2008

UMH

Xavier Dubuc

Voici un résumé de toute la matière à étudier pour l'examen de janvier 2008 du cours de Fonctionnement des Ordinateur donné par Monsieur Luc Onana Alima.

[FONCTIONNEMENT DES ORDINATEURS : EXAMEN 08]

Sommaire

Fonctionnement des Ordinateurs – Examen 08.

1. Introduction aux ordinateurs.

Définition : Architecture des ordinateurs.

Aspects logiques, attributs de l'ordinateur visibles au programmeur en langage machine. Les aspects architecturaux ont donc un impact direct sur l'exécution des programmes. (Ex : adressage de la mémoire, représentation de chaque type de données,…)

Définition : Organisation des ordinateurs.

Aspects physiques, unités opérationnelles et leurs interconnexions qui réalisent l'architecture c à d comment sont implémentées les caractéristiques. (Ex : technologie utilisée pour la mémoire, le nombre de registres utilisés,…)

Une architecture correspond à plusieurs organisations, en effet l'architecture est une spécification qui peut durer quelques temps alors que l'organisation est une implémentation qui varie selon les technologies.

Fonctions de base d'un ordinateur.

Elles sont au nombre de 4 :

- Traitement des données
- Déplacement des données
- Mémorisation des données
    - Lecture
      Les données sont transférées de l'environnement vers la mémoire.
    - Ecriture
      Les données sont transférées de la mémoire vers l'environnement.
    - Mémorisation temporaire
      Les données en cours de traitement.
    - Mémorisation à long terme
      Sauvegarde des données pour utilisation ultérieure.
- Contrôle des 3 autres fonctions

Structure interne d'un ordinateur.

1. **Processeur – CPU (Central Processing Unit)**
    - Le cœur de l'ordinateur, il contrôle le fonctionnement de l'ordinateur et exécute les (instructions) fonctions de traitement des données.
        - Unité de contrôle qui contrôle le fonctionnement du CPU et donc de l'ordinateur.
        - Unité Arithmétique et Logique (ALU) qui effectue les fonctions de traitements des données
        - Les registres qui sont la mémoire interne du CPU

- L'interconnexion du processeur qui gère la communication entre les 3 composants ci-haut.

2. **Mémoire principale**
   - Dispositif de mémorisation des données en cours de traitement.
3. **Entrées/Sorties (E/S ou I/O)**
   - Transfert des données entre l'ordinateur et son environnement.
4. **Interconnexion des systèmes**
   - Gère la communication entre le CPU, la mémoire et les E/S.

Evolution et performances des ordinateurs.

- Caractéristiques de l'évolution des ordinateurs :
  - ✓ Augmentation de la vitesse des CPU
  - ✓ Diminution de la taille des ordinateurs
  - ✓ Augmentation de la taille des mémoires
  - ✓ Augmentation de la capacité et vitesse des E/S
- Facteurs responsables de l'augmentation de la vitesse des processeurs :
  - ✓ Diminution de la taille des microprocesseurs ➔ *La réduction de la distance entre les composants conduit à une augmentation de la vitesse.*
  - ✓ Organisation du CPU ➔ Utilisation de techniques comme *« Le pipelining »*, *« l'exécution parallèle »* et *« l'exécution spéculative »* afin de maintenir le CPU actif presque tout le temps.

Dans la conception d'un ordinateur, il est très difficile de trouver un équilibre entre les performances des différents composants. En effet, la vitesse du CPU ayant été considérablement augmentée, le temps d'accès à la mémoire ne suit pas. On utilise alors des techniques telles que *la mémoire cache*, *large chemin de données entre CPU et la mémoire* et *des puces mémoires plus sophistiquées* pour compenser le déséquilibre.

2. Première génération d'ordinateurs.

Ils étaient construits sur la technologie des tubes à vide, ils étaient très volumineux et consommaient beaucoup d'énergie.

1. ENIAC

Premier ordinateur électronique numérique généraliste, conçu et construit à l'université de Pennsylvanie par le professeur *John Mauchly* et le doctorant *John Presper Eckert* dans le but de venir en aide aux américains pour la seconde guerre mondiale. Il avait pour objectif de faire des calculs pour le laboratoire de balistique mais, bien que sa construction ait commencé en 1943, il ne fut pas achevé avant 1946 et donc il ne fut pas utilisé pour la guerre. Cependant il fut tout de même utilisé jusqu'en 1955.

