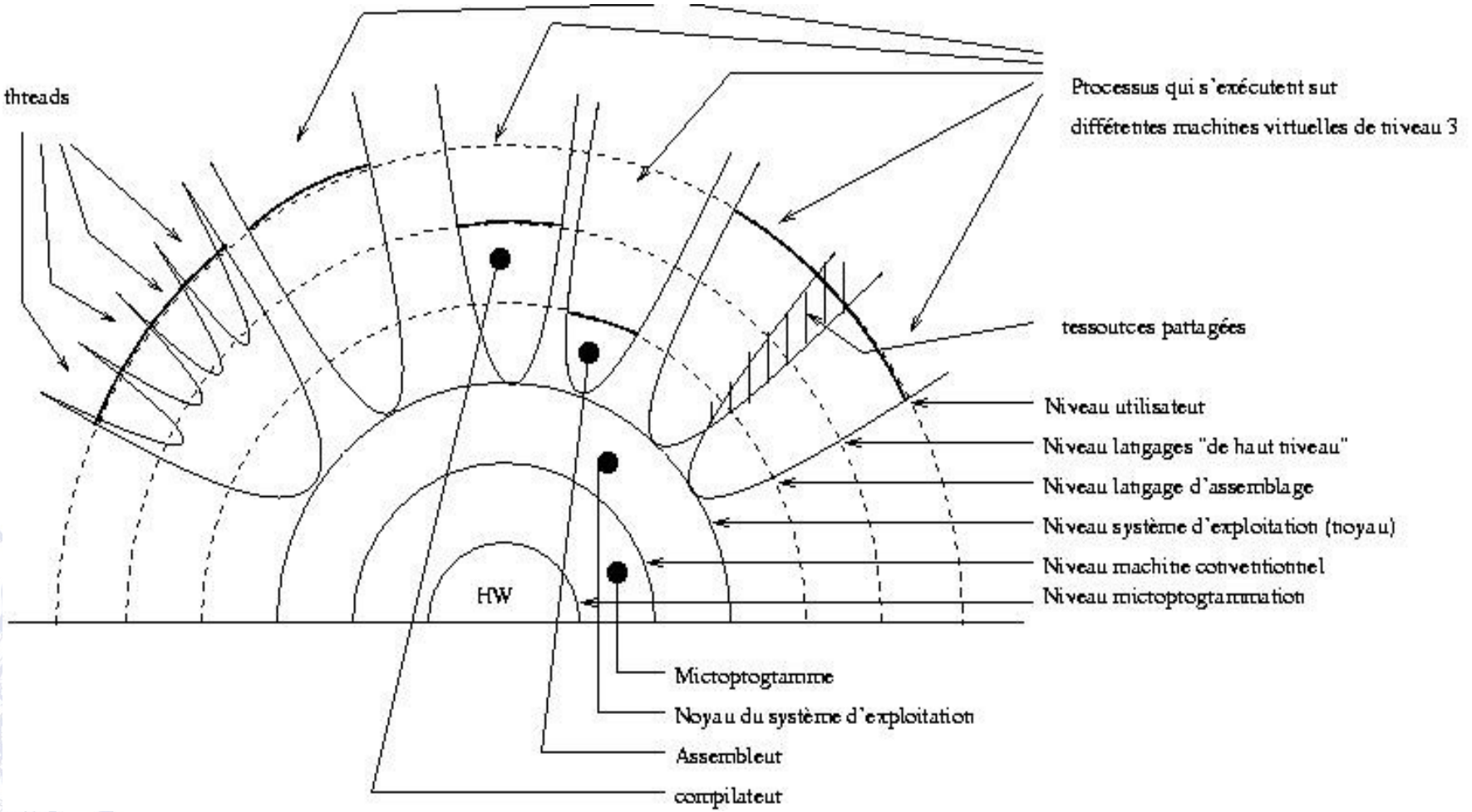
LINGI 1113: Systèmes informatiques 2

Mission 2: sujets choisis d'architecture et introduction aux systèmes d'exploitation

ÉCOLE
POLYTECHNIQUE
DE LOUVAIN

UCL

Architecture



Architecture
focus: ce qui sert à l'OS

1. interruptions

- **Processor status and control register**
  - Bits d'état pour sauts conditionnels
  - Bit indiquant si on est en “mode 2”(OS) ou “mode 3” (processus d'application)
  - Bit indiquant si les interruptions sont autorisées
  - Bits indiquant le niveau de priorité courant du processeur

- **Interruptions** (et traps ou exceptions et appels systèmes): événement provoquant l'exécution d'une fonction (au sens C) de l'OS sans passage d'arguments ni de résultats

Architecture
interruptions: l'événement déclencheur

- Interne au processeur: détection d'une erreur
  - Division par zéro
  - Code opératoire inexistant ou interdit
  - Accès à zone interdite de mémoire

  On parle de **trap** ou **exception**
- Provenant d'un contrôleur de périphérique
  - Opération ou transfert terminé
  - Erreur lors du transfert

  On parle d'**interrupt (IRQ ou NMI)**
- Provenant du programme
  - Demande d'exécution d'un appel système par un programme

  On parle de **software interrupt (SWI ou INT)**, de **syscall**

Architecture
interruptions: l'événement déclencheur

- Détection de l'événement déclencheur:

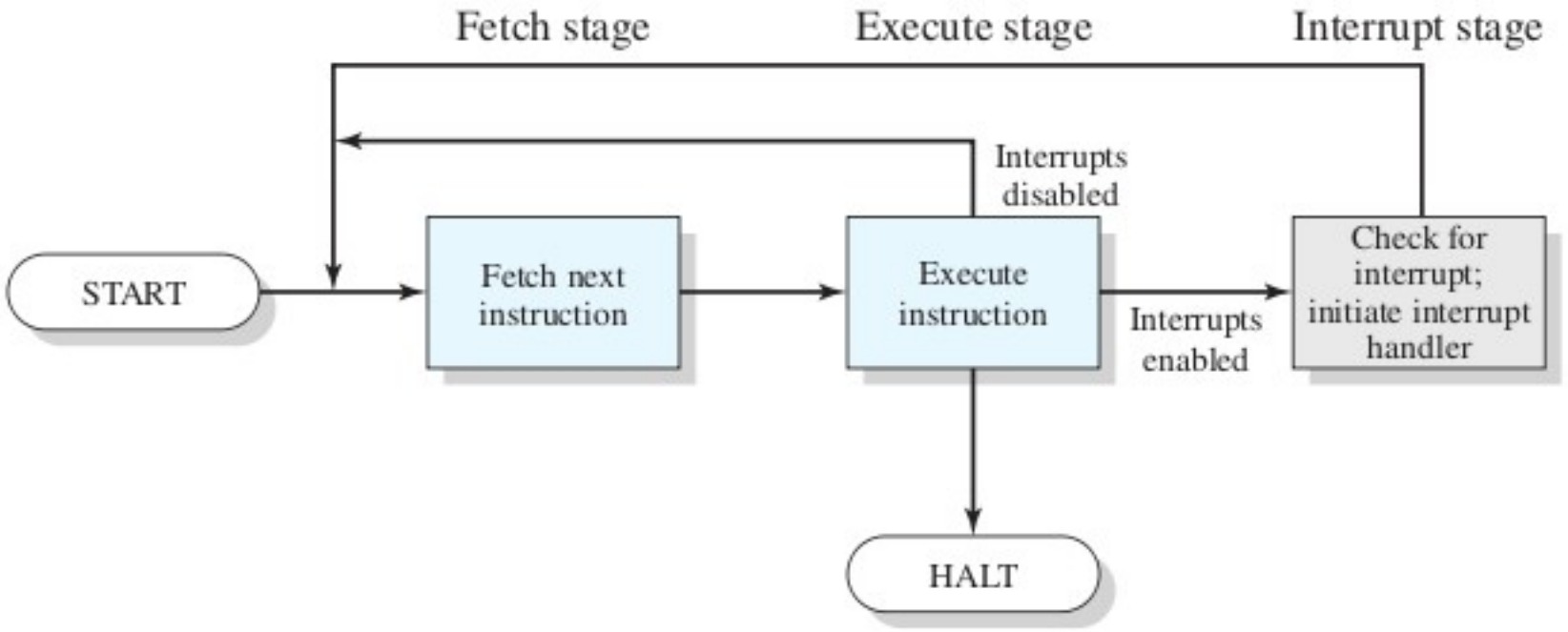


Figure 1.7 **Instruction Cycle with Interrupts**

Architecture
interruptions: mécanisme

- Comme fonction “normale”:
  - Sauver PC quelque part (pile ou registre)
  - Charger nouvelle valeur dans compteur de programme
  - Exécuter code
  - Reprendre programme appelant
- Différences avec fonction”normale”:
  - La source (processeur, périph, programme)
    - ne peut pas choisir adresse de la fonction
    - ne peut pas passer d'arguments
    - ne reçoit pas de résultat
  - La fonction s'exécute en mode système quelque soit le mode du prog. interrompu

