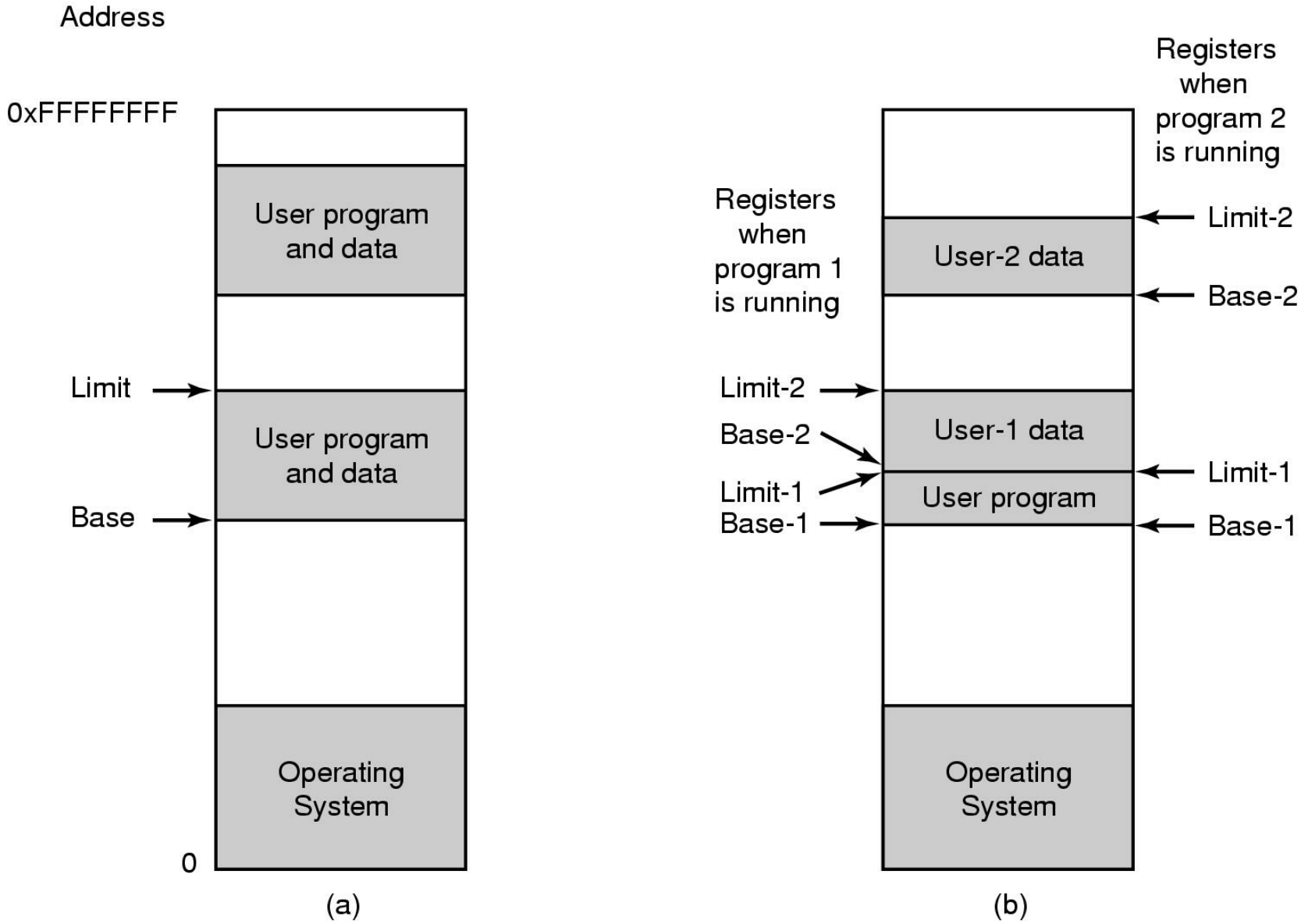
Comment empêcher d’accéder au hardware?

