IFT-2001
SYSTÈMES D'EXPLOITATION

Mondher Bouden

Chargé de cours

Département d'informatique et de génie logiciel
Université Laval

Automne 2020

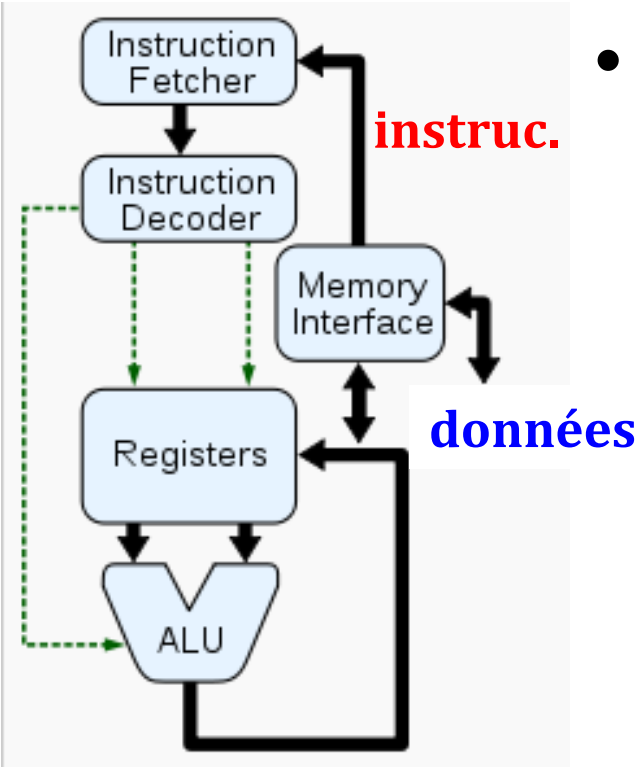
©P. Giguère, Dép. IFT.

Structure matérielle d’un ordinateur

Composantes de base d'un ordinateur

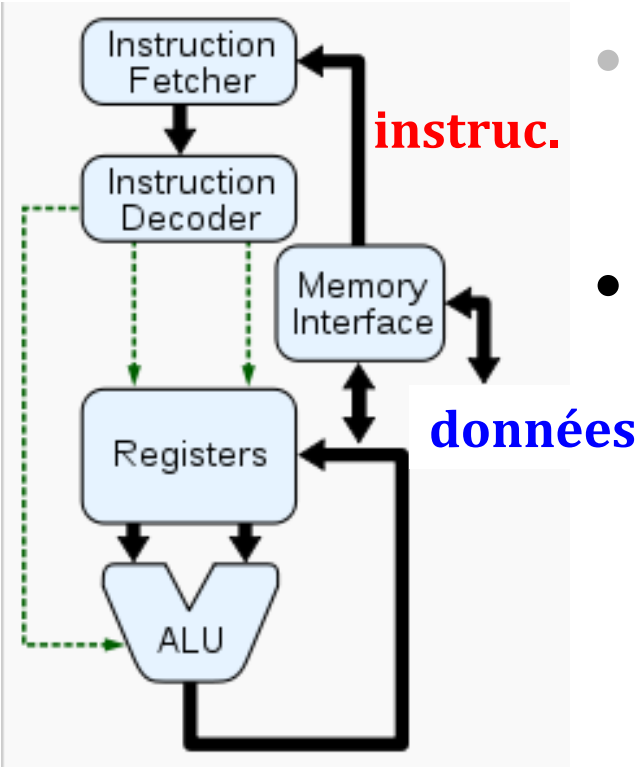
- Processeur(s) ou CPU
- Mémoire
- Disques
- Périphériques d'entrées et sorties (E/S)
- Les bus (ISA, PCI, etc…)

Processeur (CPU)



- Base même de tous les ordinateurs
- Extrait les instructions de la mémoire et les exécute.

Processeur (CPU)



- Base même de tous les ordinateurs
- Extrait les instructions de la mémoire et les exécute.
- Chaque modèle possède
  - jeu d'instructions
  - registres
  - architecture
  - mémoire (cache)
  - autres (gestion mémoire, etc…)

SE s'appuie sur les capacités d'un processeur

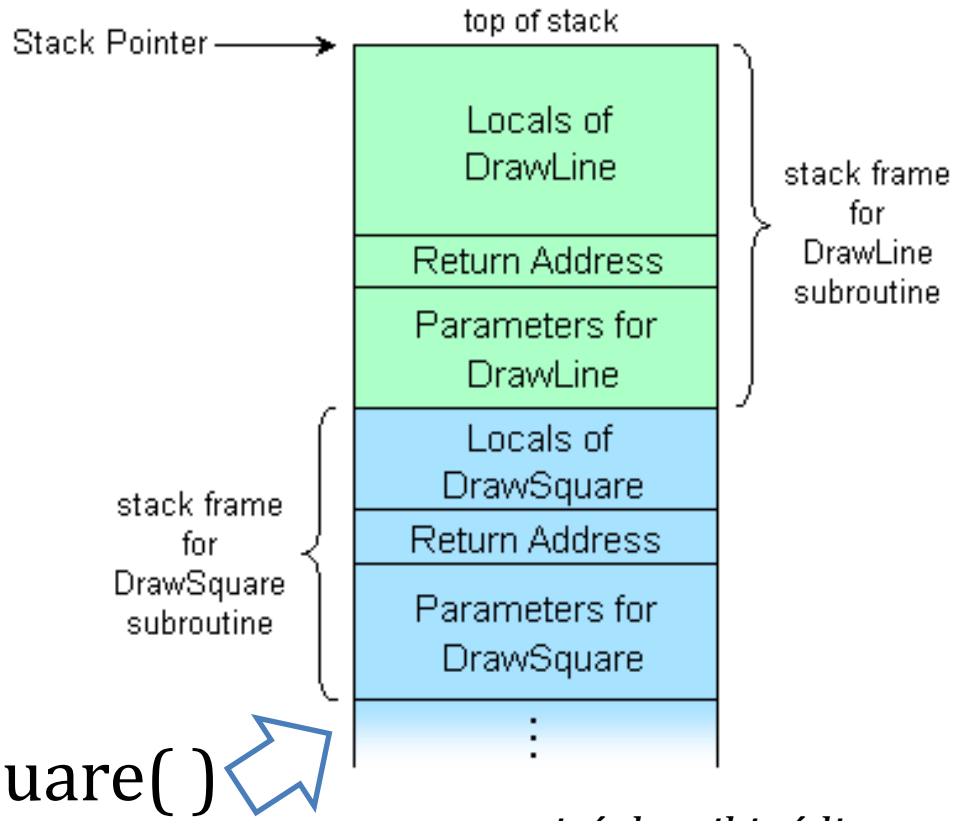
Principaux registres CPU

- Compteur ordinal / *Program Counter* (PC)
  - adresse de l'instruction en cours
- Pointeur de pile / *Stack pointer*
  - adresse courante du sommet de la pile
- Mot d'état / *Program status word*
  - état du processeur +

    bits de comparaison (zéro, overflow, etc)

Révision : pile (*stack*)

- Sert pour :
  - passer les paramètres lors d'un appel de fonction
  - adresse de retour
  - variables locales de cette fonction

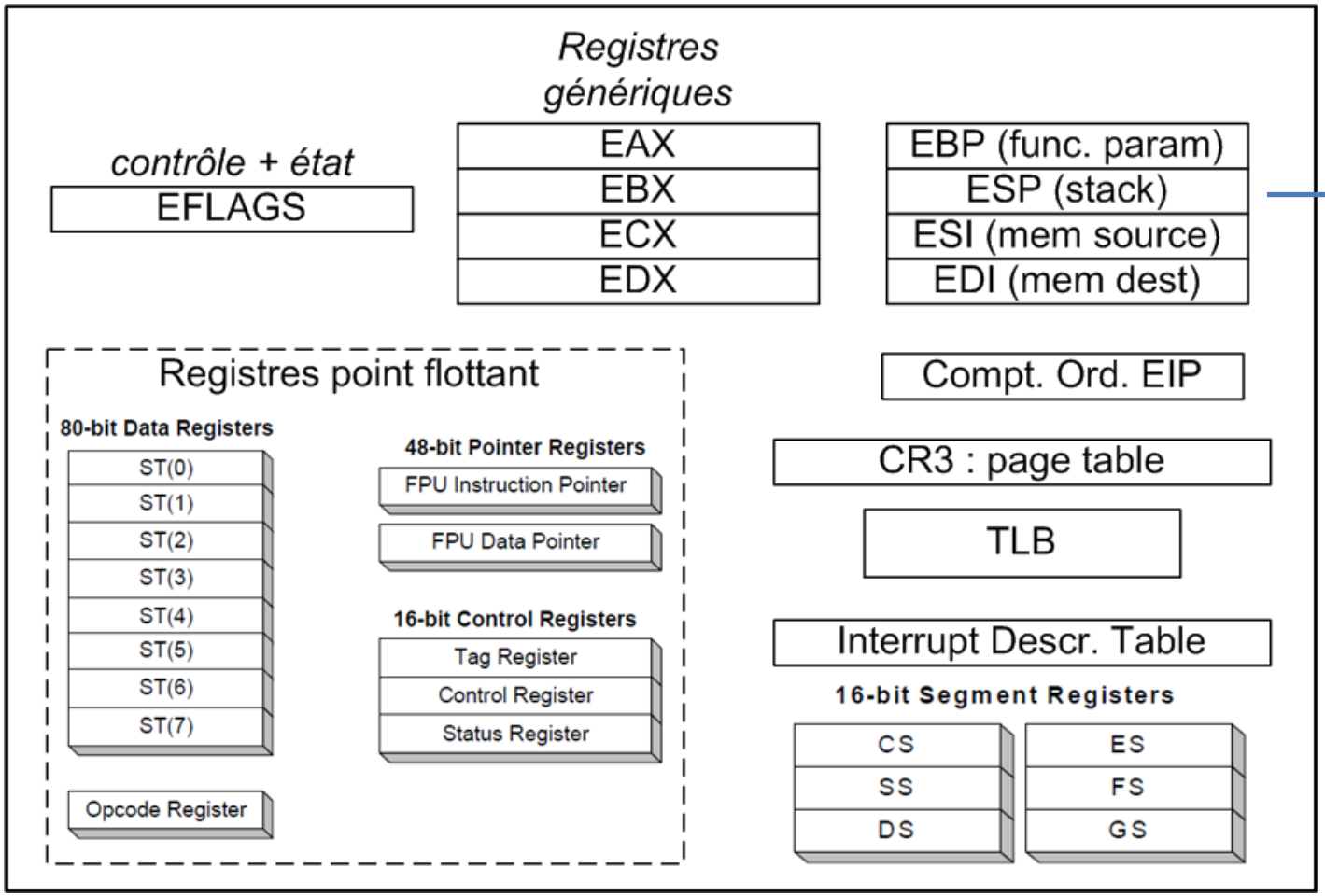


DrawLine( ) est appelé de DrawSquare( )

tiré de wikipédia
auteur : R. S. Shaw

Principaux registres CPU

- Architecture Intel x86



Principaux registres CPU

- Exemple avec addition

mémoire

A=?