Architecture
interruptions: mécanisme

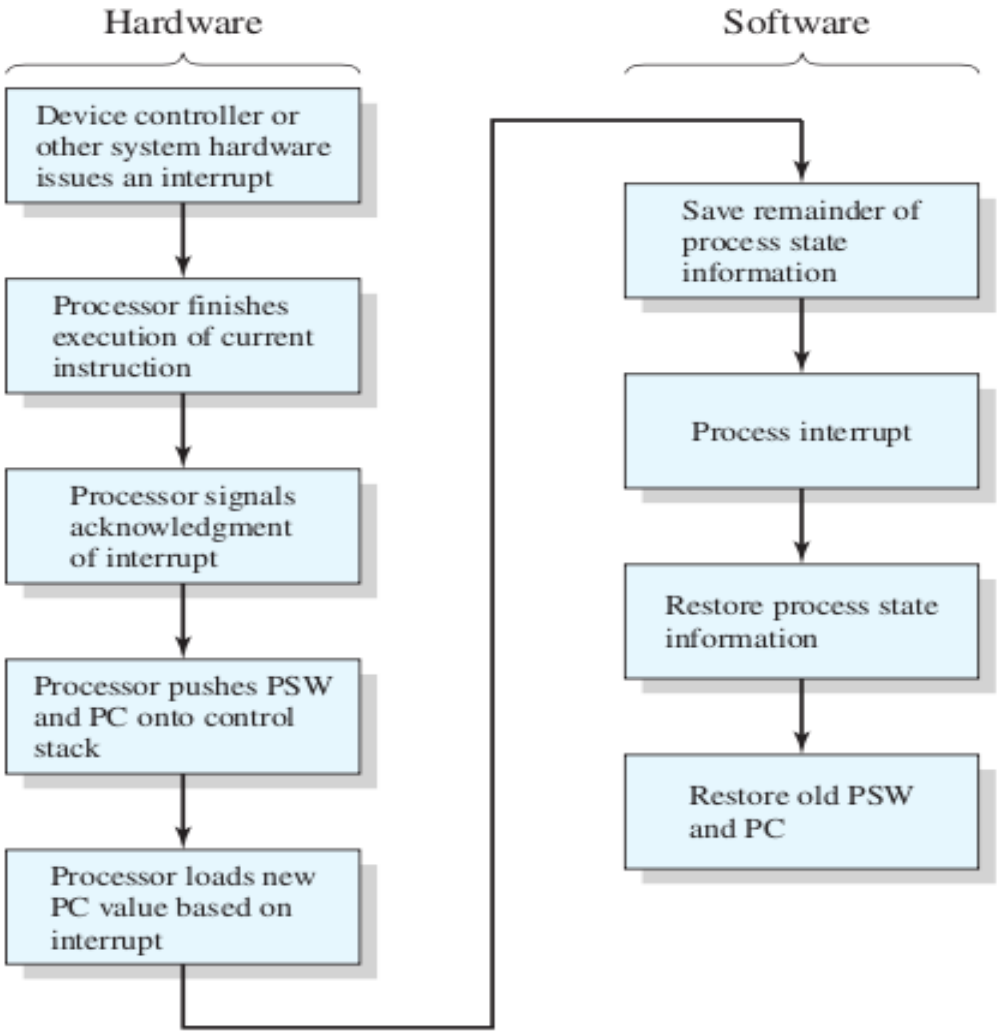


Figure 1.10 **Simple Interrupt Processing**

Rappels d'architecture interruptions: mécanisme

Qui choisit l'adresse de la fonction ?

Plusieurs possibilités:

- Toujours la même fonction qui s'exécute,
  - soit son adresse est soit imposée (p. ex 0000000)
  - soit l'adresse de l'adresse est imposée

  =>la fonction doit d'abord déterminer qui l'appelle
  (comme le génie de la lampe)
- Pour chaque type d'événement, il y a une adresse qui indique l'adresse de la fonction a exécuter (table de vecteurs d'interruption)

Architecture
interruptions: mécanisme

La fonction, appelée “routine d'interruption” doit savoir où aller chercher et déposer elle même paramètres et résultats

- Registres de contrôleurs de périphériques ou du processeur
- Structures de données de l'OS
- Structures de données du programme appelant (appel système en C: fonction C qui copie ses paramètres dans la structure de données où la fonction système vient les chercher; l'inverse pour les résultats)

Architecture
interruptions: mécanisme

Conditions d'acceptation d'une interruption

- Bit **interrupt enable** du registre de contrôle et d'état du processeur = “vrai”
- (Bit interrupt enable du périphérique = “vrai”)
- Niveau de priorité de la demande > niveau de priorité inscrit dans registre de contrôle et d'état du processeur
- Certaines interruptions sont non masquables (toujours acceptées): traps, NMI, RESET

Architecture
interruptions: mécanisme

Identification de la source de l 'interruption

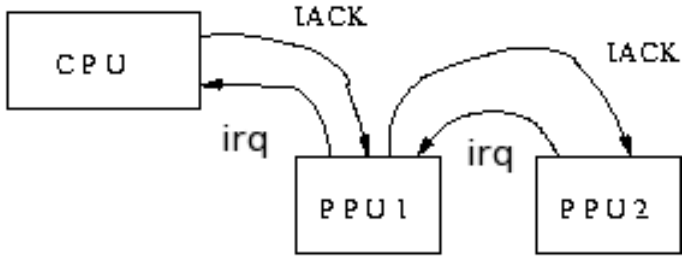
Sources internes: processeur ou programme: le processeur a les informations en lui-même

Sources externes: il faut identifier le contrôleur de périphérique qui demande l'interruption: la demande est un signal électrique envoyé au CPU

1. Une seule ligne pour tous les contrôleurs et identification de la source par software
2. Une ligne par périphérique (15 sur les PCs)
3. Une seule ligne, puis le CPU demande au périphérique son code d'identification à envoyer sur bus de données

Architecture
interruptions: mécanisme

3. Probléme si plusieurs sources demandent interruption en même temps: à qui répondre .
   solution : daisy chain



- Priorité: au lieu d'une ligne irq, plusieurs
  - soit de priorités différentes,
  - soit de priorité codée.

2. hardware nécessaire à l'OS

- Mécanisme pour les appels systèmes= bit MODE 2/3 du PSCR + SWI
- Moyen de protéger la mémoire: empêcher chaque processus d'aller dans les zones de la mémoire centrale attribuées à d'autres:
  - Autres processus
  - Noyau de l'OS lui-même

Sans ces 2 fonctionnalités , l'OS est à la merci des applications !