- Certaines instructions privilégiées (permettant l’accès au HW) ne sont accessibles qu’en mode OS.
- On distingue l'OS par un flag de mode superviseur (ou noyau) situé dans le **registre d'état** ou Program Status Word (**PSW**)

Instructions privilégiées

- Quelles sont les instructions privilégiées?
- I/O (accès ports)
  - In, out
- I/O (accès mémoire)
  - Toutes les adresses ne sont pas accessibles
- Protection mémoire
  - Mov non restreint
- Gestion des droits
  - Changement du PSW

Protection mémoire



Mode superviseur/instruction privilégiées

- **C'est le fait d'être en mode superviseur dans le processeur qui distingue l'OS d'un programme utilisateur!**
- **On parle aussi de**
  - **Mode noyau**
  - **Mode OS**
  - **Mode protégé**
- **Concept clef!!**

Appel Système

- Un programme utilisateur peut demander à accéder au hardware via un **Appel Système** à l'OS.

Instructions TRAP

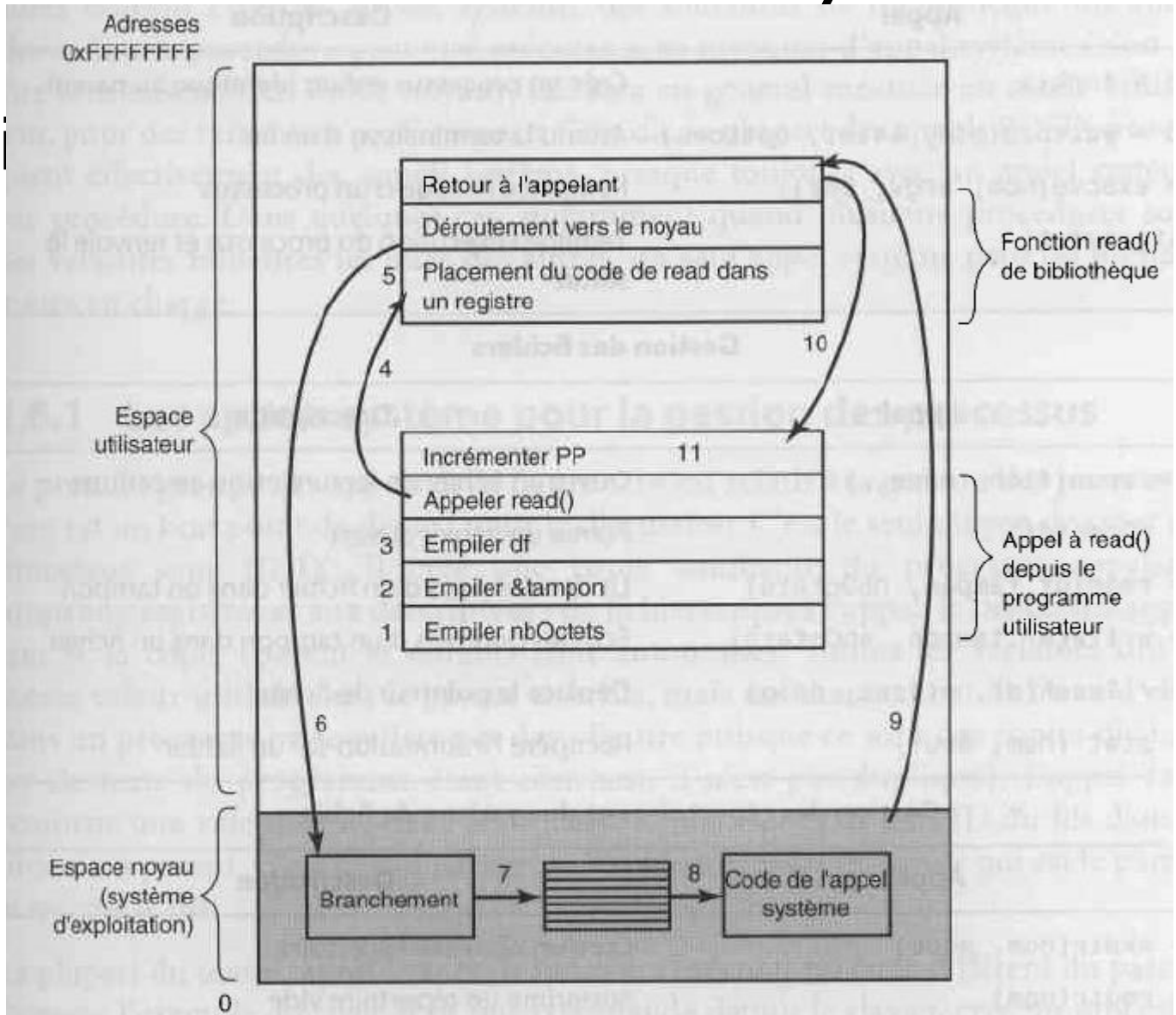
- Comment faire appel à l'OS?
- Instruction spéciale (TRAP) changeant de mode et appelant l'OS.
- Instruction TRAP change juste le mode (les privilèges) ?