Registre **EIP**

Registre **EAX**

```
int A;
A = 3;

A = A + 4;
```

Principaux registres CPU

- Exemple avec addition

mémoire

A=?

Registre **EIP**

Registre **EAX**

```
            int A;
            A = 3;
001F138E    mov   dword ptr [A],3
            A = A + 4;
001F1395    mov   eax,dword ptr [A]
001F1398    add   eax,4
001F139B    mov   dword ptr [A],eax
```

(Sortie du compilateur Visual C++ 2008 Express Edition)

Principaux registres CPU

- Exemple avec addition

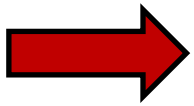
mémoire

A=?

Registre **EIP**

0x001F138E

Registre **EAX**



```
            int A;
            A = 3;
001F138E    mov   dword ptr [A],3
            A = A + 4;
001F1395    mov   eax,dword ptr [A]
001F1398    add   eax,4
001F139B    mov   dword ptr [A],eax
```

Principaux registres CPU

- Exemple avec addition

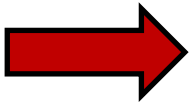
mémoire

A=3

Registre **EIP**

0x001F1395

Registre **EAX**



```
            int A;
            A = 3;
001F138E    mov   dword ptr [A],3
            A = A + 4;
001F1395    mov   eax,dword ptr [A]
001F1398    add   eax,4
001F139B    mov   dword ptr [A],eax
```

Principaux registres CPU

- Exemple avec addition

mémoire

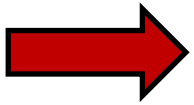
A=3

Registre **EIP**

0x001F1398

Registre **EAX**

3



```
            int A;
            A = 3;
001F138E    mov   dword ptr [A],3
            A = A + 4;
001F1395    mov   eax,dword ptr [A]
001F1398    add   eax,4
001F139B    mov   dword ptr [A],eax
```

Principaux registres CPU

- Exemple avec addition

mémoire

A=3

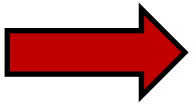
Registre **EIP**

0x001F139B

Registre **EAX**

7

```
            int A;
            A = 3;
001F138E    mov   dword ptr [A],3
            A = A + 4;
001F1395    mov   eax,dword ptr [A]
001F1398    add   eax,4
001F139B    mov   dword ptr [A],eax
```



Principaux registres CPU

- Exemple avec addition

mémoire

A=7

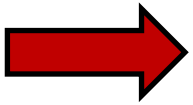
Registre **EIP**

0x001F139E

Registre **EAX**

7

```
            int A;
            A = 3;
001F138E    mov   dword ptr [A],3
            A = A + 4;
001F1395    mov   eax,dword ptr [A]
001F1398    add   eax,4
001F139B    mov   dword ptr [A],eax
001F139E    ...
```



Principaux registres CPU

- Exemple avec addition

mémoire

A=7

Registre **EIP**

0x001F139E

Registre **EAX**

7

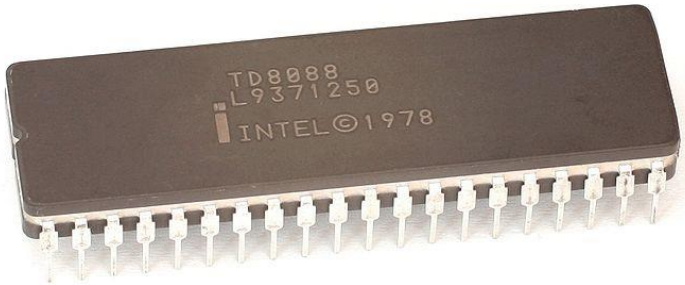
```
            int A;
            A = 3;
001F138E    mov   dword ptr [A],3
            A = A + 4;
001F1395    mov   eax,dword ptr [A]
001F1398    add   eax,4
001F139B    mov   dword ptr [A],eax
001F139E    ...
```

est en fait 3 instructions assembleur

Évolution : monotâche

- 8088 : Accès à tous les registres, adresses mémoires

Intel 8088
16 bits interne, 5-10 MHz



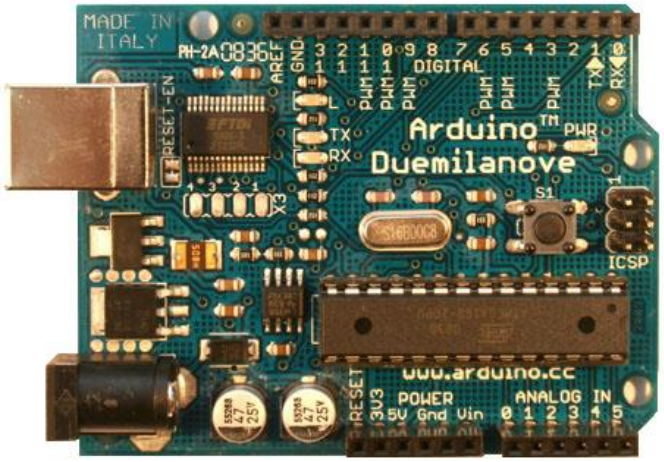
- Si désire faire tourner + d'un processus, il faut :
  - protéger les autres processus des incursions (accidentelles ou volontaires) ?
  - le faire de façon efficace (rapide)
  - comment : **nécessite CPU plus évolué**

(S.E. s'appuient énormément sur le matériel)

Évolution : Multitâches

- Mode protégé (*protected*)
- Chez *Intel* en 1982 : 80286

- Pas sur tous les CPU récents :
  - Microcontrôleur (ATmega168)



Arduino

Intel 80286

An 8MHz Intel 80286 Microprocessor

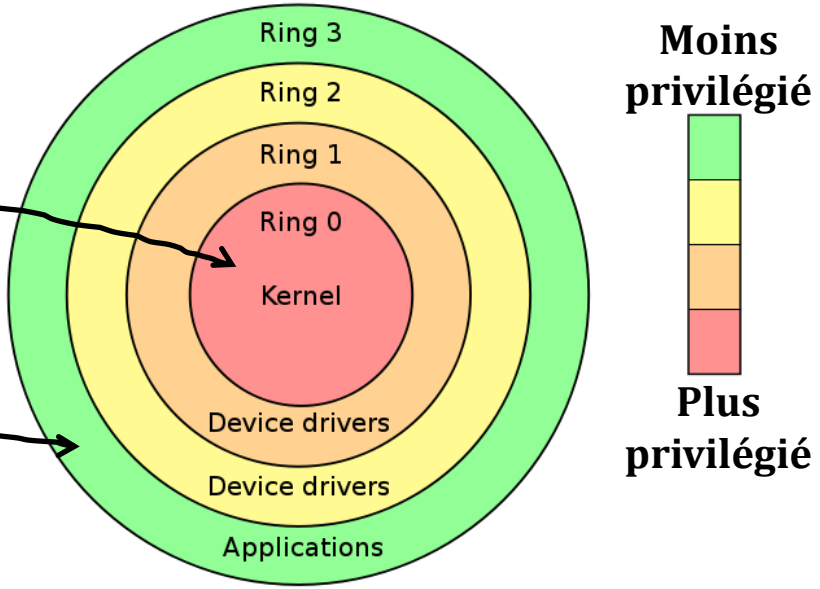
Produced	From 1982 to early 1990s
Common manufacturer(s)	Intel, IBM, AMD, Harris (Intersil), Siemens AG, Fujitsu
Max. CPU clock rate	6 MHz (4 MHz for a short time) to 25 MHz
Min. feature size	1.5µm
Instruction set	x86-16 (with MMU)
Package(s)	PGA, CLCC and PLCC 68 -pin

Mode protégé (*protected*)

- Pour avoir **superviseur (S.E.)**, mécanisme **matériel**
  - avoir des privilèges d'accès *(au matériel)* particuliers à chaque anneau
  - limiter comment on passe d'un anneau à l'autre
  - pouvoir attribuer des niveaux de privilèges au code

• Mode **noyau** (0) peut exécuter code 0,1,2,3

•Mode **utilisateur** (3) peut exécuter code 3 seulement



anneaux de protection (Intel)

Instructions machines privilégiées

- Provoque déroutement si mode **utilisateur (3)**

(interrupt)

```
ARPL - Adjusted Requested Privilege Level of
Selector
```

Arrêter le processeur

```
HLT  - Halt processor until next interrupt
```

segments mémoires
(spécifie niveau privilège du code)

```
LGDT - Load IFTbal Descriptor Table
SGDT - Store IFTbal Descriptor Table
SLDT - Store Local Descriptor Table
LLDT - Load Local Descriptor Table
```

caches internes

```
INVD   - Invalidate internal Caches
WBINVD — Write Back and Invalidate Cache
INVLPG - Invalidate TLB entry
```

entrées/sorties

```
IN   - Read port
OUT  - Write port
```

tables interruptions

```
LIDT - Load Interrupt Descriptor Table
SIDT - Store Interrupt Descriptor Table
```

niveau actuel de privilège

```
SMSW - Store Machine Status Word
LMSW - Load Machine Status Word
```

(donc, uniquement exécutable en mode **noyau (0)**)

etc…

```
LSL  - Load Segment Limit
LTR  - Load Task Register
STR  - Store Task Register
CLTS - Clear Task Switched Flag

SWAPGS—Swap GS Base Register
```

Instructions machines privilégiées

- Provoque déroutement *(interrupt)* si mode **utilisateur (3)**

Arrêter l

segments m
(spécifie niveau privi

caches

entrée

tables inte

niveau actuel de
(donc, unique
exécutable en mode
noyau (0))

