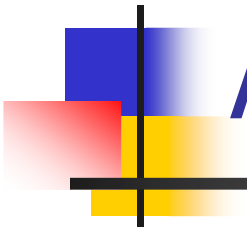
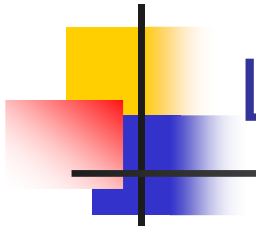


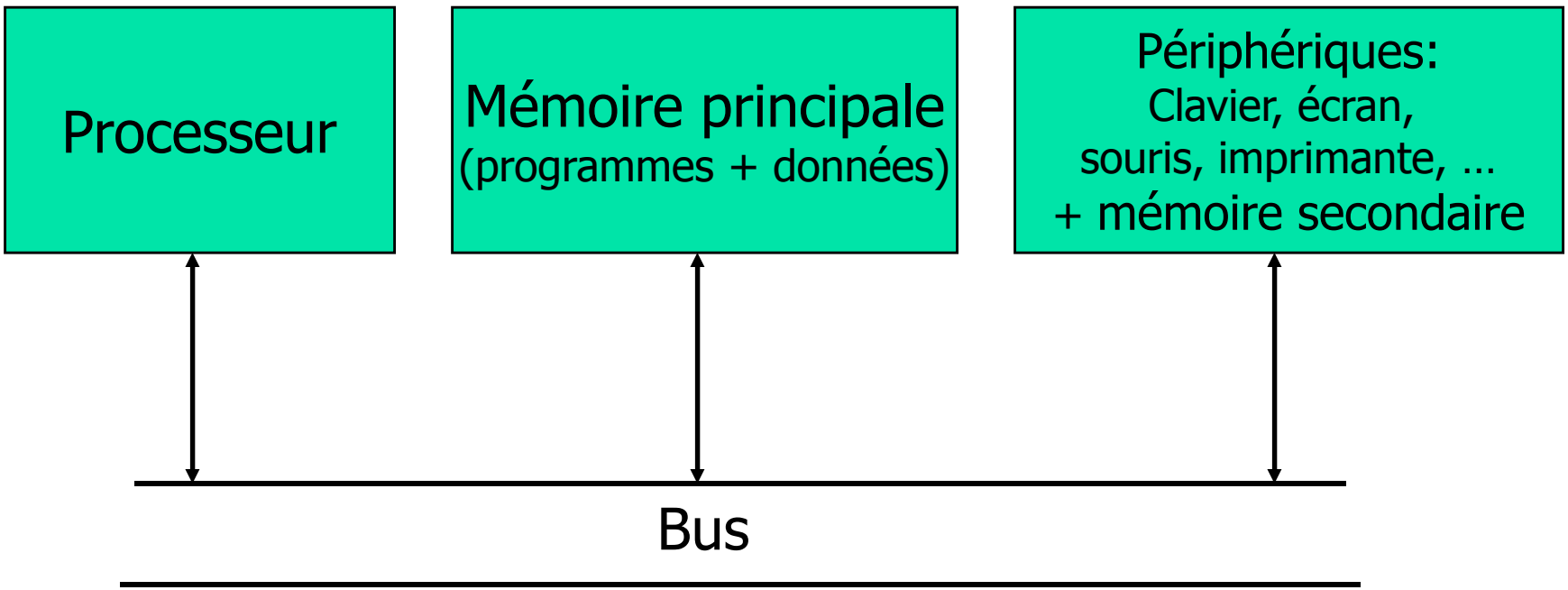
Architecture des ordinateurs



Les composants principaux

- Le processeur
- Les mémoires
- Les périphériques
- Les bus

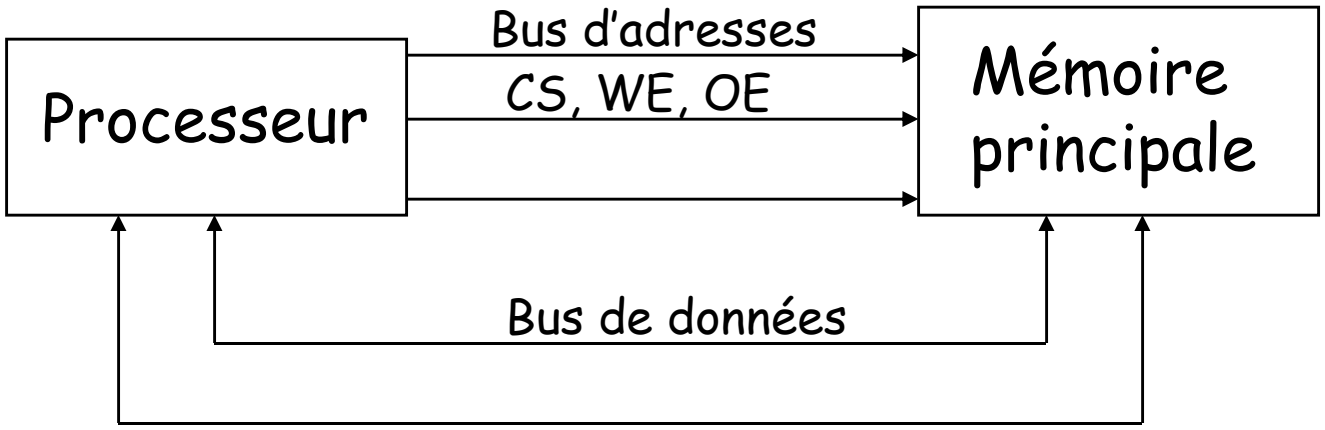
Principe de fonctionnement



I. Bus

Bus

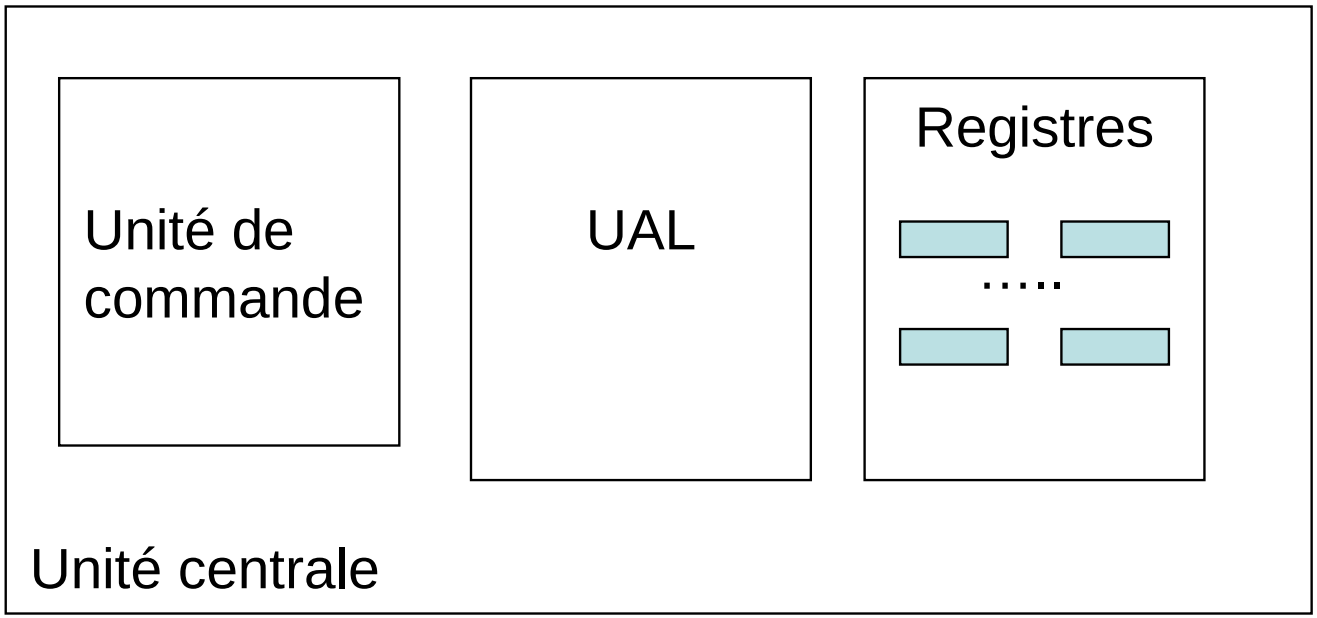
- Communication entre les composants d'un ordinateur
- Bus d'adresses : unidirectionnel
  - processeur -> mémoire, m fils conducteurs pour des adresses de 0 à 2^m-1
- Bus de données : bidirectionnels
  - Lecture : mémoire -> processeur, écriture : processeur -> mémoire
- Bus de contrôle: unidirectionnel
  - interruptions matérielles, indicateur lecture/écriture ...