Solution actuelle de protection de la mémoire: **système de traduction dynamique d'adresses**

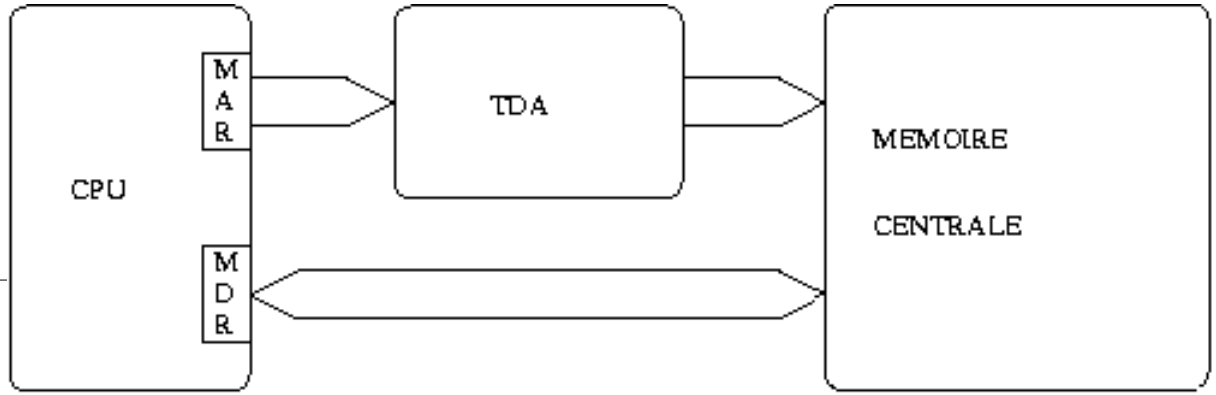
Architecture
système de traduction dynamique d'adresses

But principal:

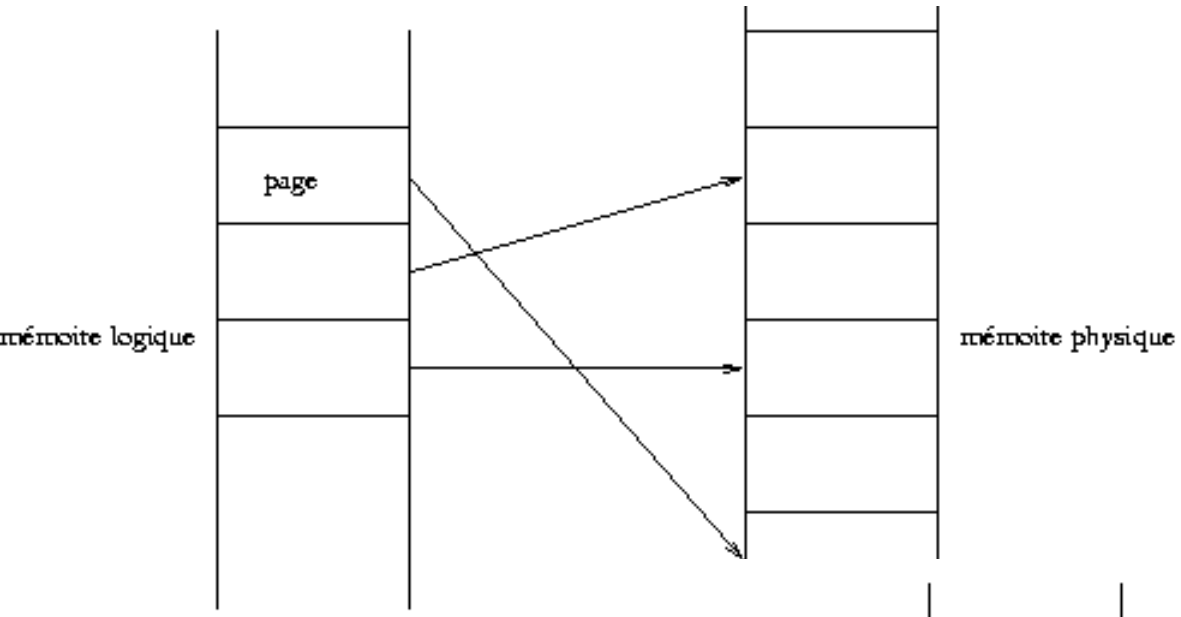
- Ne pas devoir refaire la “relocation” de chaque processus avant de la charger en mémoire.

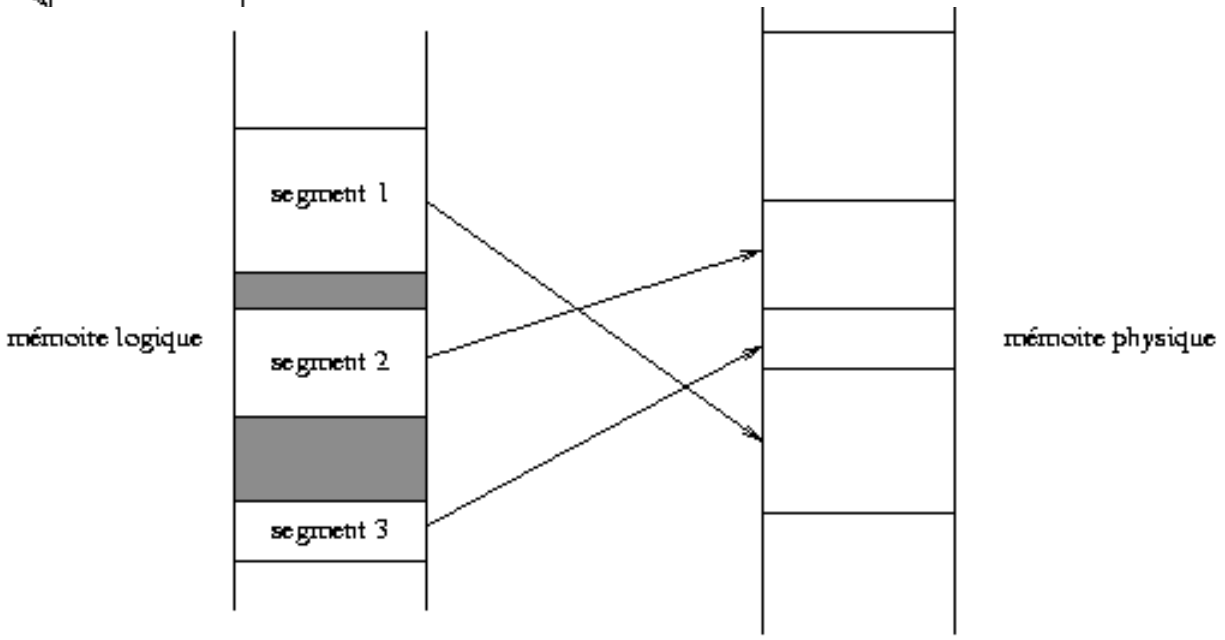
Buts secondaires:

- Pouvoir allouer facilement de la mémoire à un processus
  - Pas forcément contiguë
  - Eventuellement au milieu de l'espace d'adressage déjà alloué
- Empêcher le processus d'accéder à la mémoire qui ne lui a pas été allouée.



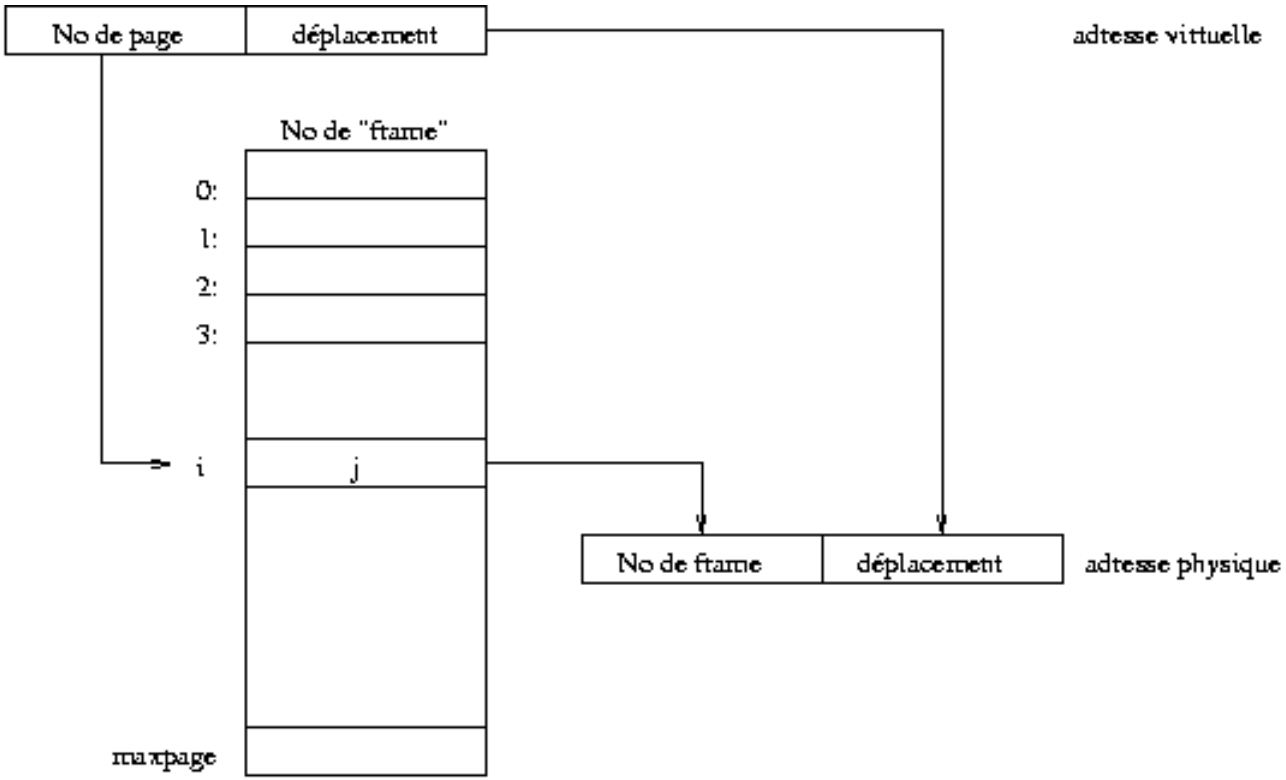
Architecture
système de traduction dynamique d'adresses





Architecture
système de traduction dynamique d'adresses

Comment ça marche:TDA paginé (plus de détails en mission4)