ARPL - Adjusted Requested Privilege Level of
ssor until next interrupt
l Descriptor Table
al Descriptor Table
l Descriptor Table
Descriptor Table
ce internal Caches
ck and Invalidate Cache
ce TLB entry
cupt Descriptor Table
crupt Descriptor Table
ine Status Word
ne Status Word
it Limit
Register
Register
CLTS - Clear Task Switched Flag
SWAPGS—Swap GS Base Register

CPU multitâches : 2 modes

Niveau privilège (CPU)	Système d'exploitation	Accès
Ring 0	**noyau** (*kernel*)	accès complet
Ring 3	**utilisateur** (*user*)	accès **limités** pour : instructions registres mémoire

- Avantages
  - contrôle serré des ressources, protège la mémoire
- Défaut : basculer (**`trap`**) d'un à l'autre est cher :
  - approx. 1500 cycles machines → **performance**

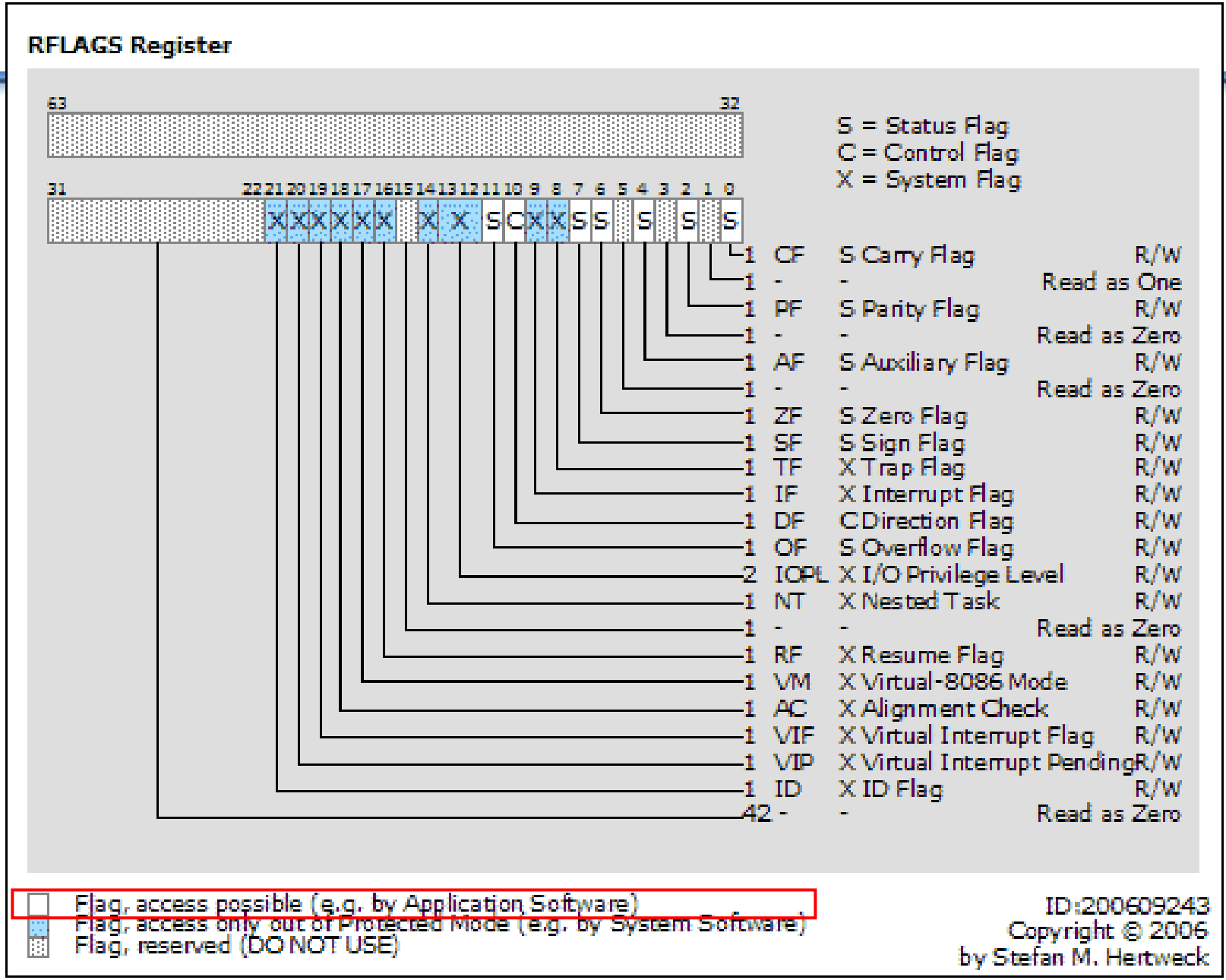
Niveau privilège processeur vs. admin

deux mécanismes différents!!!!

niveau privilège noyau (0)	≠	compte administrateur
matériel		**logiciel**

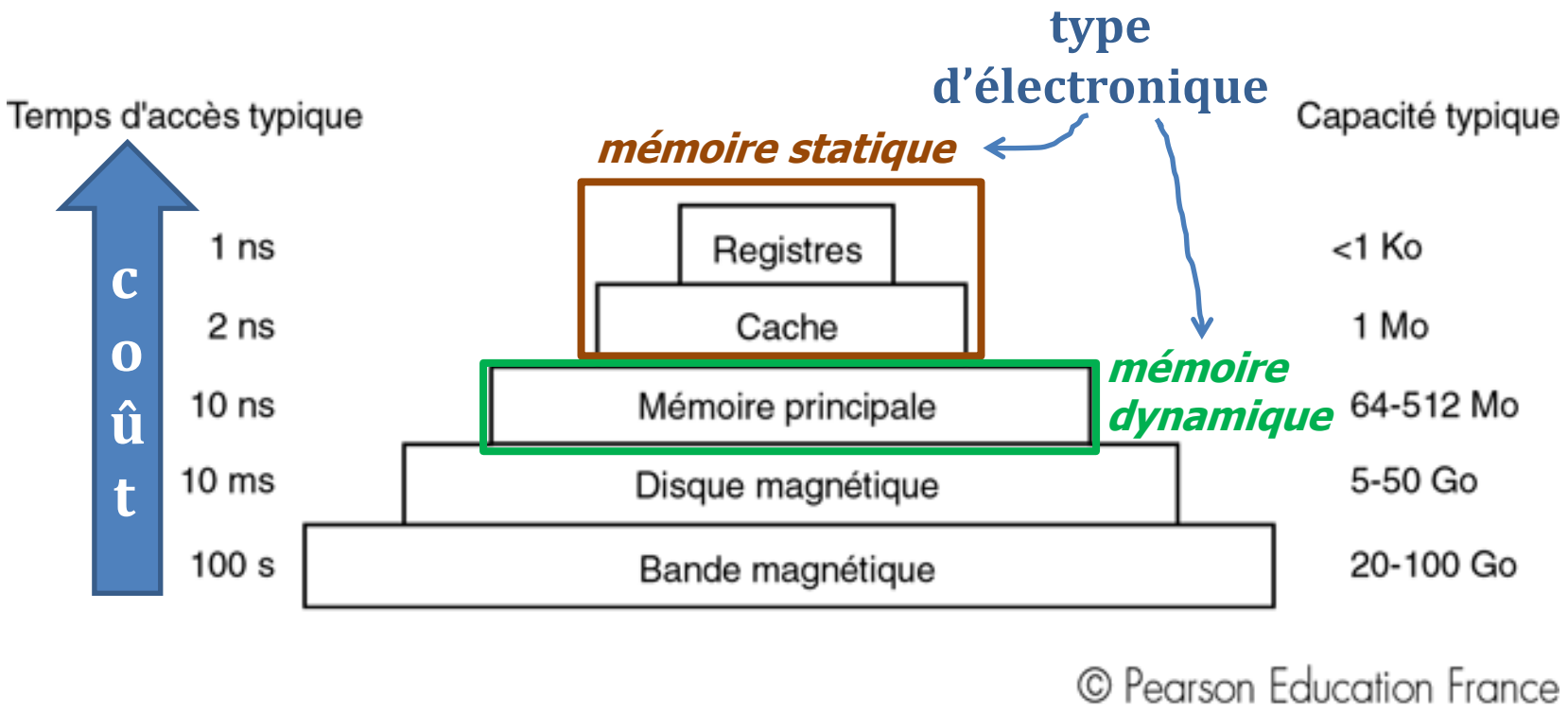
Un compte *admin* est un concept inconnu pour un processeur

Intel EFLAGS register (équiv. *status*)