- Caractéristiques :
  - Poids : 30 Tonnes.
  - 5000 Additions/seconde.
  - 18000 tubes à vide.
  - 140 kilowatts de consommation électrique
  - Arithmétique en base 10, chaque chiffre étant représenté par un anneau de 10 tubes à vide.
  - Programmé manuellement ➔ Positionnement des commutateurs, brancher et débrancher des câbles **manuellement**. Ceci était un **inconvénient majeur.**

2. IAS (Machine de Von Neumann)

Conçu dans l'optique de pallier à la grande difficulté de programmer l'ENIAC. L'idée essentielle est de mettre le programme et les données dans une mémoire centrale, c'est le concept de programme enregistré (*stored-program concept*). Sa construction a été lancée en 1946 et s'est terminée en 1952, et cette machine sert de prototype pour tous les ordinateurs à usage général.

- Caractéristiques :
  - Une mémoire principale qui contient des instructions machines et des données.
  - Une unité arithmétique et logique (ALU) qui effectue les opérations sur des données binaires (addition, soustraction, multiplication et division)
  - Une unité de contrôle qui interprète les instructions en mémoire et cause leur exécution
  - Un dispositif d'entrées/sorties qui est géré par l'unité de contrôle et qui échange des données avec les périphériques.

Définition : Un mot mémoire est une unité d'informations accessible en une seule opération de lecture.

Pour l'IAS, la mémoire consistait en 1000 mots de 40 bits chacun.

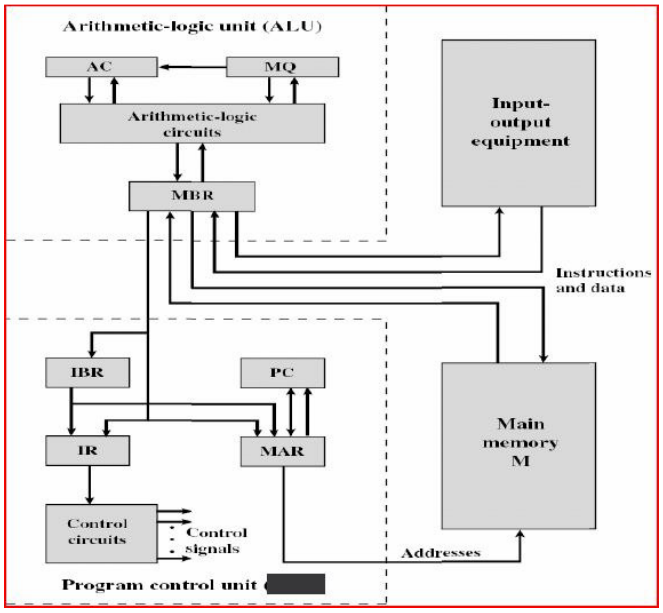
2 types de mots :

Mots de type « donnée » (nombre)

Mots de type « instruction » : 2 instructions de 20 Bits avec un *code op* qui indique l'opération à effectuer suivi d'une adresse indiquant un des mots mémoire.

Fonctionnement de l'IAS.

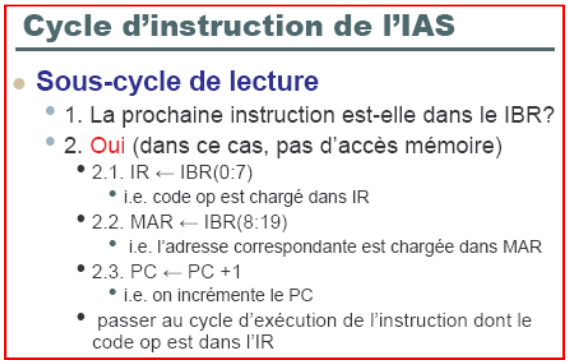
L'unité de contrôle fait fonctionner l'IAS en causant l'extraction des instructions de la mémoire et l'exécution de ces instructions une à la fois.