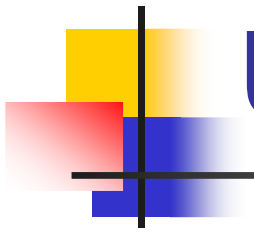


II. Processeur

Le processeur (unité centrale, CPU)

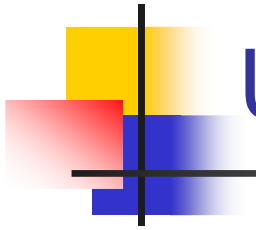
- Circuit électronique complexe qui exécute les instructions
- Cycle d'horloge : fréquence en Hz (voire MHz)
- Organisation:





Unité de commande

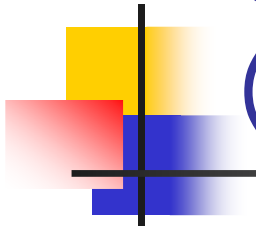
- Décodage de l'instruction
- Calcul des données à traiter
- Contrôle de l'unité de calcul (UAL) pour l'exécution de l'instruction



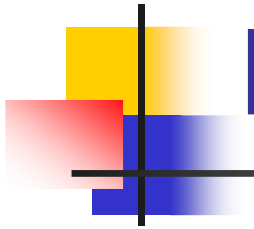
Unité arithmétique et logique (UAL)

- Chargée de l'exécution
  - des opérations booléennes
  - des opérations arithmétiques (addition, soustraction, multiplication, division, comparaison, etc.) pour des entiers

Unité de calcul en virgule flottante (FPU)

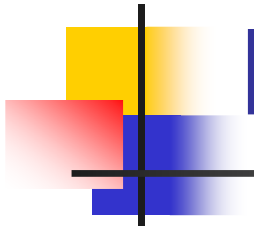


- Spécialement conçue pour effectuer des opérations sur les réels
- optionnel: En son absence, le processeur utilise du microcode pour émuler les fonctions de la FPU en utilisant l'UAL



Registres

- Petites mémoires internes très rapides d'accès
  - donnée, instruction, adresse
  - 8, 16, 32 bits
  - de 10 à 100
- 2 types de registre :
  - Usage général : stocker les résultats intermédiaires (données ou adresses)
  - Registres particuliers :
    - RI (registre instruction): instruction en cours d'exécution
    - CO (compteur ordinal): adresse de la prochaine instruction à charger dans l'unité centrale pour l'exécution
    - ACC (registre accumulateur): résultat des opérations arithmétiques et logiques



Exécution d'une instruction

1. Charger la prochaine instruction dans RI
2. Modifier CO pour qu'il pointe sur l'instruction suivante
3. Décoder l'instruction dans RI
4. Localiser en mémoire d'éventuelles données
5. Charger les données dans les registres appropriés
6. Exécuter l'instruction dans RI
7. Revenir à 1 pour exécuter l'instruction suivante.

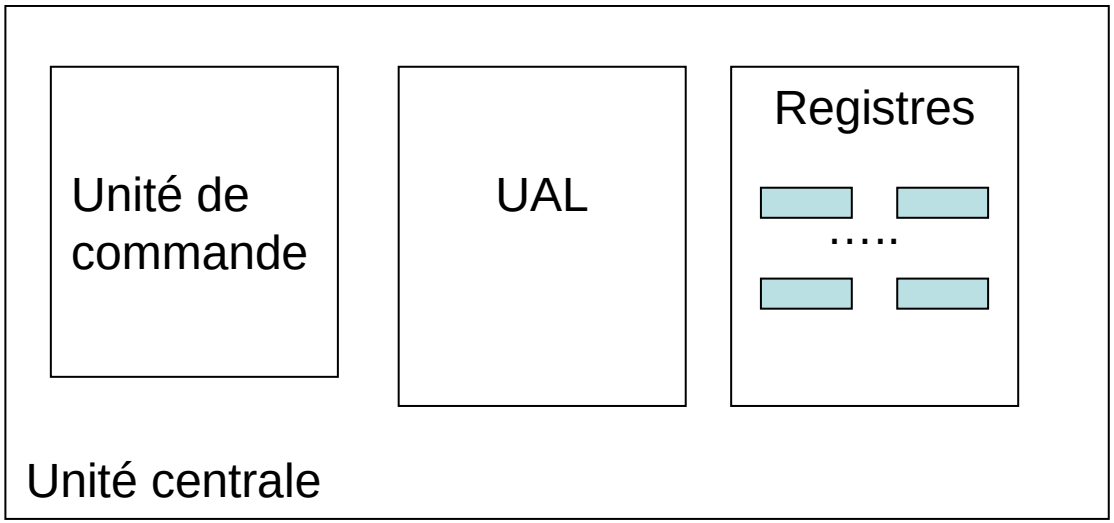
=> cycle de chargement – décodage - exécution