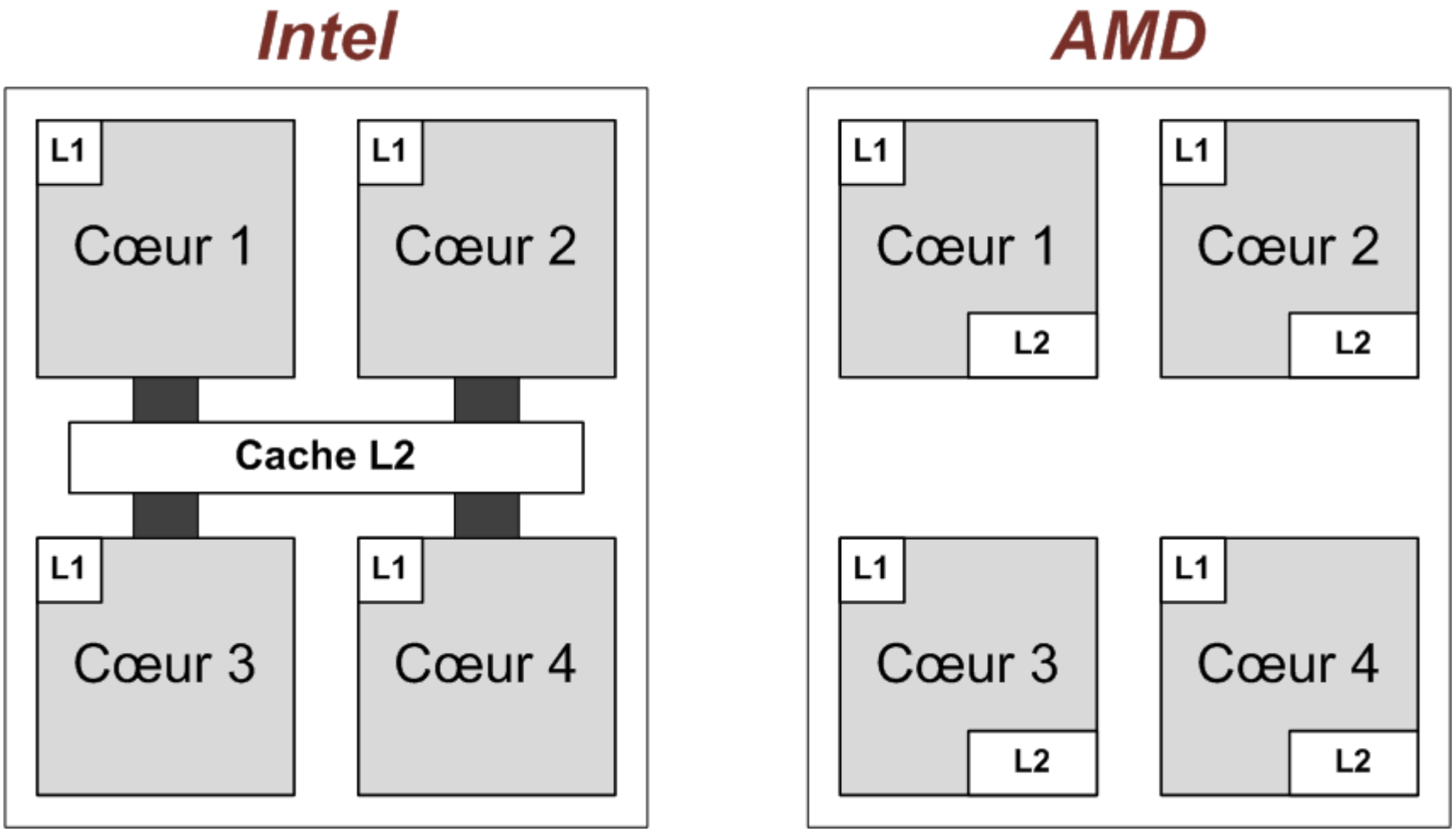
Mémoire

- Architecture mémoire : augmenter performance



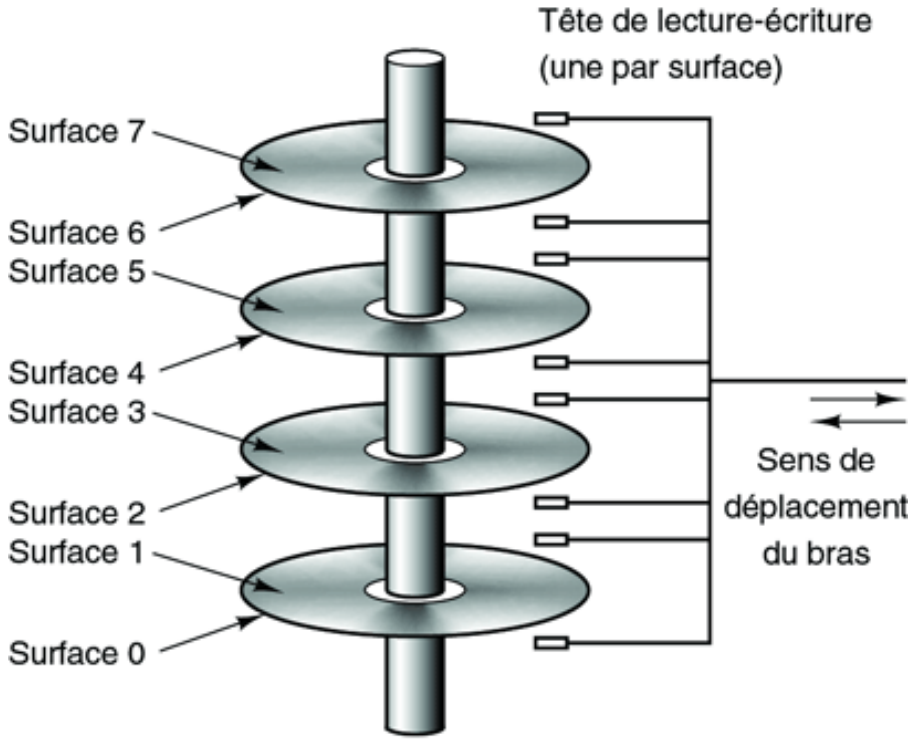
statique : 6 transistors
dynamique : 1 transistor

Mémoire cache : multi-coeurs



Disque

- Disques magnétiques
- Tête lecture-écriture
- Capacité de stockage élevée
  - 4To vs *16 Go RAM*
- Non-volatile
- Temps accès : **~10 ms**
- Débit : 50-160 Mo/s



© Pearson Education France

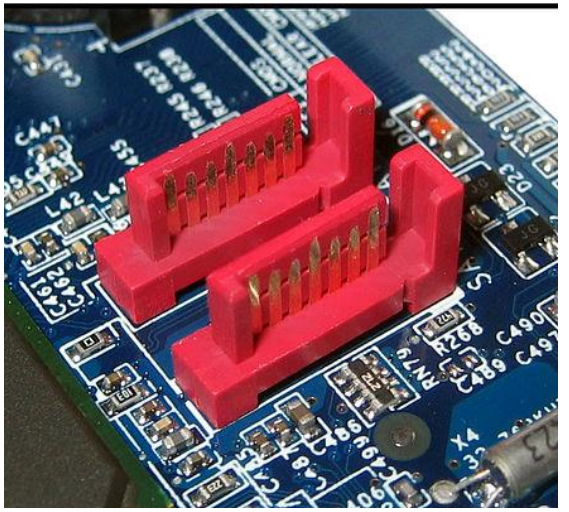
Disque

- Serial ATA : bus pour disques durs sur PC
- Extrait commandes contrôleur pour écritures

WRITE BUFFER
WRITE BUFFER DMA
WRITE DMA
WRITE DMA EXT
WRITE DMA FUA EXT
WRITE DMA QUEUED
WRITE DMA QUEUED EXT
WRITE DMA QUEUE FUA EXT
WRITE DMA WITHOUT RETRIES
WRITE FPDMA QUEUED
WRITE LOG DMA EXT
WRITE LOG EXT
WRITE LONG
WRITE LONG WITHOUT RETRIES
WRITE MULTIPLE
WRITE MULTIPLE EXT
WRITE MULTIPLE FUA EXT
WRITE SECTOR(S)

WRITE SECTOR(S) EXT
WRITE SECTOR(S) WITHOUT RETRIES
WRITE STREAM DMA EXT
WRITE STREAM ERROR LOG
WRITE STREAM EXT
WRITE UNCORRECTABLE EXT

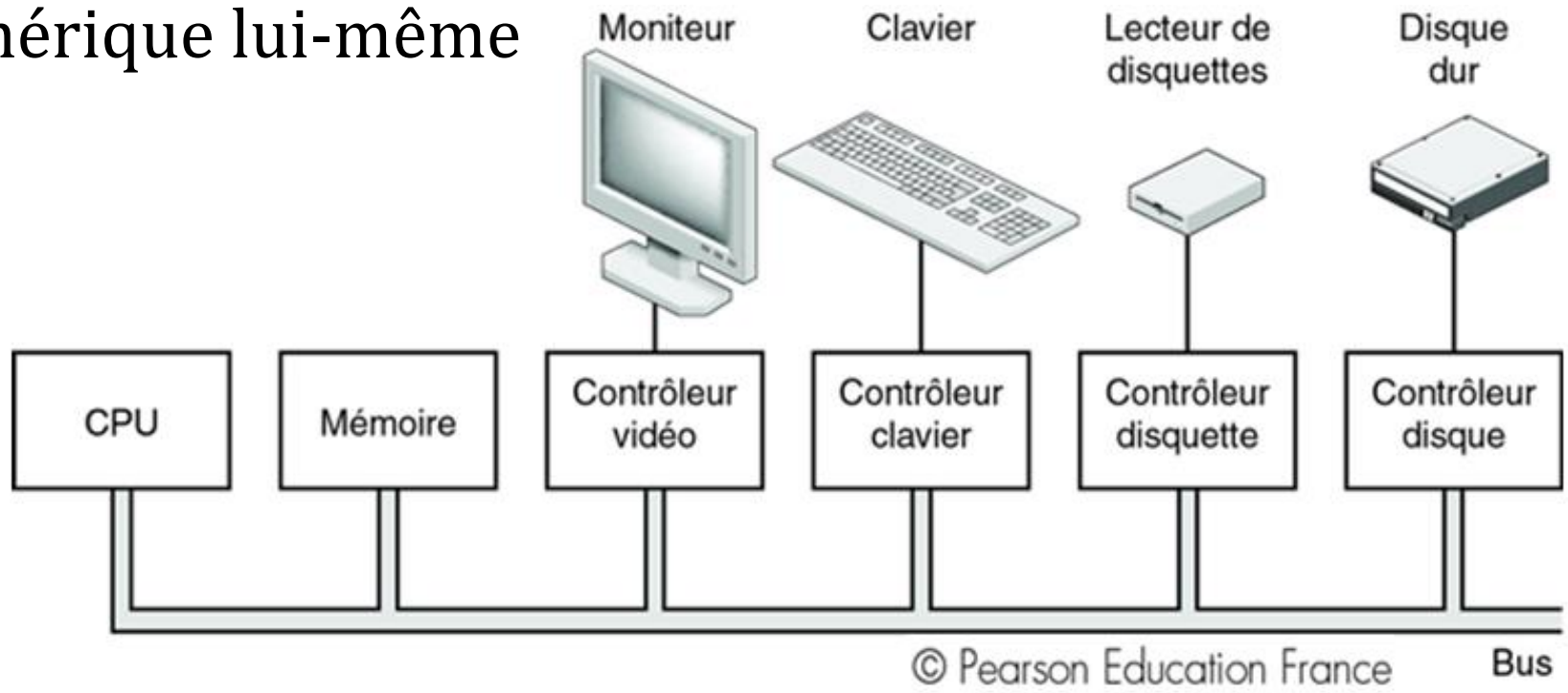
WRITE UNCORRECTABLE EXT_FL
WRITE UNCORRECTABLE EXT_FNL
WRITE UNCORRECTABLE EXT_PL
WRITE UNCORRECTABLE EXT_PNL
WRITE VERIFY