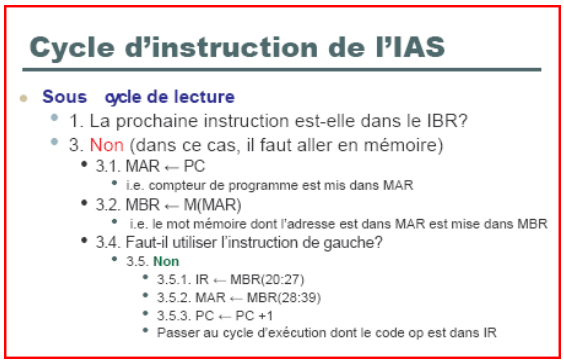


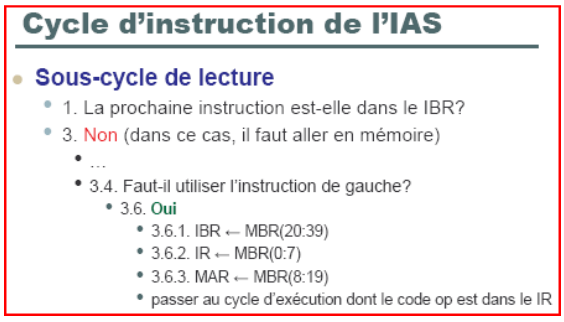
On peut découvrir l'existence d'unité de mémorisation à l'intérieur du CPU, ce sont les **registres**. Il y a les registres de l'ALU et les registres de l'unité de contrôle.

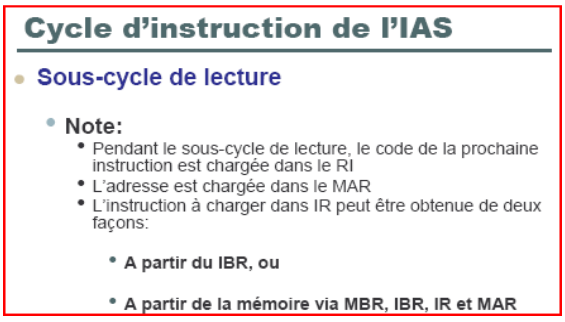
- **AC** *(Accumulator)* et **MQ** *(Multiplier and Quotient)* : Ils sont utilisés pour stocker temporairement des opérandes et des résultats (bits les plus significatifs) des opérations de l'ALU.
- **MBR** *(Memory Buffer Register)* est utilisé pour stocker un mot mémoire devant être transféré en mémoire ou pour recevoir un mot mémoire venant de la mémoire.
- **MAR** *(Memory Address Register)* spécifie l'adresse du mot mémoire dans lequel on va mettre le contenu du **MBR** *(écriture)* ou à partir duquel on va extraire le contenu à mettre dans le **MBR** *(lecture)*.
- **IR** *(Instruction Register) est utilisé pour stocker le code en 8bits de l'instruction en cours d'exécution.*
- **IBR** *(Instruction Buffer Register)* est utilisé pour stocker temporairement l'instruction de droite d'un mot mémoire.
- **PC** *(Program Counter)* est utilisé pour stocker l'adresse de la prochaine paire d'instructions à extraire de la mémoire.

Le fonctionnement se fait par cycle d'instruction répété, chaque cycle consistant en 2 sous-cycles : la **lecture** et l'**écriture**.









- **Sous-cycle d'exécution.**
  Le code op, maintenant dans l'**IR**, est interprété et exécuté par les circuits de contrôle. L'exécution de l'instruction par les circuits de contrôle génère des signaux de contrôle particuliers qui provoquent une certaine action : **Mouvement des données** ou **Traitement des données** *(exécution d'une opération)* **par l'ALU.**

Il y avait 21 instructions de l'**IAS** regroupées en 5 catégories :

- ♠ Transfert de données : Déplacement des données entre la mémoire et les registres de l'**ALU** ou entre deux registres de l'**ALU**.
- ♠ Branchement inconditionnel : Modifier le contenu du **PC** pour faire des branchements (exemple : répétitions)
- ♠ Branchement conditionnel : Modifier le contenu du **PC** sur base d'une condition ➔ Points de décisions.
- ♠ Arithmétiques : Opérations sur des données, exécutées par l'**ALU**.
- ♠ Modification d'adresse : Permet de calculer des adresses dans l'**ALU**, puis de les insérer dans des instructions rangées en mémoire. Similaire aux tableaux indexés.

3. Deuxième génération d'ordinateurs.

Définition : Un transistor est un composant solide en silicium inventé en 1947 dans les laboratoires Bell. Il est plus petit par rapport aux tubes à vide et il réduit la consommation d'énergie.