Exécution d'une instruction

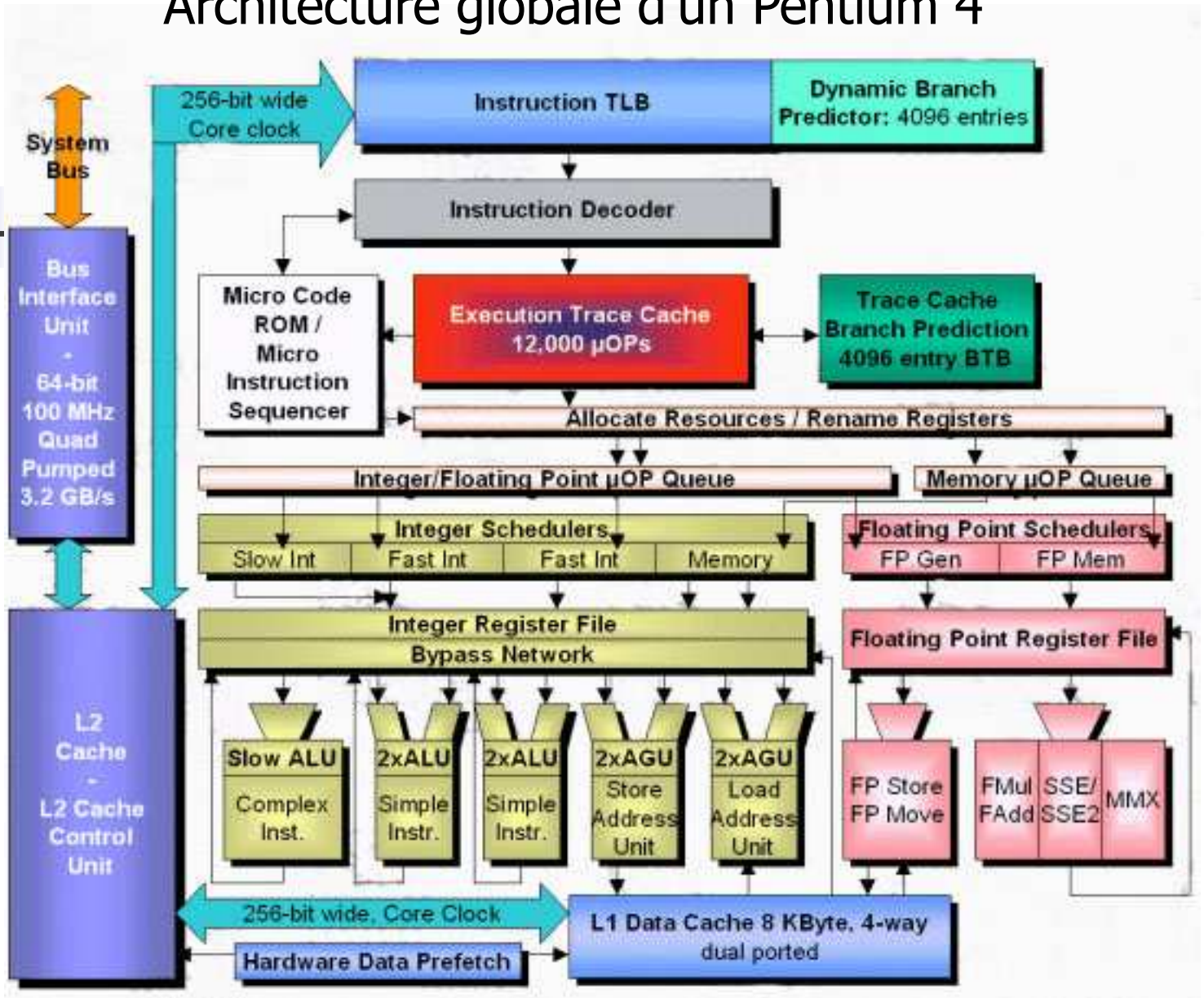
« *Incrémenter le contenu de la case mémoire d'adresse XYZ* » :

- le processeur lit et décode l'instruction dans RI
- le processeur modifie CO
- le processeur demande à la mémoire le contenu de l'emplacement XYZ
- la valeur lue est rangée dans l'accumulateur ACC
- l'unité arithmétique (UAL) ajoute 1 au contenu d'ACC
- le contenu de l'accumulateur est écrit en mémoire à l'adresse XYZ.

Simple, non?