- commande système : **`write()`**

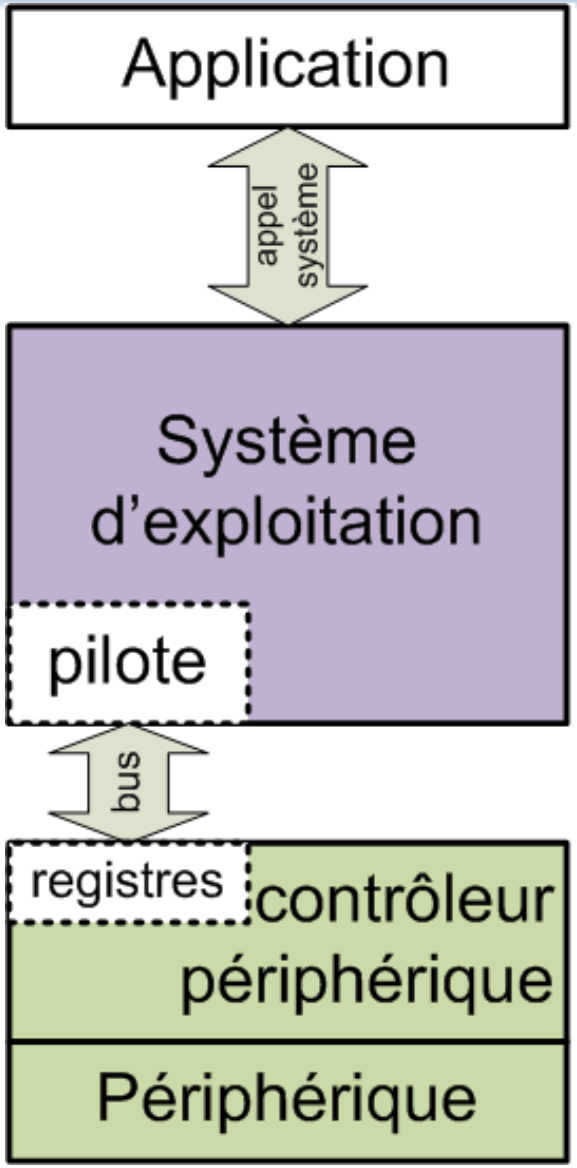
Périphériques d'entrées/sorties (E/S)

- 2 parties :
  - contrôleur (jeu de puces) reçoit les commandes et les exécute
  - périphérique lui-même

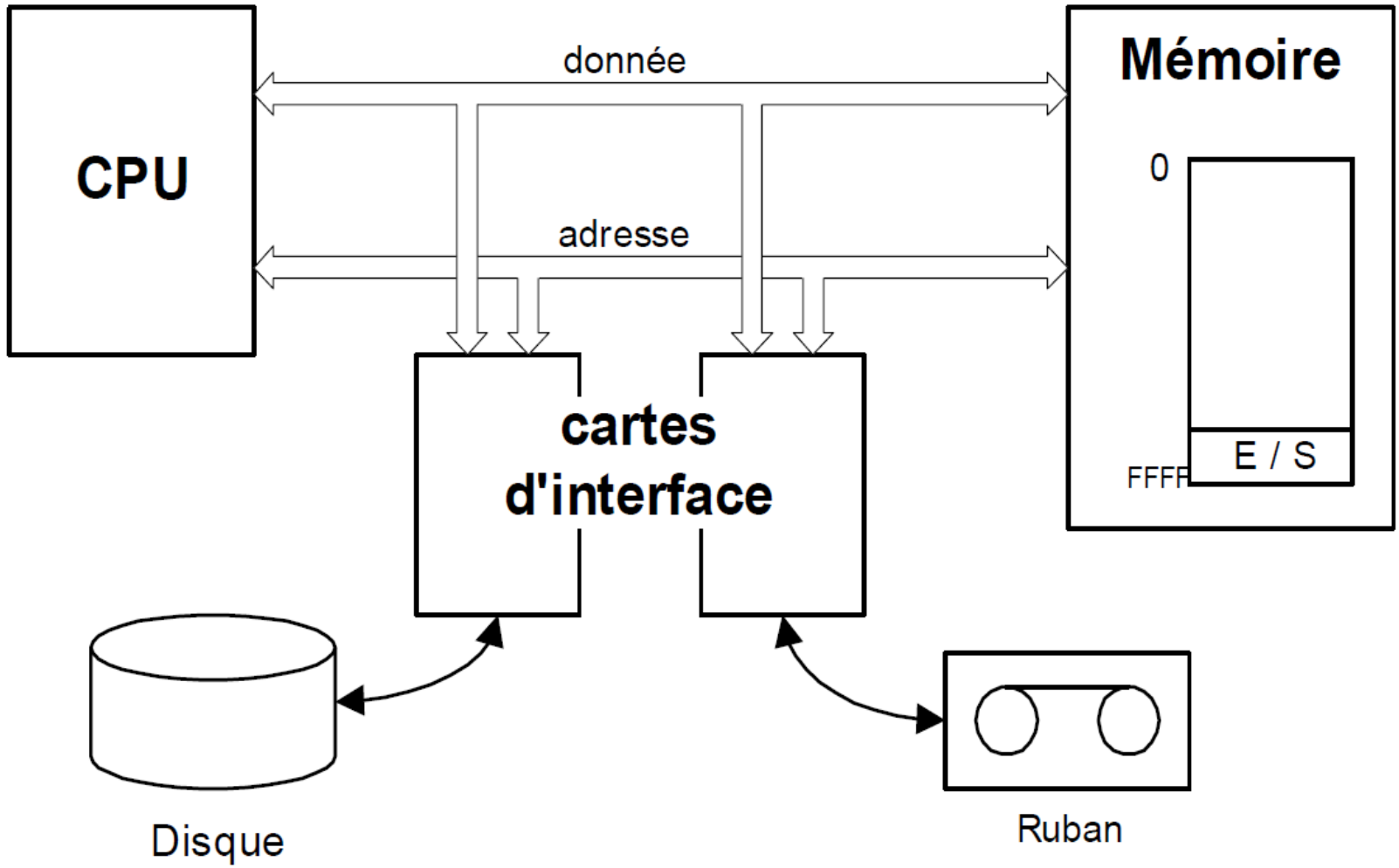


Périphériques d'entrées/sorties (E/S)

- Pilote de périphérique
- Fournis par le manufacturier
  - 1 pilote par système d'exploitation
- Pilote tourne généralement en mode **noyau** (*kernel)*
  - accès sans restrictions aux bus

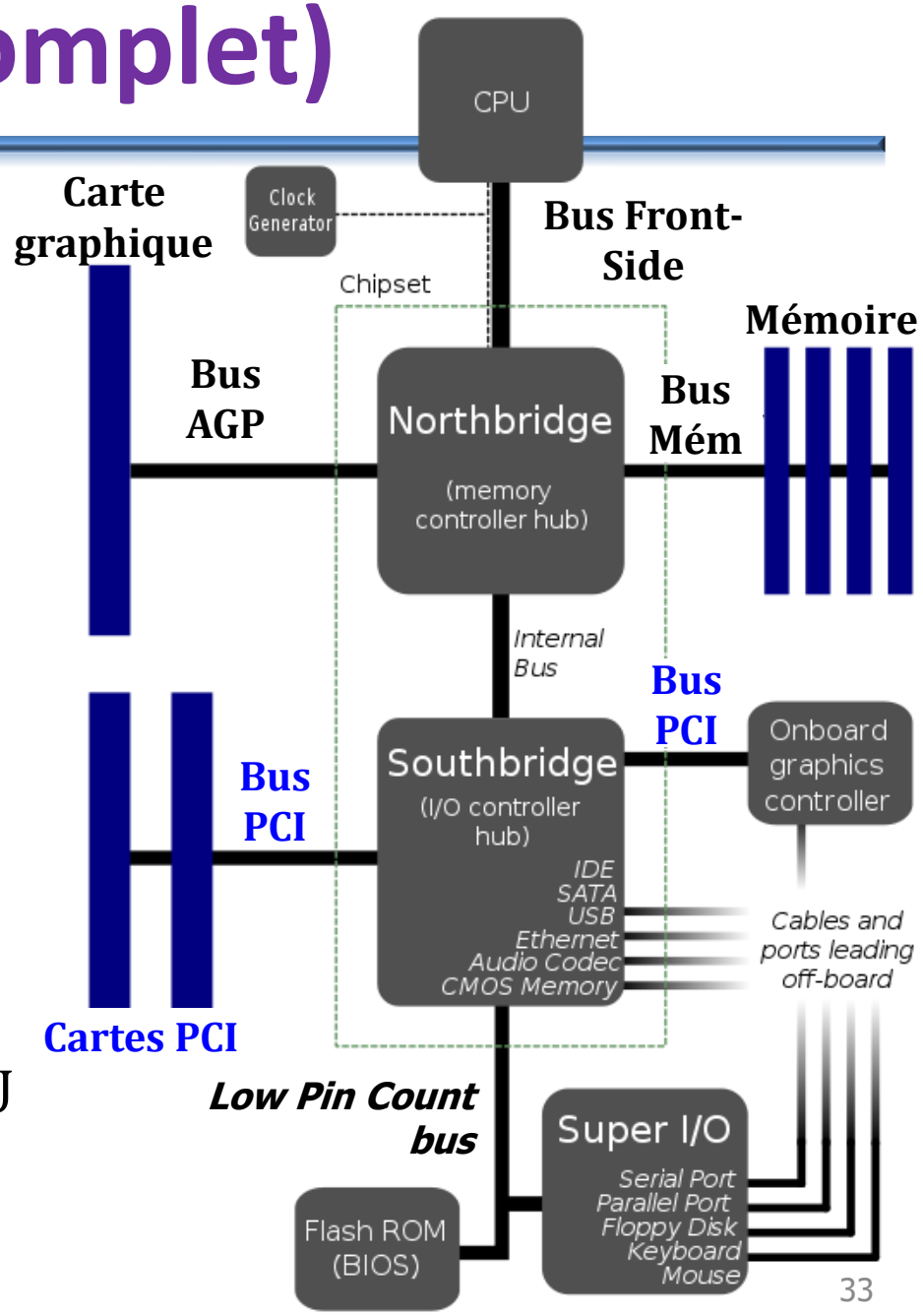


Bus de données (simplifiée)



Bus (+ complet)

- Nombreux bus
  - front-side bus
  - mémoire
  - AGP (graphique)
  - PCI/PCI Express
  - SATA/IDE : disques
  - USB
  - Firewire (IEEE-1394)
  - ISA
- PCI : norme courante
- ISA, SCSI : désuétude
- Cache : presque toujours sur CPU



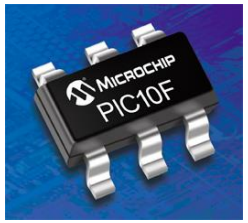
Coévolution matériel et S.E.