Instruction TRAP

- Instruction TRAP change juste le mode?
- Non elle change le contexte(**context switch**):
  - Sauvegarde de **tous** les registres
  - Changement de mode
  - Chargement du contexte de l'OS (avec les droits)
  - Redirection vers une adresse d'une fonction de supervision de l'OS via une **Table de branchement.**

Appel Système read(df, tampon, nboctets)

- Fi



Appel Système

- Changement de contexte (**context switch**):
  - Sauvegarde de **tous** les registres
  - Changement de mode
  - Chargement du contexte de l'OS (avec les droits)
  - Redirection vers une adresse d'une fonction de supervision de l'OS via une Table de branchement.

Table de branchement

- La table de déroutement peut-elle être dans l’espace utilisateur?
- Non! Sinon l’utilisateur pourrait changer les adresses de déroutement et exécuter en mode OS du code à lui!

Appel système Linux en ASM

```
mov     eax, 4
; specify the sys_write function code (from
   OS vector table)
mov     ebx, 1
mov     ecx, str
mov     edx, strLen
int 80h
; tell kernel to perform the system call we just
   set up
```

Quelques appels système UNIX

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

Quelques appels système UNIX

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Quelques appels système UNIX

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Ce qui se passe dans le shell

```
while (TRUE) {                                  /* repeat forever */
    type_prompt( );                             /* display prompt */
    read_command (command, parameters)          /* input from terminal */

if (fork() != 0) {                              /* fork off child process */
    /* Parent code */
    waitpid( -1, &status, 0);                   /* wait for child to exit */
} else {
    /* Child code */
    execve (command, parameters, 0);            /* execute command */
 }
}
}
```