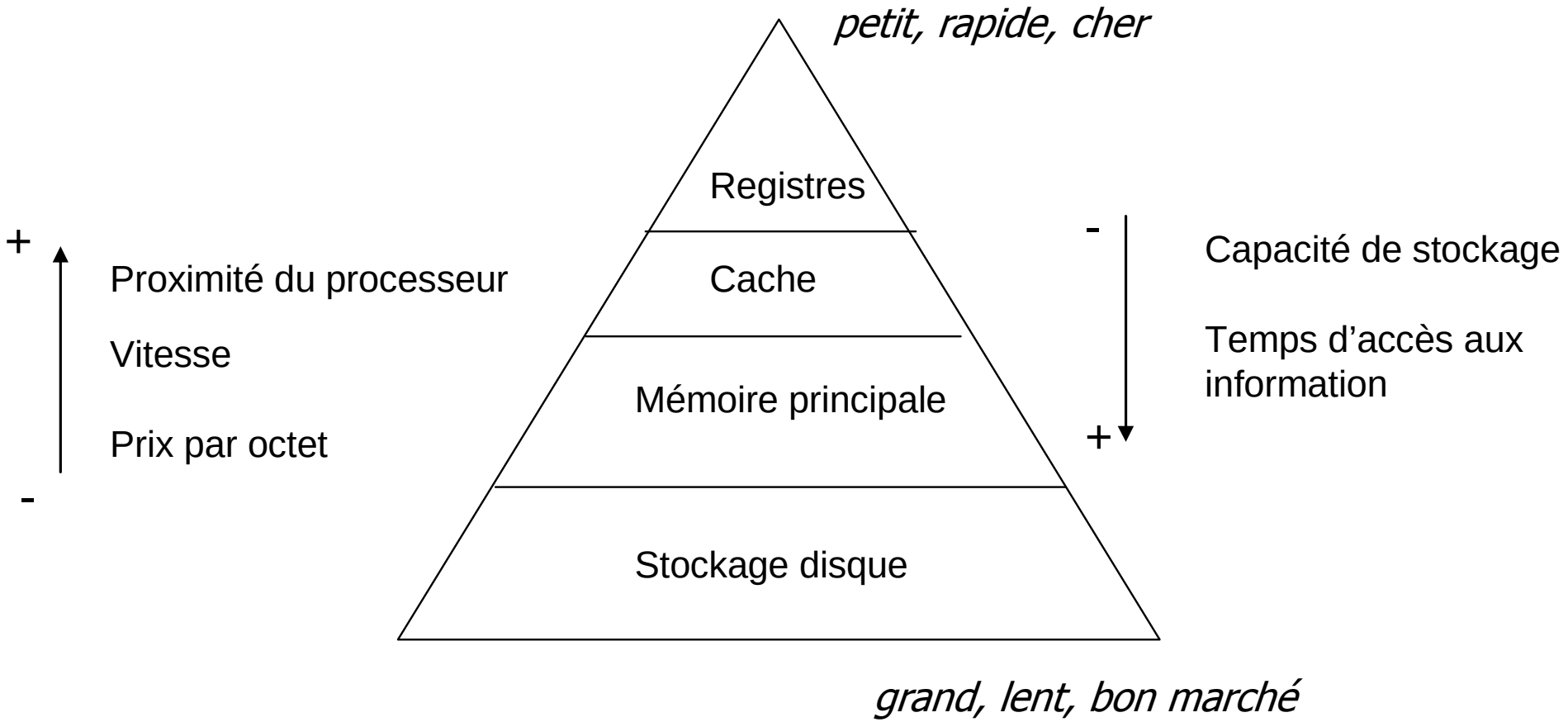


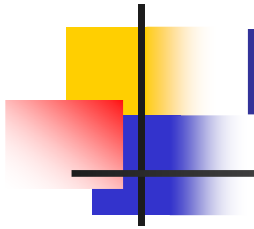
Architecture globale d'un Pentium 4



III. Mémoire

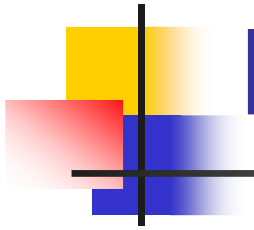
Types de mémoire





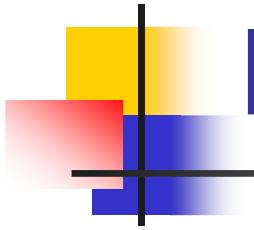
Mémoire : caractéristiques

- **Capacité de stockage** : taille de la mémoire en nombre d'octets (Ko, Mo, Go, To, …)
- **Temps d'accès** : temps qui s'écoule entre une demande de lecture ou écriture et son accomplissement
- **Débit** : nombre d'octets ou bits pouvant être lus ou écrits par seconde
- **Volatilité** : une mémoire volatile perd son contenu lorsqu'on coupe le courant
  - mémoire vive (RAM): volatile
  - mémoire morte (ROM): non volatile (hors disque dur), utilisée au démarrage de l'ordinateur



La mémoire principale

- RAM
  - mémoire vive (lecture + écriture)
  - volatile
  - contient programmes + données (architecture de *Von Neumann*)
- Organisation :
  - La mémoire est formée de plusieurs mots: 8, 16, 32, 64 bits selon l'architecture. Chaque mot a une adresse permettant à un programme de le référencer
  - Exemple : 4Ko RAM de mémoire composée de mots à 32 bits contiennent 1K mots
    => 10 fils d'adresse sont nécessaires ($1K = 1024 = 2^{10}$)