Architecture
La hiérarchie des mémoires

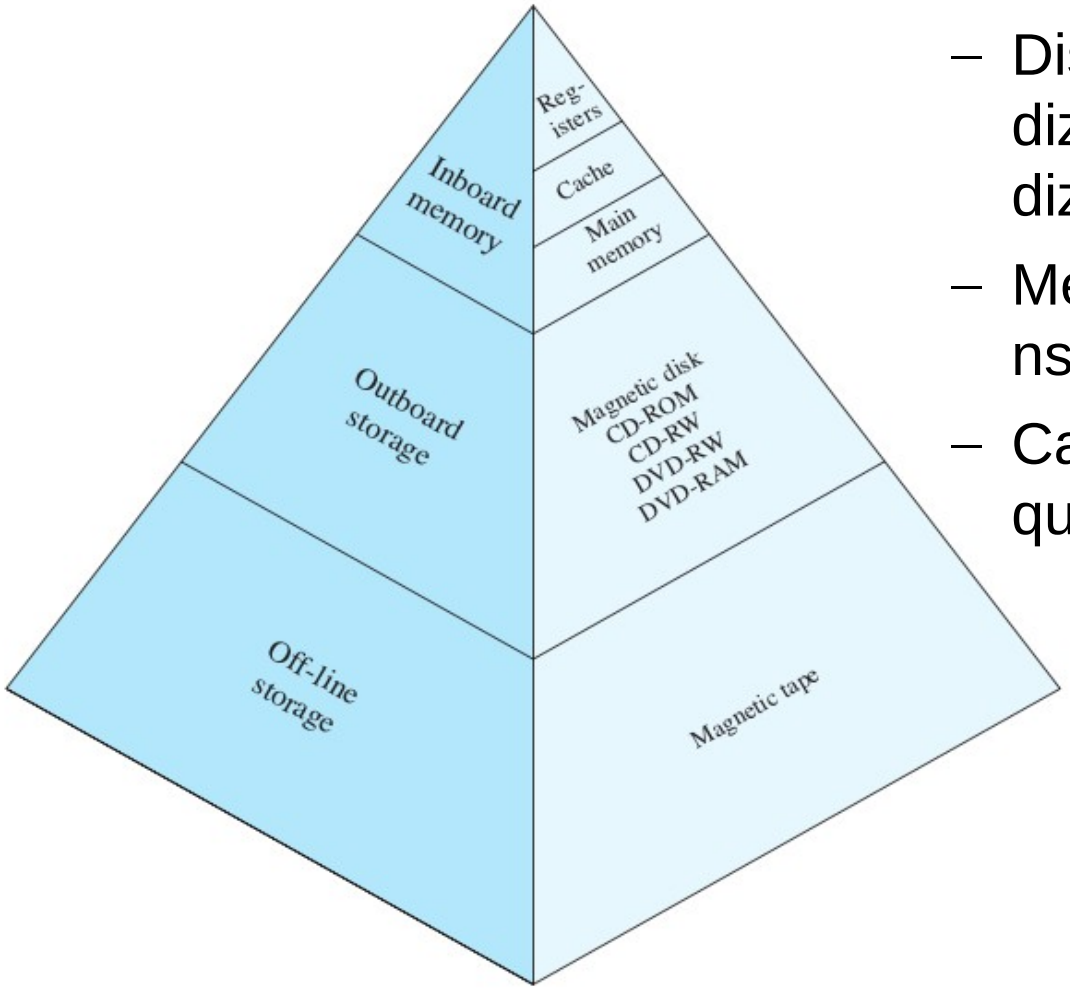


Figure 1.14 The Memory Hierarchy

Hiérarchie des mémoires:

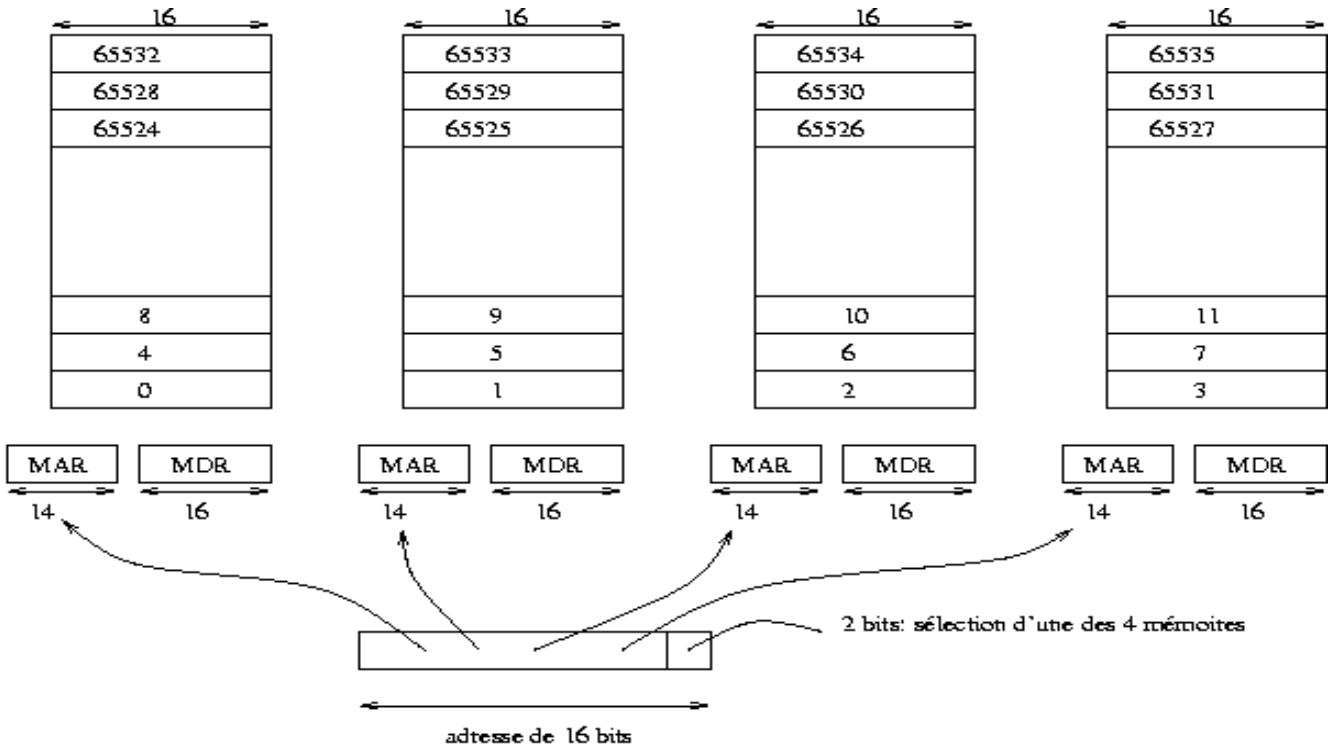
- Disques: jusqu'à 3 térabytes, dizaines à centaines d'euros, dizaine de msec
- Mémoire centrale: dizaines de nsec
- Caches: ordre de la nsec ou quelques nsec