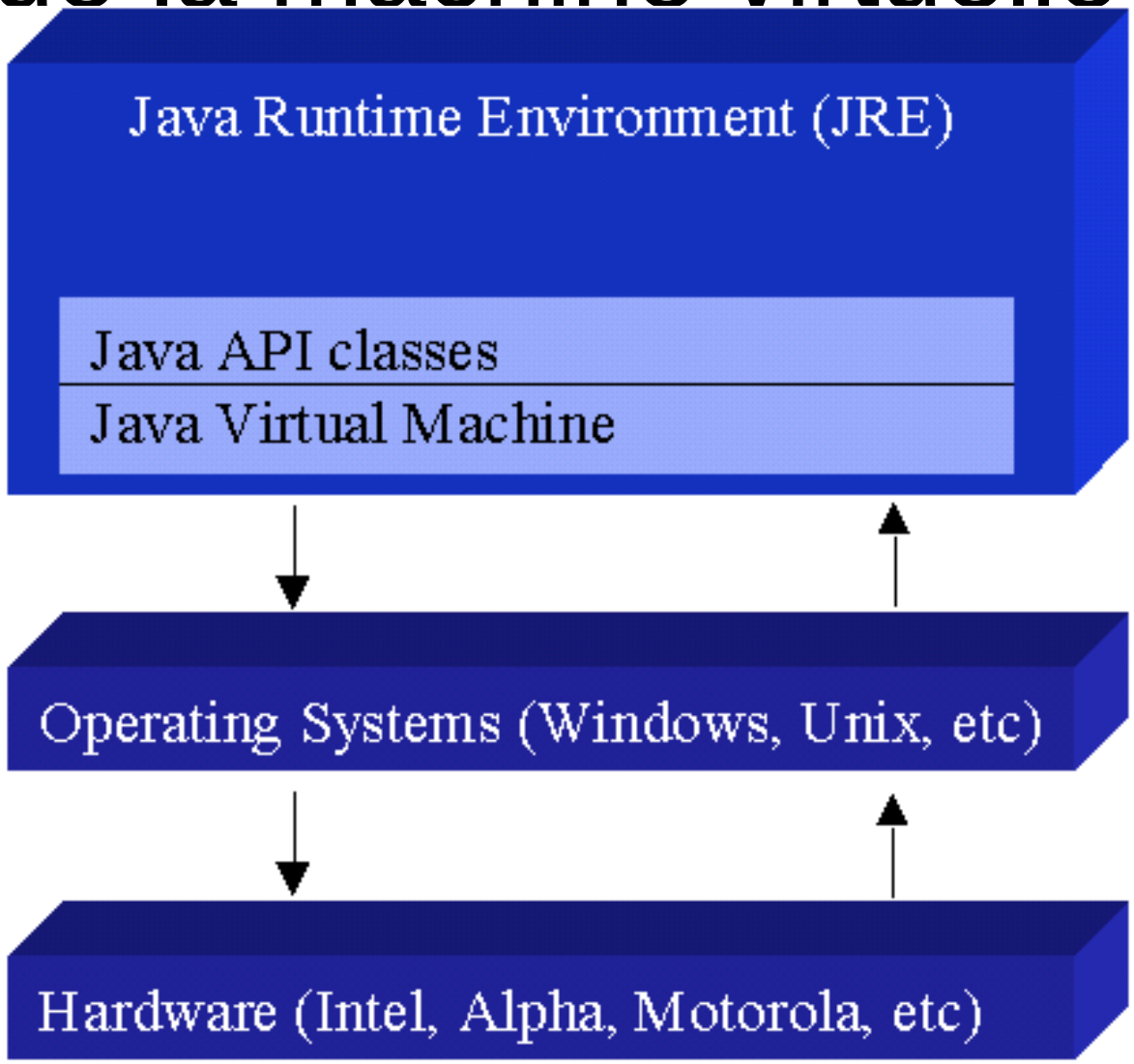
Quelques appels système Win32

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

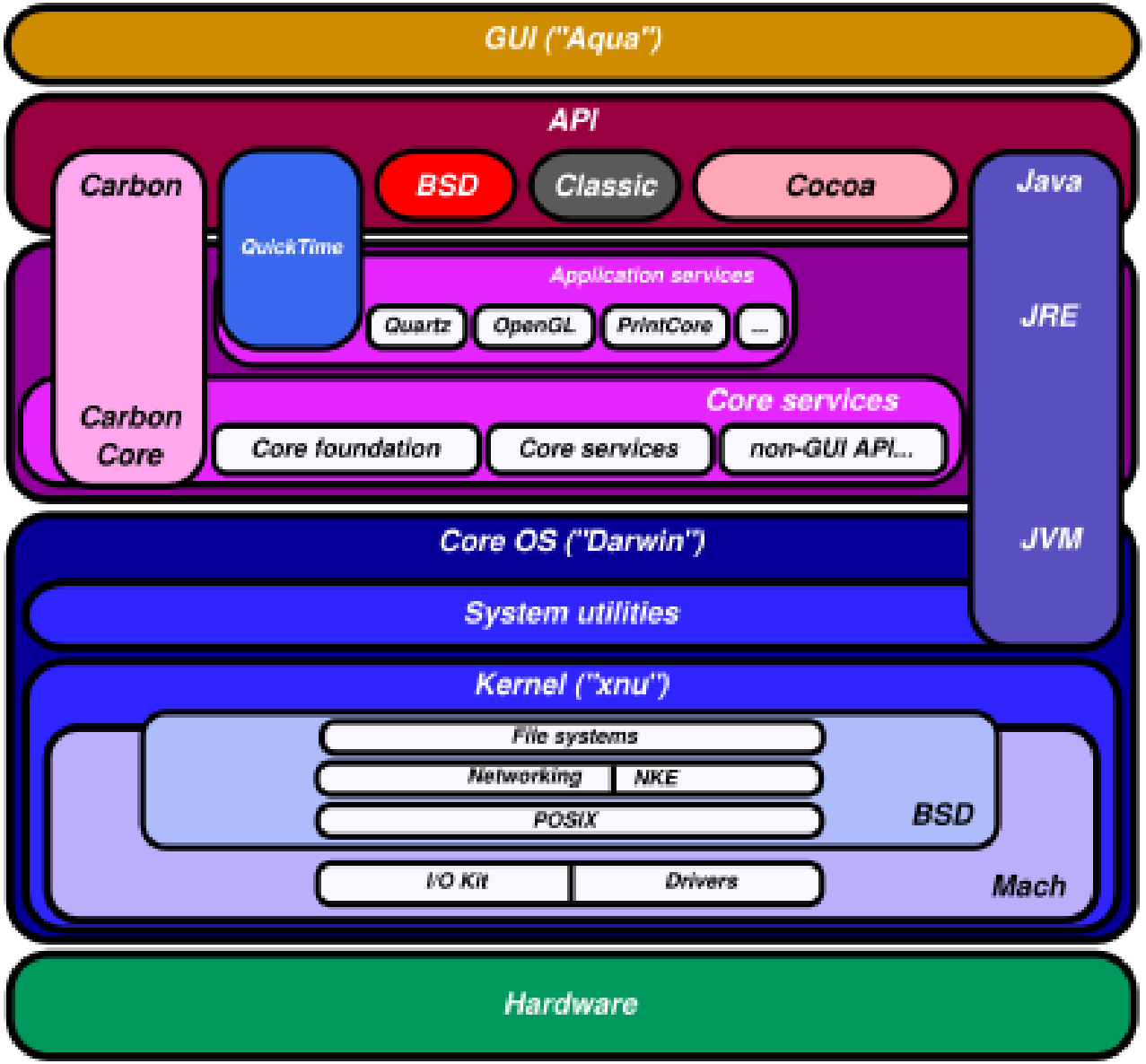
En bref

- Pas d’accès au HW en mode utilisateur
- Distinction dans le PSW
- Accès HW interdit via des instructions privilégiées
- Accès seulement via des appels systèmes
- Appel via empilement sur le stack et une instruction TRAP
- TRAP provoque un changement de contexte (context switching).