- Matériel permet d'offrir ou d'accélérer certaines opérations nécessaires au S.E.
- Exemple de matériel :
  - Mémoire cache
  - Memory Management Unit (MMU)
  - *Translation Look-aside Buffer* (TLB)
  - Anneaux de protections (ring)
  - Adressage par segment
  - Mécanismes d'interruptions (horloge)
  - DMA (Direct Memory Access)

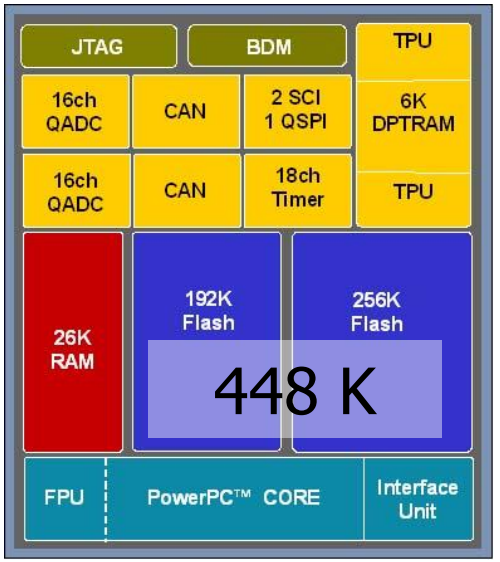
Différentes catégories de S.E.

Quel est le meilleur S.E. ?

- Réponse : aucun!
  - Dépend des applications
  - Dépend de l'ordinateur



PIC 8-bit
256 bytes ROM,
16 bytes RAM



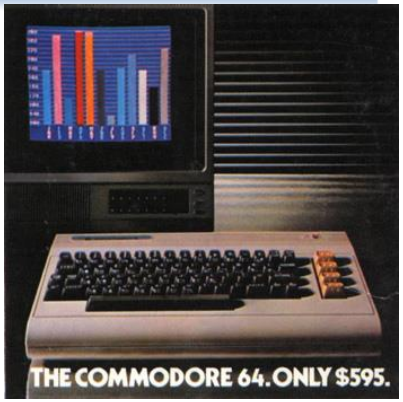
MPC555 40 MHz

z10 d'IBM
77 processeurs et
>1000 GB RAM

- Différentes catégories de S.E.

Catégorie : S.E. des PC

- Certainement les plus familiers
- Monotâche → multitâches
- Importance GUI conviviale
- Support pour nombreux périphériques
- Nombreux logiciels disponibles
- Exemples:
  - Windows XP/Vista/7/8/10
  - MacOS
  - Ubuntu (Linux)

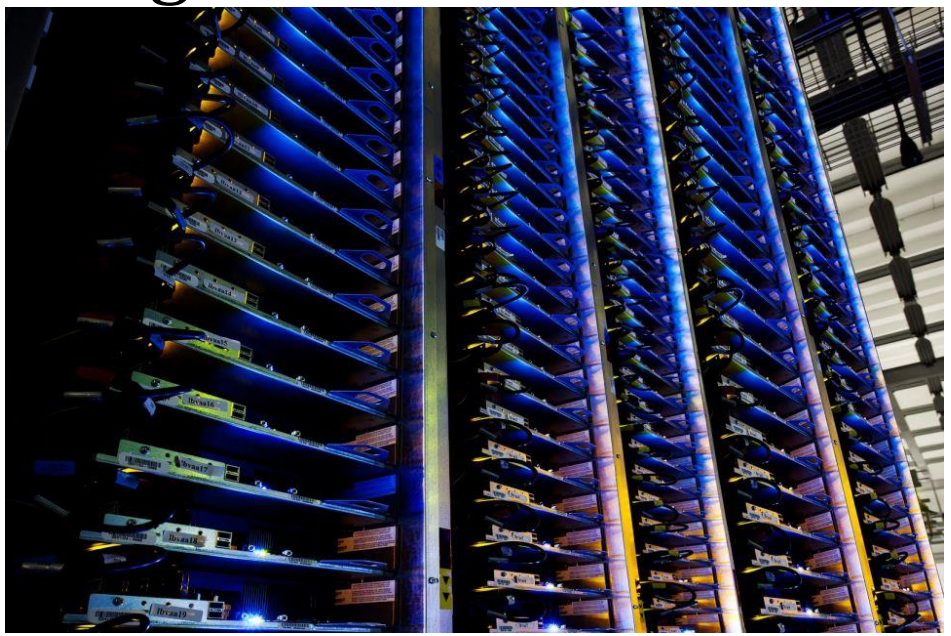


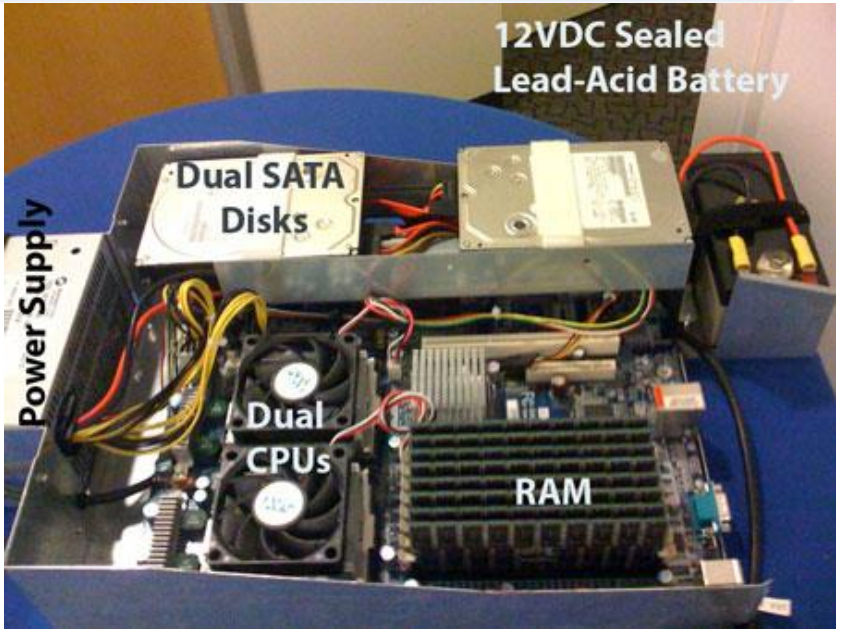
Catégorie : Serveurs

- Application typiques :
  - web
  - fichiers
  - email
- Grand nombre de petites tâches
- Exemples de S.E. :
  - Solaris
  - FreeBSD (Berkeley Software Distribution)
  - Linux
  - Windows Server 20xx

Serveur : ère des *data centers*

- Google : 450, 000 serveurs





- Facebook : > 60,000 serveurs
- Utilisent souvent des versions propriétaires comme S.E. → développent interne des S.E.

(1.3% électricité dans le monde est pour *data centers*)

Catégorie : Mainframes

- Serveurs à très grande capacité
- Grosses tâches (météo)
- Accent sur la fiabilité :
  - banques
  - trafic aérien
- OS/390, LINUX on *System z (IBM)*
- Aussi systèmes anciens
  - VAX/VMS

z10 d'IBM
77 processeurs et
>1000 GB RAM

Catégorie : Systèmes embarqués (*embedded*)

- S.E. simplifiés
  - peu ou pas d'interface usager
  - taille mémoire + CPU réduite
- Peuvent être mono-tâches
- Pas de démarrage de nouveaux processus par l'usager : statique
- Exemples :
  - QNX, VxWorks
  - Kernel Linux + *Busybox*
  - *uClinux, eCos*

Catégorie : Systèmes temps réel

- Met l'accent sur la **prédictibilité** des temps d'exécution
- e.g. exécuter tâche exactement à chaque milliseconde
- Exemples :
  - QNX
  - VxWorks
  - FreeRTOS
  - Real Time Linux

Big Dog de Boston Dynamics

Catégorie : Assistants personnels (PDA)

- Systèmes généralement fermés (pas d'ajout périphérique)
- Petit / CPU limitée / téléphonie / réseau / GPS
- Faible consommation électricité
- Emphase sur l'interface graphique, pas la performance calcul
- Démarrage rapide, utilisation courte