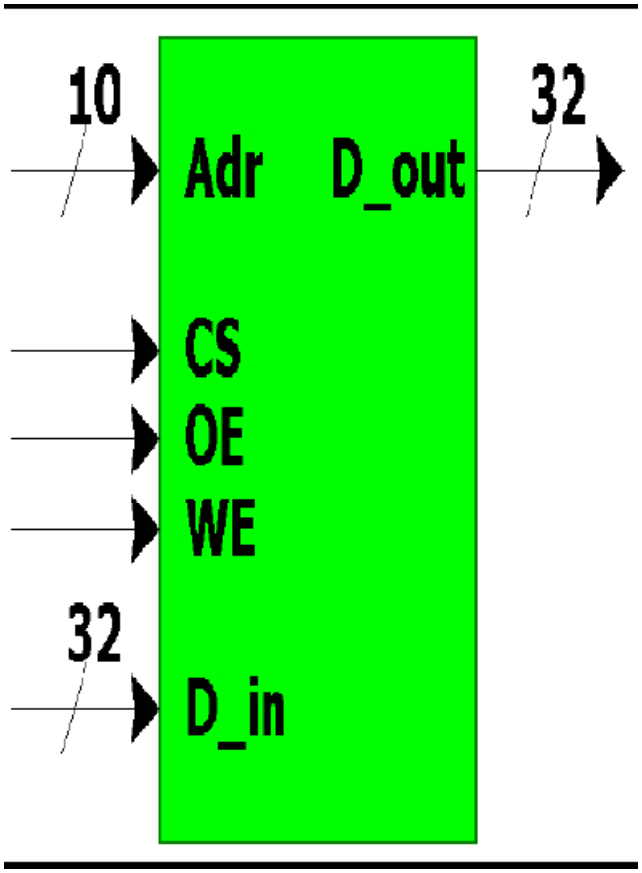


La mémoire principale

Exemple : 16Ko RAM de mémoire composée de mots à 16 bits

Combien de fils d'adresse sont nécessaires?

Le schéma de fonctionnement



Entrées de commande:

- CS (chip select): autorise une écriture ou une lecture.
- OE (output enable): contrôle la sortie des données
- WE (write enable): active l'écriture

Entrée de l'adresse mémoire:

- Adr : bus d'adresse

Entrée/sortie des données:

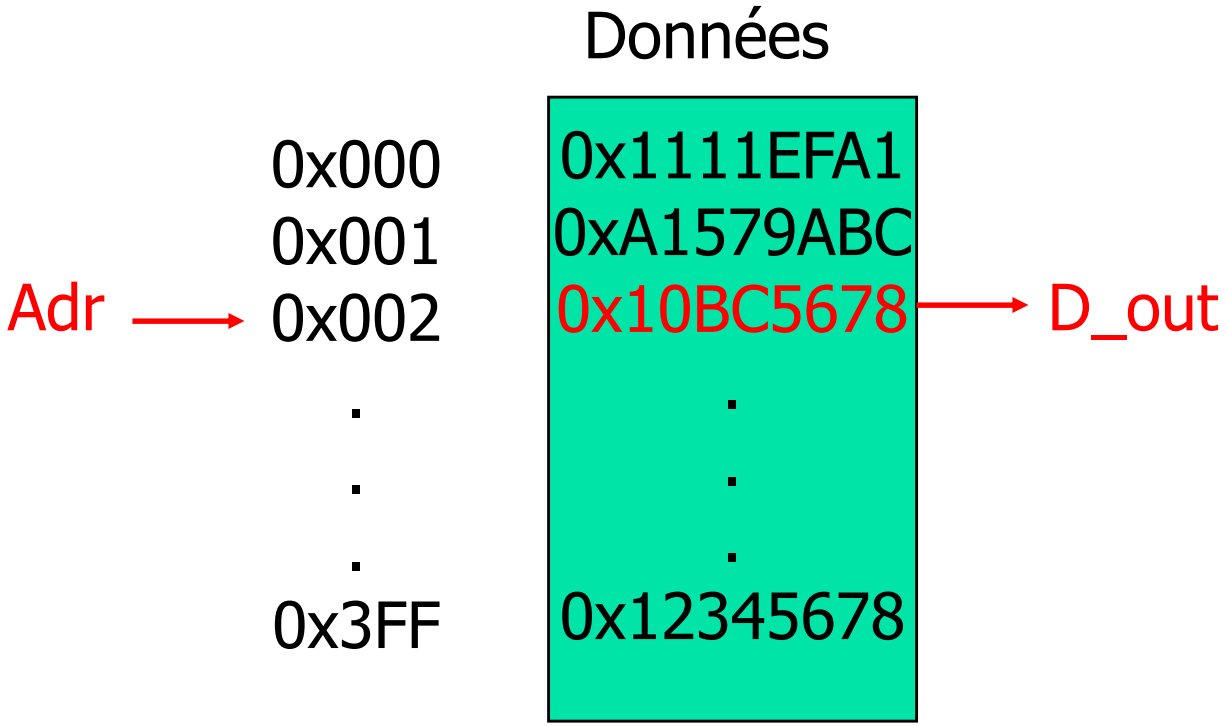
- D_in : bus de données, écriture
- D_out : bus de données, lecture