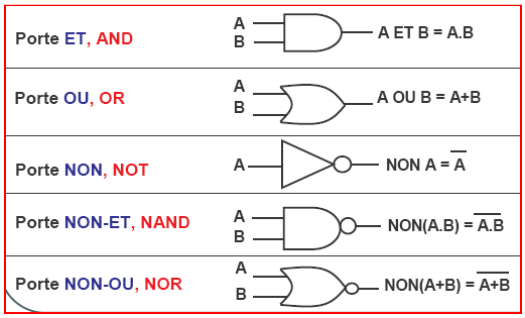
Cette génération est basée sur les transistors. Les unités arithmétiques et logiques sont plus complexes, on introduit les langages de haut niveau et les logiciels système.

4. Troisième génération d'ordinateurs.

Basée sur la micro-électronique ➔ Les circuits intégrés.

Les éléments fondamentaux de l'ordinateur sont les portes et les cellules de mémoires.

Définition : Une **porte logique** est un composant capable de mettre en œuvre une fonction booléenne ou logique simple. (On les appelle portes car elles fonctionnent de la même manière d'une écluse.) C'est un circuit électronique qui produit un signal de sortie sous forme d'une opération booléenne sur ses signaux d'entrée. Chaque porte possède une ou plusieurs (souvent 2) entrées et une sortie. Un changement de données à l'entrée entraîne presque instantanément la sortie correcte.



Un jeu de portes est dit fonctionnellement complet s'il permet d'implémenter n'importe quelle fonction booléenne. De telle façon, on simplifie la conception et la fabrication des circuits car on se limite à un ou 2 types de portes. Les circuits numériques sont souvent implémentés uniquement avec des portes NON-ET ou uniquement avec des portes NON-OU.

Définition : Une **cellule de mémoire** est un composant capable de mémoriser un bit de donnée.

Lien entre éléments de bases et les 4 fonctions d'un ordinateur → voir transparents.

Conception pour la performance.

Du point de vue de l'organisation et de l'architecture des ordinateurs, 2 aspects sont frappants : les blocs de base des ordinateurs d'aujourd'hui sont les mêmes que ceux de l'**IAS** et les techniques d'amélioration des performances du matériel sont de plus en plus sophistiquées.

Vitesse des microprocesseurs.

Les puces de mémoire ont quadruplé tous les trois ans la capacité des mémoires dynamiques à accès aléatoire (**DRAM**, *Dynamic Random Access Memory*), qui forme la technologie de base des mémoires principales d'ordinateurs.

Les microprocesseurs contemporains sont caractérisés par une grande vitesse, qui découle de la réduction des distances entre composants fondamentaux. Il faut donc des techniques appropriées pour maintenir le microprocesseur constamment occupé pour ainsi exploiter toute la puissance brute des processeurs. Voici quelques-unes de ces techniques :

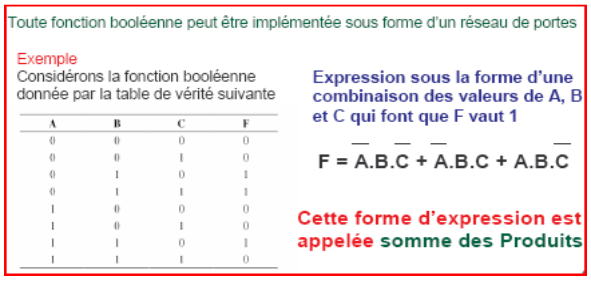
- Prédiction des branchements
  Le processeur essaie de trouver, sur base de l'instruction courante, les branchements ou les groupes d'instructions à exécuter dans la suite.
- Analyse des flots de données
  Le processeur essaie de créer un ordonnancement optimisé des instructions, en analysant les instructions qui sont dépendantes des résultats d'autres instructions. Ceci évite des attentes inutiles.
- Exécution spéculative
  Exécution anticipée des instructions avant qu'elles n'apparaissent réellement dans l'exécution du programme. Les résultats de telles exécutions sont mis dans des emplacements temporaires.
- Pipelining
  Subdiviser le cycle d'instruction en tâches et exécuter ces tâches en parallèle. (Ex : lire la prochaine instruction en parallèle avec l'exécution de celle en cours.)
- Superscalaire
  Formation d'un flot d'exécution dynamique, grâce à l'utilisation de la prédiction de branchement. Le processeur analyse ce flot afin de déterminer des dépendances des données et élimination des dépendances artificielles. ➔ Exécution des instructions suivant les vraies dépendances.