Architecture
La hiérarchie des mémoires

- Magnetic tapes, disques etc: cfr Mission 6, entrées-sorties et systèmes de fichiers
- Ici, on se concentre sur la relative lenteur de la mémoire centrale par rapport aux CPU et aux dispositifs accélérateurs

Architecture
accélération des transferts mémoire-cpu

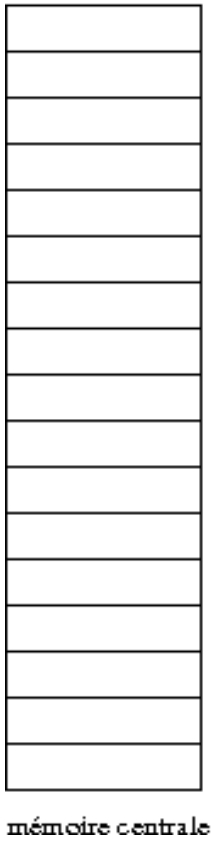
Si on ne peut transférer plus vite, transférer plus à la fois: mémoires entrelacées (~RAID0 sur les disques)

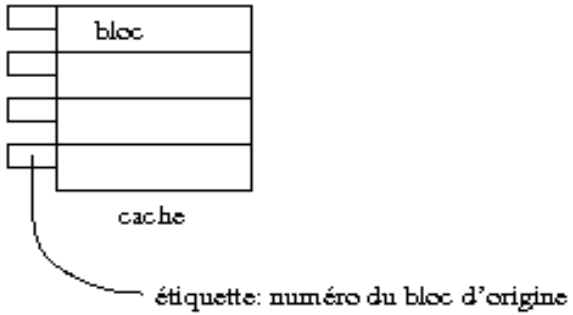


les 14 bits de gauche de l'adresse sont envoyés simultanément comme adresse aux 4 mémoires