Cas de la machine virtuelle Java



Cas de la machine virtuelle Java



Cas de la machine virtuelle Java

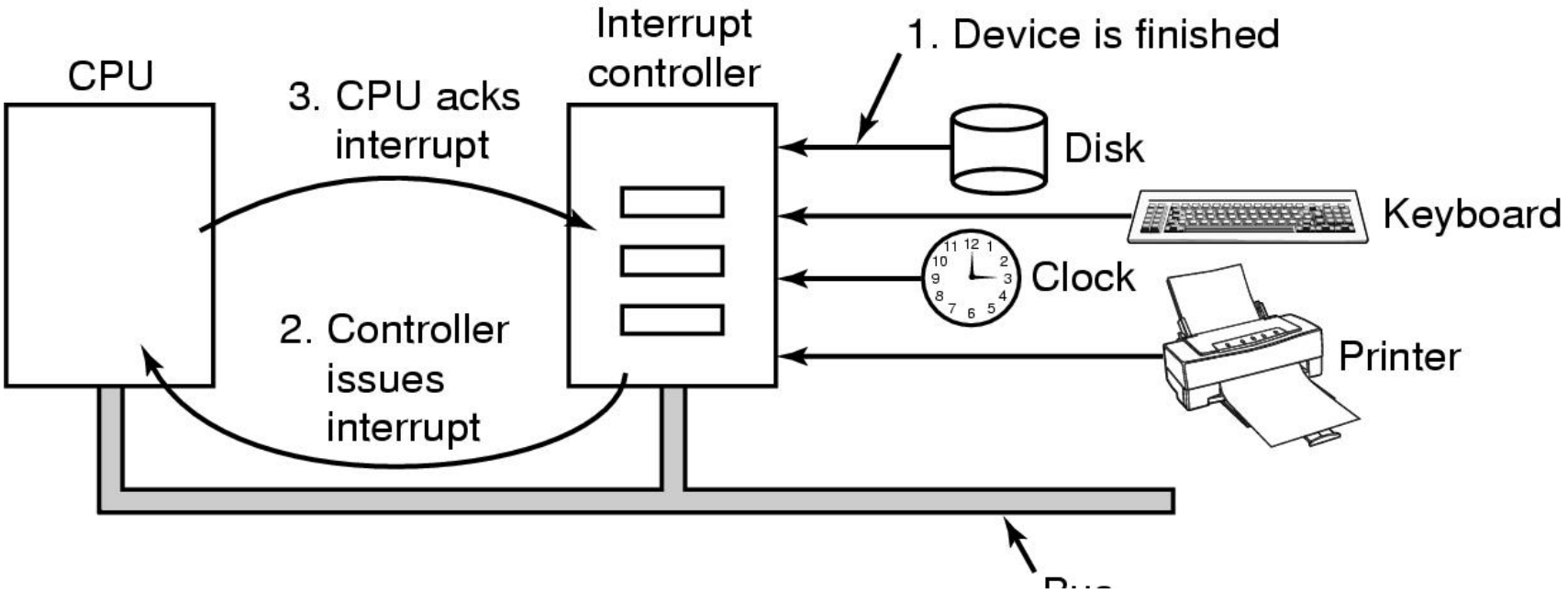
- Vous ne faites pas directement des appels systèmes en Java
- Il faut passer par un langage compilé (C, C++, assembleur) pour directement appeler l’API de l’OS