Lecture d'un mot mémoire

CS : **activé**

WE : désactivé

OE : **activé**

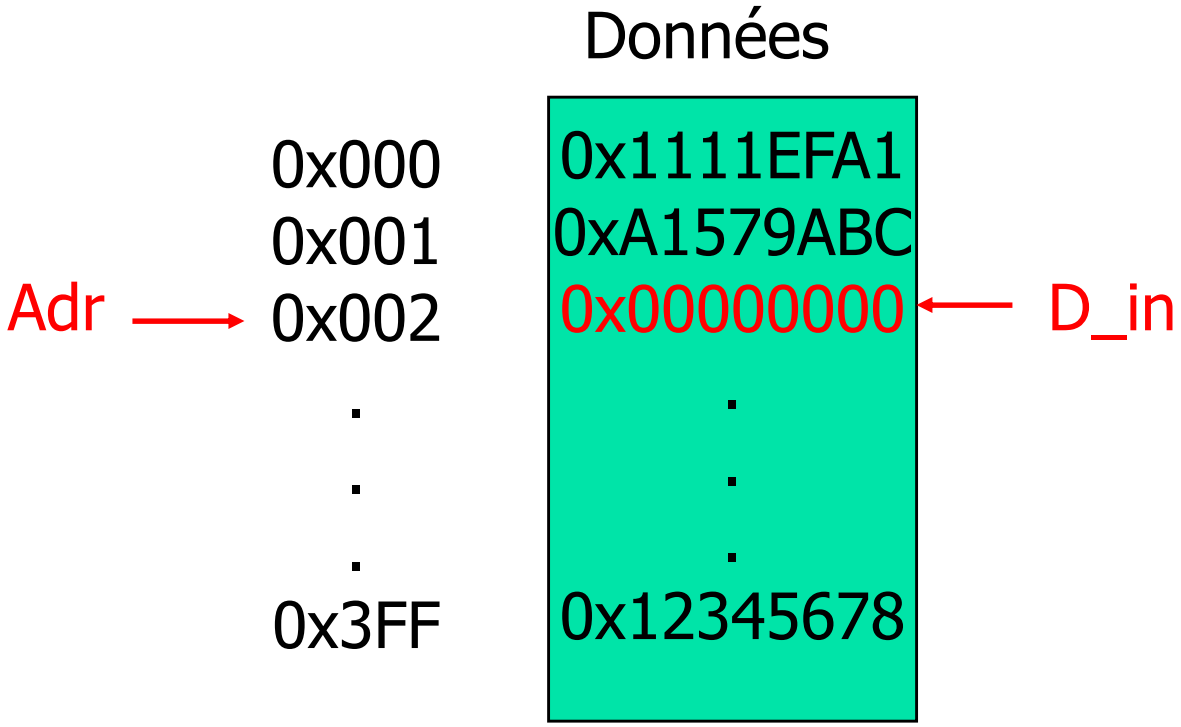


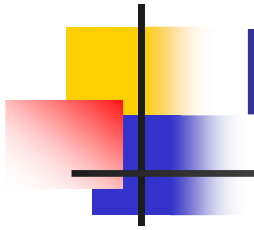
Ecriture d'un mot mémoire

CS : **activé**

WE : **activé**

OE : désactivé

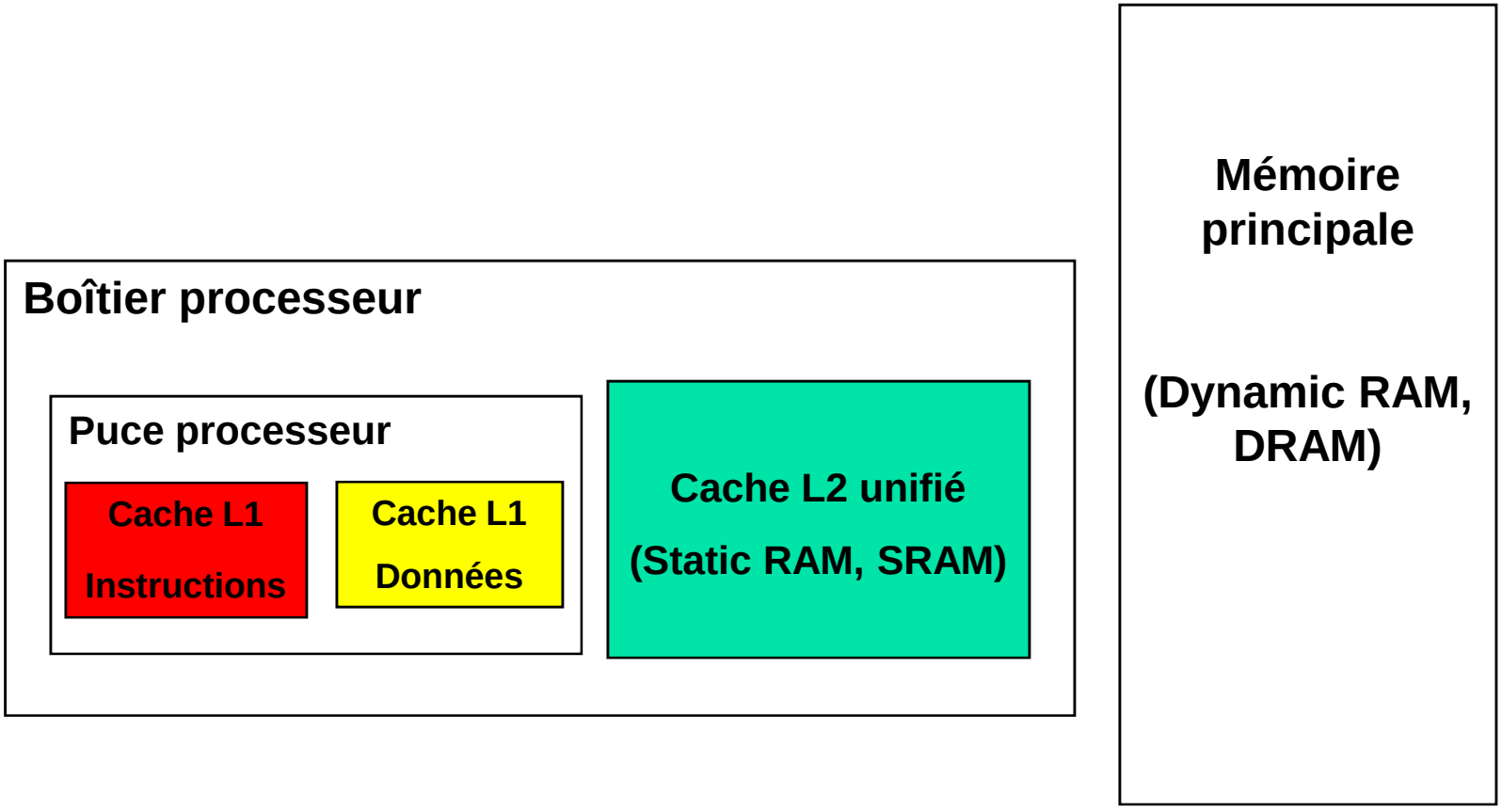


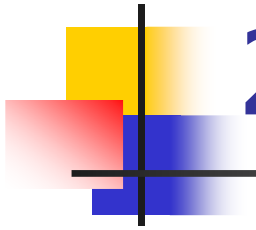


La mémoire cache

- But :
  - éviter de rechercher en mémoire principale des données déjà cherchées précédemment en les conservant près du processeur dans une petite mémoire à accès rapide.
- Principe de localité :
  - Localité temporelle : tendance à réutiliser des données récemment accédées (instructions d'une boucle);
  - Localité spatiale : tendance à référencer des données voisines d'autres données récemment accédées (tableau a[i], a[i+1],..).

2 niveaux de cache





2 niveaux de cache

- Cache interne (niveau 1 ou L1) :
  - dans le processeur
  - unifié (contient instructions et données)
  - ou séparé (1 cache de données et 1 cache d'instructions)
  - 16, 32, 64,128 Ko
- Cache externe (niveau 2 ou L2) :
  - à côté du processeur
  - unifié (contient instructions et données)
  - de 512 Ko à 1 Mo.

Fonctionnement : contrôleur de cache

1. **Le processeur demande une information** en plaçant l'adresse de cette information sur le bus d'adresse.
2. **Le contrôleur de cache cherche** si l'information est dans le cache en fonction de son adresse.
   - Si oui : succès cache (cache hit) => l'utiliser
   - Si non : défaut de cache (cache miss), on continue 3 et 4.
3. **Le contrôleur de cache place l'adresse sur le bus d'adresses** et commande ainsi une lecture en mémoire principale.
4. **La mémoire principale place l'information** sur le bus de données. Le processeur récupère cette donnée et **le contrôleur de cache la copie** au passage dans le cache.

=> au prochain accès à cette même instruction, le contrôleur de cache enverra directement la donnée au processeur sans démarrer un cycle de lecture en mémoire principale.

FIN