Architecture
accélération des transferts mémoire-cpu

Le principe de la cache: mémoire associative

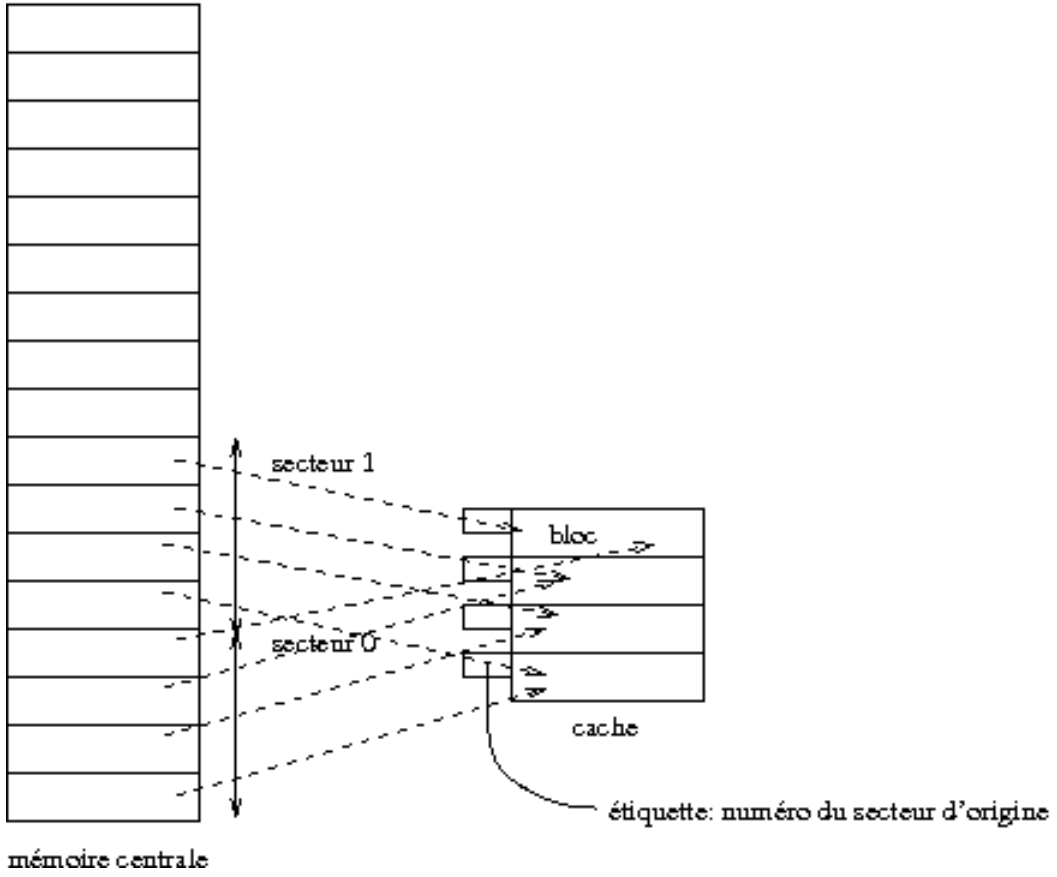




un comparateur par étiquette

Architecture
accélération des transferts mémoire-cpu

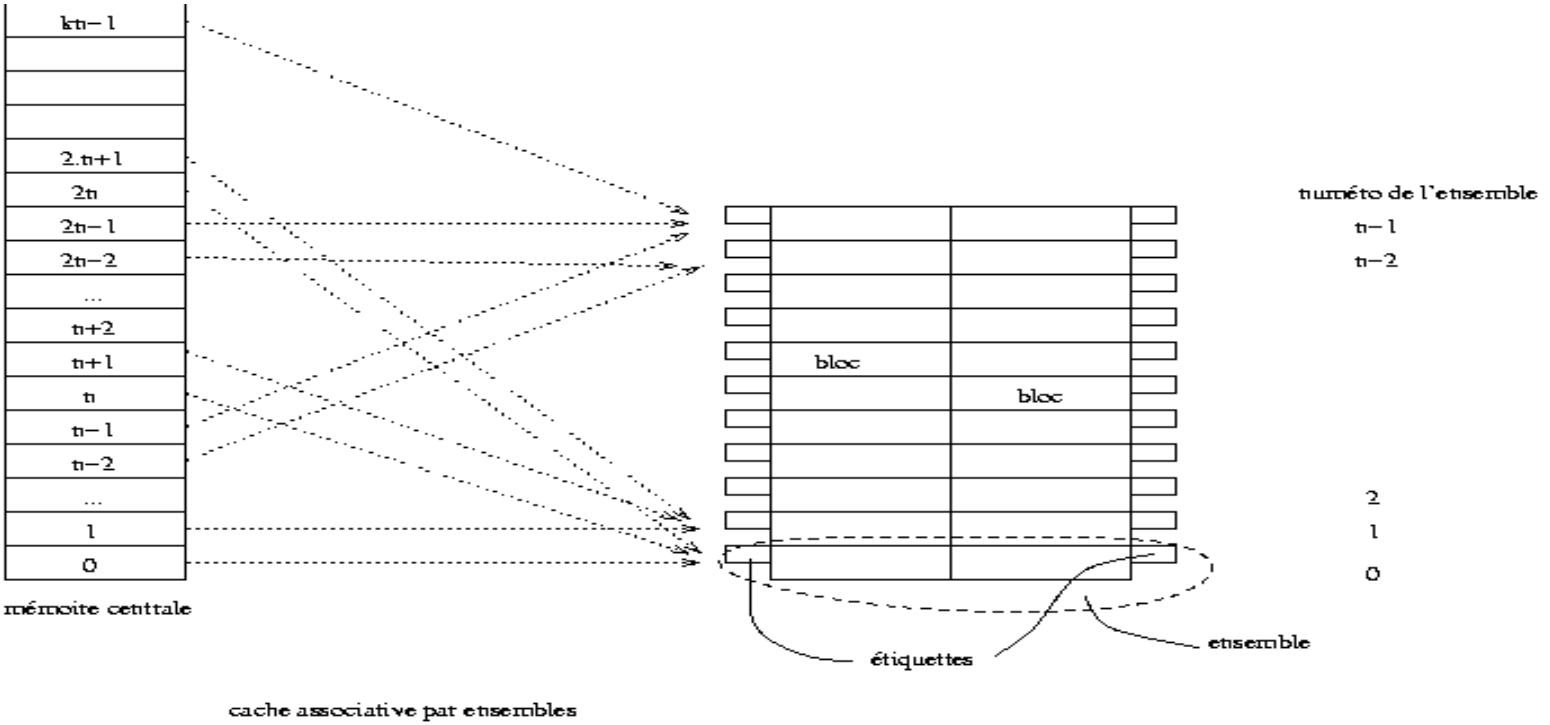
La réalisation “low cost” de la cache: cache à correspondance directe



un seul comparateur

Architecture
accélération des transferts mémoire-cpu

La réalisation de la cache: le bon compromis: cache associative par ensembles



cache associative par ensembles

un comparateur par ensemble

Qu'appelle-t-on système d'exploitation: ensemble de programmes:

- Noyau et processus noyaux: ce qui ne peut se faire qu'en mode système
  - On peut réduire le noyau à un micronoyau et faire faire un maximum de travail par des processus serveurs:
    plus facile, plus sûr, plus lourd
- Programmes utilitaires fournis avec le noyau: gestionnaire de fenêtres, gestionnaire d'imprimantes, éditeurs, parfois compilateurs et autres outils de développement

Introduction aux systèmes d'exploitation

Les services du système d'exploitation:

- Lancer l'exécution des programmes (shell+noyau)
- Offrir une interface simple aux périphériques (noyau + serveurs spécialisés dans certains cas (imprimantes)
- Gérer l'accès aux fichiers et leur placement sur support non volatil (noyau)
- Gérer les utilisateurs (noyau et programmes spécialisés, comme login)
- Gérer l'utilisation des ressources (coeurs, mémoire,...) par les processus
- Gérer les erreurs et statistiques d'utilisation

Introduction aux systèmes d'exploitation

Ce que vous trouverez dans le chapitre 2:

- Perspective historique de l'évolution des OS
- Introduction aux principales fonctionalités
- Evolutions « récentes » des OS
  (certains les avaient il y a 25 ans, d'autres tout juste ou pas du tout)
  - Multithreading
  - Gestion de multiprocesseurs et multicoeurs
  - Systèmes d'exploitations répartis
  - Micro-noyaux
  - Conception orientée objet
  - Virtualisation
- Exemples d'OS actuels: Windows, UNIX, LINUX