Les interruptions

Communications HW vers l'OS

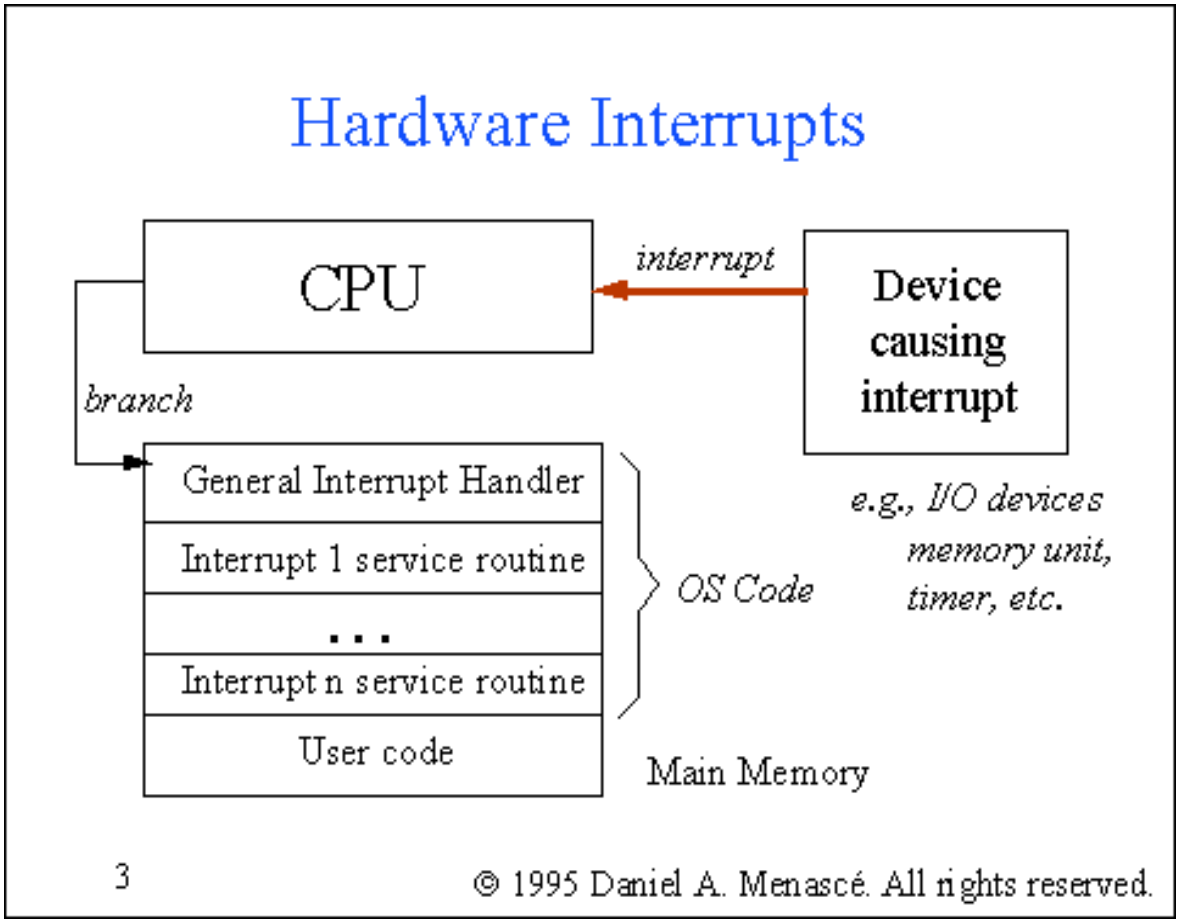
- Qu'est-ce que le HW peut signaler à l'OS?
  - Nouveau caractère au clavier
  - Lecture disque finie
  - Bientôt plus de batterie
  - Timer écoulé
  - …