Apple iOS

Palm OS

Google Android

Symbian OS

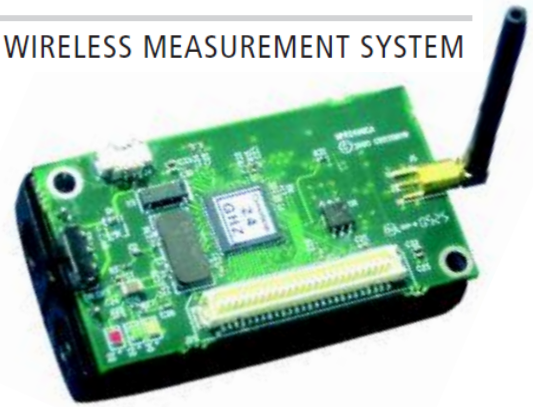
Windows 7 Mobile

Catégorie : Réseaux de capteurs

- Microcontrôleur + batteries + sans-fil + capteurs

MICAz

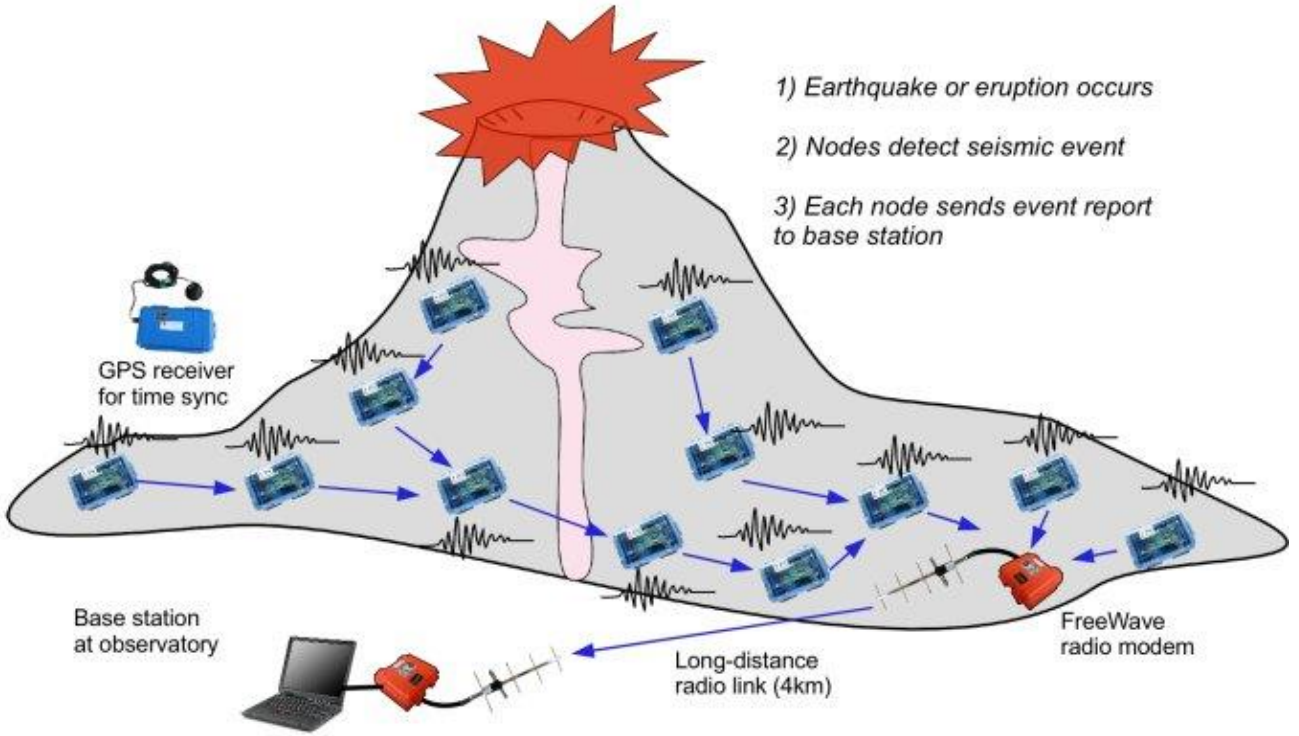
WIRELESS MEASUREMENT SYSTEM



CPU : Atmel ATmega128L

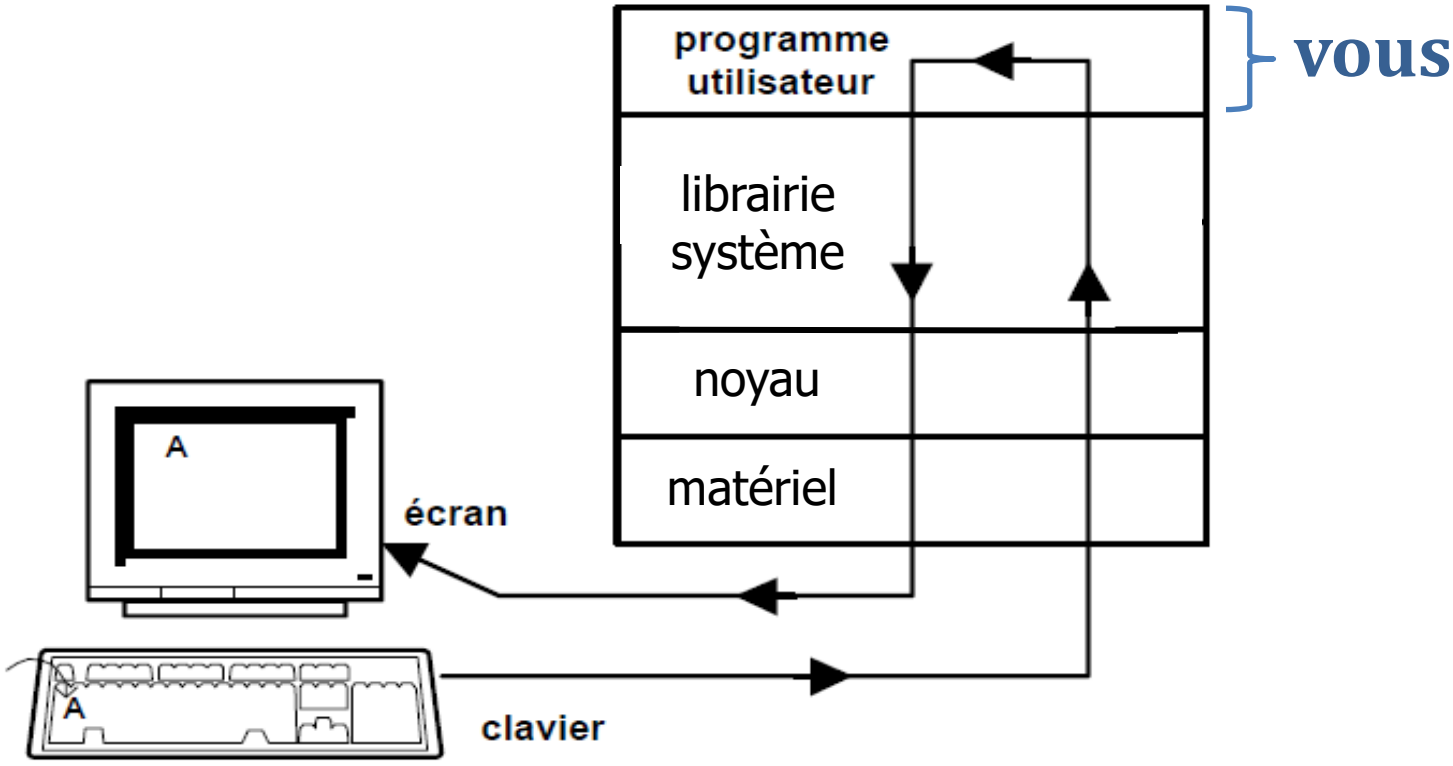
Expansion Connector for Light, Temperature, RH, Barometric Pressure, Acceleration/Seismic, Acoustic, Magnetic and other MEMSIC Sensor Boards

priorité : faible consommation électrique



Structure du système d’exploitation

Structure en couche du S.E.



Les informations traversent les couches du système

Composantes du S.E.

- **Le noyau**
  - partie centrale du S.E.
  - y consacrer la majorité du temps en classe
  - tourne niveau privilège **noyau (0)** du processeur
- **La librairie système**
  - tourne niveau privilège **utilisateur (3)**
  - implémente interface pour C/C++
- **Les programmes systèmes**
  - `ls`, `cp`, `mv`

Le noyau (kernel)

- Partie essentielle du S.E.
- Les fonctionnalités communes des S.E. y sont généralement implémentées
- Beaucoup d'information y réside
  - tableaux gestion processus/mémoire/fichier
- Tourne au niveau de privilège **noyau (0)**
  - Accès à toutes les informations /périphériques
  - Protège S.E. et information contre les utilisateurs
  - « Blindage »

Le noyau (kernel) Linux

- Sur Linux, le noyau (*kernel image*) est le fichier **`/vmlinuz`**

z pour vmlinux compressé avec LZMA ou BZIP2

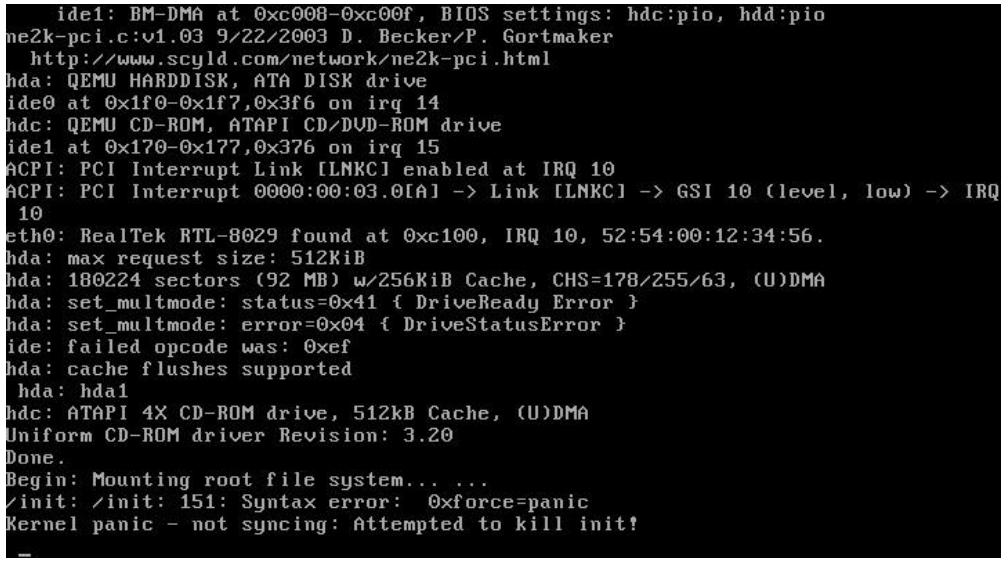
```
root@ubuntu:/# ls -la /vmlinuz
lrwxrwxrwx 1 root root 30 2010-04-27 06:55 /vmlinuz -> boot/vmlinuz-2.6.32-21-generic
```