Communications HW vers l'OS



Interruption

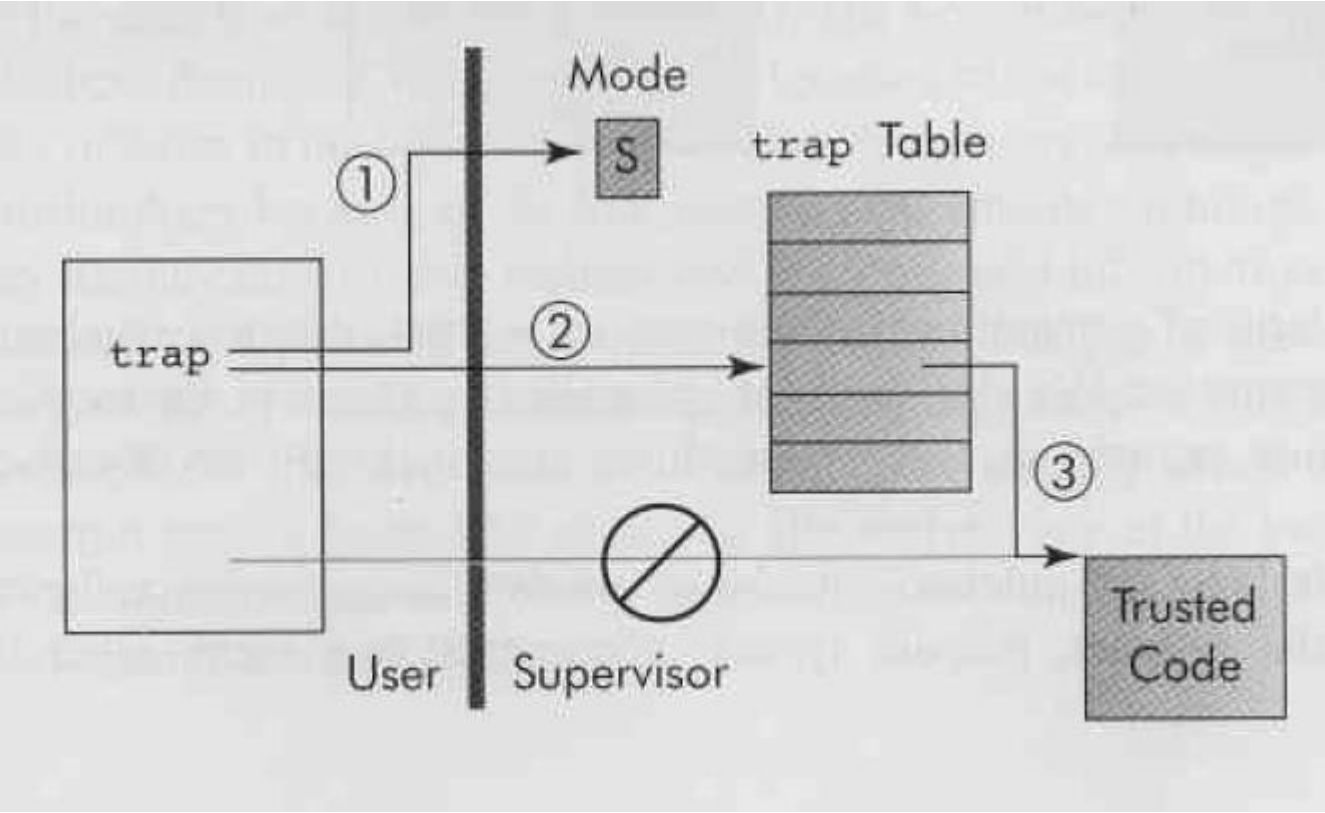
- **Interruption = signal électrique**



Gestion des Interruptions

- Interruption hardware = appel système provoqué par le matériel
- Il y a des entrées « physiques » sur le processeur pour recevoir les demandes d'interruptions (Interrupt ReQuest ou IRQ chez Intel)

Gestionnaire d'interruption



Gestion des Interruptions

- Interrupt = appel système:
  - Sauvegarde de **tous** les registres
  - Changement de mode
  - Chargement du contexte du gestionnaire d'interruptions (avec les droits)
  - Redirection vers une adresse d'une fonction de supervision de l'OS via une Table de redirection.

Interruptions concurrentes

- Plusieurs périphériques: danger?
- Si 2ème interruption pendant le traitement de la 1ère?
- On parle de “**course**”.
- => On va **masquer** les interruptions pendant le traitement
- Certaines plus prioritaires => on a un tableau de masques

Hiérarchie des interruptions

- Interruptions non masquables (reboot, HW failure)
- Interruptions Hardware
- Interruptions Software (Appels Systèmes)

Sur Intel

- On utilise l’instruction ‘int’ pour faire un appel système
- Ne pas confondre
  - Interruption software == appel système par un programme
  - Interruption hardware == appel par le matériel

En bref

- Une interruption est un « appel système » généré par le hardware lorsqu'ils ont quelque chose à dire à l'OS
- Géré comme tel (passage en mode OS)
- C'est un signal électrique
- Attention aux interruptions concurrentes