http://en.wikipedia.org/wiki/Vmlinux

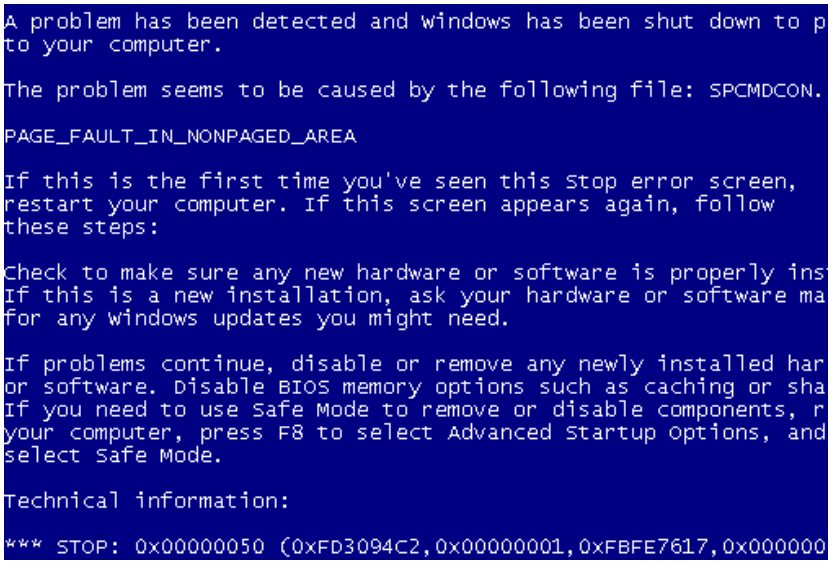
```
root@ubuntu:/boot# ls -la
total 14296
drwxr-xr-x  3 root root    4096 2010-04-27 07:02 .
drwxr-xr-x 22 root root    4096 2010-04-27 06:55 ..
-rw-r--r--  1 root root  640617 2010-04-16 06:01 abi-2.6.32-21-generic
-rw-r--r--  1 root root  115847 2010-04-16 06:01 config-2.6.32-21-generic
drwxr-xr-x  3 root root    4096 2010-04-27 06:55 grub
-rw-r--r--  1 root root 7977158 2010-04-27 07:02 initrd.img-2.6.32-21-generic
-rw-r--r--  1 root root  160280 2010-03-23 02:37 memtest86+.bin
-rw-r--r--  1 root root 1687378 2010-04-16 06:01 System.map-2.6.32-21-generic
-rw-r--r--  1 root root    1196 2010-04-16 06:03 vmcoreinfo-2.6.32-21-generic
-rw-r--r--  1 root root 4029792 2010-04-16 06:01 vmlinuz-2.6.32-21-generic
root@ubuntu:/boot# uname -a
Linux ubuntu 2.6.32-21-generic #32-Ubuntu SMP Fri Apr 16 08:10:02 UTC 2010 i686 GNU/Linux
```

Le noyau (kernel)

- Partie la plus risquée à développer :
  - si une fonction plante dans le noyau, c'est fatal



Kernel panic dans Linux



« blue screen of death » avec Windows

Librairie système Linux

- Point d'accès des applications au S.E.
- GNU C library **`glibc`** est utilisée sur Linux

```
root@ubuntu:~/OS# /lib/libc.so.6
GNU C Library (Ubuntu EGLIBC 2.11.1-0ubuntu7) stable release version 2.11.1, by
Roland McGrath et al.
Copyright (C) 2009 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
Compiled by GNU CC version 4.4.3.
Compiled on a Linux >>2.6.24-27-server<< system on 2010-04-22.
Available extensions:
        crypt add-on version 2.1 by Michael Glad and others
        GNU Libidn by Simon Josefsson
        Native POSIX Threads Library by Ulrich Drepper et al
        BIND-8.2.3-T5B
For bug reporting instructions, please see:
```

Interface Standard POSIX

POSIX: **P**ortable **O**perating **S**ystem **I**nterface for **U**nix

Gestion des processus	
Appel	**Description**
`pid = fork()`	Crée un processus enfant identique au parent
`pid = waitpid(pid, &stat, options)`	Attend la terminaison d'un fils
`s = execve(nom, argv, env)`	Remplace l'image d'un processus
`exit(statut)`	Termine l'exécution du processus et renvoie le statut
Gestion des fichiers	
Appel	**Description**
`df = open(fich, mode,…)`	Ouvre un fichier en lecture et/ou en écriture
`s = close(df)`	Ferme un fichier ouvert
`n = read(df,tampon, nbOctets)`	Lit des données d'un fichier dans un tampon
`n = write(df,tampon, nbOctets)`	Écrit les données d'un tampon dans un fichier
`pos = lseek(df, offset, orig)`	Déplace le pointeur de fichier
`s = stat (nom, &buf)`	Récupère l'information sur un fichier

Interface Standard POSIX

Gestion des répertoires et du système de fichiers	
Appel	**Description**
`s = mkdir(nom, mode)`	Crée un nouveau répertoire
`s = rmdir(nom)`	Supprime un répertoire vide
`s = link(nom1, nom2)`	Crée une nouvelle entrée (nom2) pointant sur nom1
`s = unlink(nom)`	Supprime l'entrée de répertoire correspondante
`s = mount(spec;nom, flags)`	Monte un système de fichiers
`s = umount (spec)`	Démonte un système de fichiers
Autres	
Appel	**Description**
`s = chdir(nomRep)`	Change le répertoire de travail
`s = chmod(nom, mode)`	Change les droits d'accès d'un fichier
`s = kill(pid, signal)`	Envoie un signal à un processus
`sec = time(&sec)`	Renvoie le temps écoulé depuis le 1er janvier 1970

Libraire système Windows

- Les appels aux systèmes d'exploitation sur Windows se font avec le **Windows API** (anciennement API Win32)

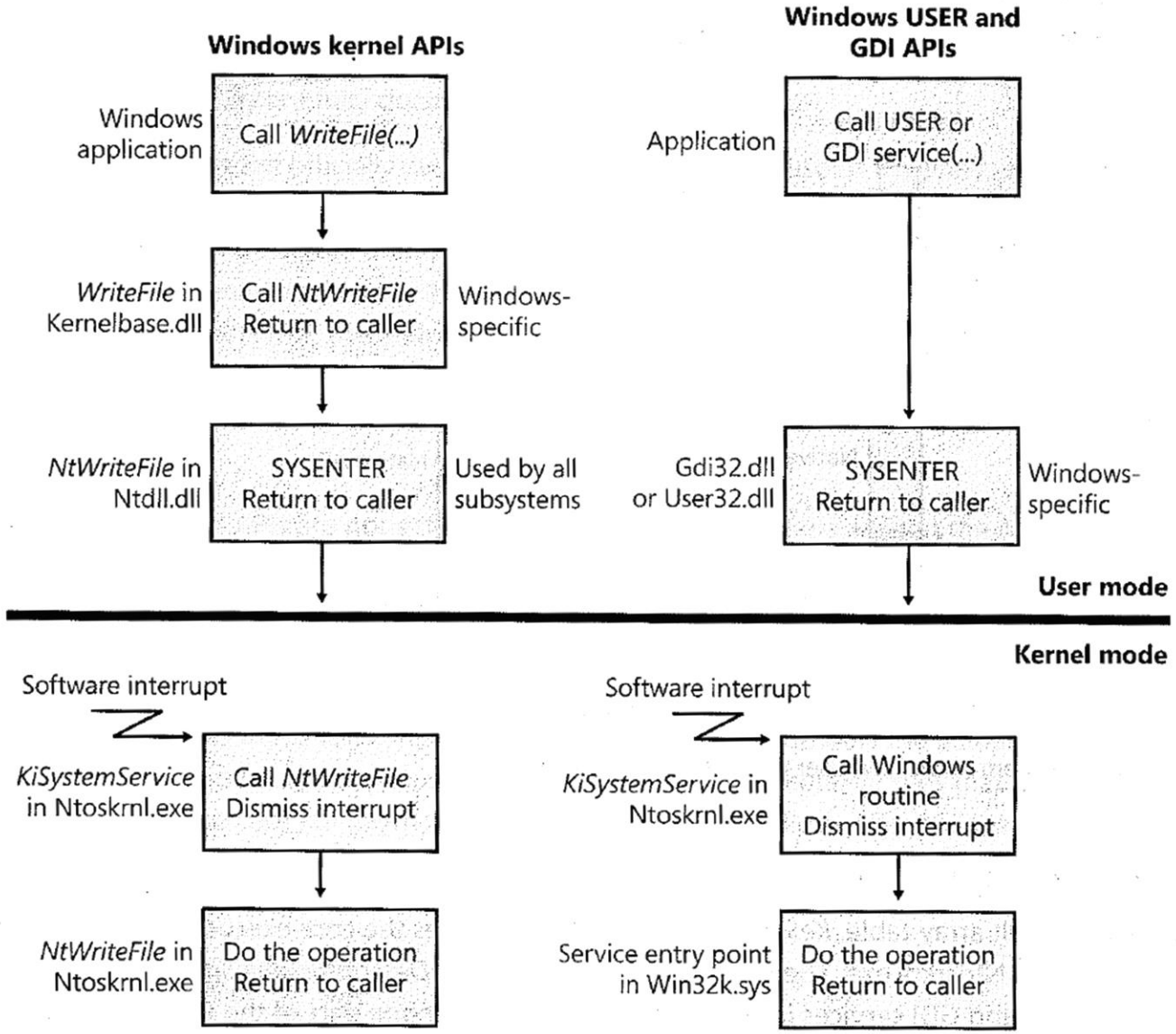


FIGURE 3-17 System service dispatching

Interface Windows API (Win32)

UNIX	Win32	Description
fork	CreateProcess	Crée un nouveau processus
waitpid	WaitForSingleObject	Attend la fin d'un processus
execve	(rien)	CreateProcess = fork + execve
exit	ExitProcess	Termine l'exécution du processus
open	CreateFile	Crée un nouveau fichier (ou en ouvre un existant)
close	CloseHandle	Ferme un fichier
read	ReadFile	Lit des données depuis un fichier
write	WriteFile	Écrit des données dans un fichier
lseek	SetFilePointer	Déplace le pointeur de fichier
stat	GetFileAttributesEx	Trouve l'information sur un fichier
mkdir	CreateDirectory	Crée un nouveau répertoire
rmdir	RemoveDirectory	Supprime un répertoire vide
link	(rien)	Win32 ne supporte pas les liens

Interface Windows API (Win32)

UNIX	Win32	Description
`unlink`	`DeleteFile`	Détruit un fichier existant
`mount`	(rien)	Win32 ne supporte pas le montage
`umount`	(rien)	Win32 ne supporte pas le montage
`chdir`	`SetCurrentDirectory`	Change le répertoire de travail courant
`chmod`	(rien)	Win32 ne supporte pas la sécurité (bien que NT le fasse)
`kill`	(rien)	Win32 ne supporte pas les signaux
`time`	`GetLocalTime`	Trouve l'heure courante