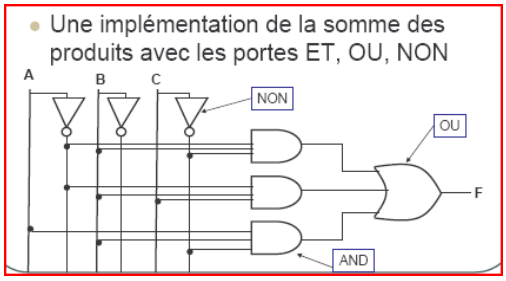
- Les processeurs et la capacité des mémoires a augmenté très rapidement mais la vitesse de transfert entre la mémoire centrale et le processeur augmente très lentement. C'est donc le chemin le plus critique de l'ordinateur.

Les circuits combinatoires.

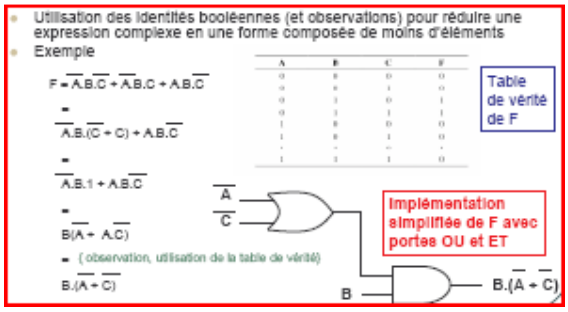
Définition : Ensemble de portes interconnectées dont la valeur de sortie à un instant donné ne dépend que des valeurs d'entrée à cet instant. Il se compose de n entrées binaires et de m sorties binaires. On peut les définir de 3 manières, en donnant la table de vérité, les symboles graphiques ou une équation booléenne.

Exemple :



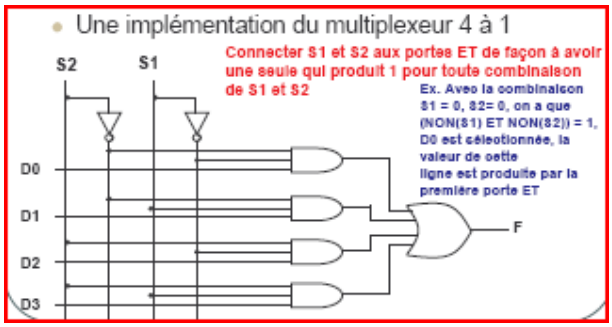


D'où l'intérêt de simplifier les expressions booléennes afin d'utiliser peu de portes pour l'implémentation.



Multiplexeurs.

Ce sont des circuits combinatoires connectant plusieurs entrées à une seule sortie : à chaque moment, une seule entrée est sélectionnée pour être passée à la sortie. Il permet de transmettre sur une même ligne (sortie) des informations provenant de plusieurs sources (entrées).



Décodeurs.

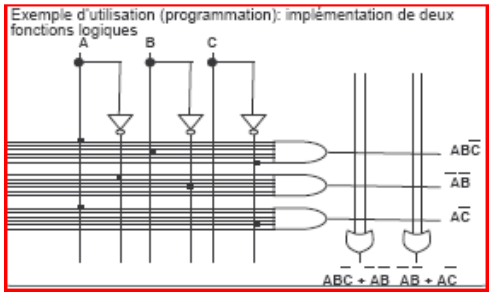
Circuit combinatoire ayant n entrées et 2^n sorties, en fonction de la chaîne de bits en entrée, une seule des 2^n lignes de sorties est sélectionnée.

Décodeur avec 3 entrées et 8 sorties

Entrée			Sortie							
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Tableaux logiques programmables.

L'objectif est de développer une puce à usage général adaptable à des objectifs spécifiques, le principe est fondé sur le fait que toute fonction booléenne peut être exprimée sous la forme d'une somme de produits.



Ils sont souvent appelés des circuits sans mémoires parce que la sortie d'un circuit combinatoire dépend uniquement de l'entrée en cours et donc il n'y a pas de mémorisation de l'histoire des entrées précédentes.

5. Conversion binaire-décimal-hexadécimal.

- Binaire ➔ Décimal.

$\ldots b_2 b_1 b_0, b_{-1} b_{-2} b_{-3} \ldots$ un nombre binaire. Alors la valeur de ce nombre en base décimale est : $\left(\sum_i b_i . 2^i\right)_{10}$ autrement dit, il faut multiplier chaque 0 ou 1 par 2^{position}.

- Décimal ➔ Binaire.

Succession de divisions du nombre décimal par 2 et on note le reste à l'envers. (Conversion de la partie entière)

Exemple : **11**, 11/2 = 5 (+1) ; 5/2 = 2 (+1) ; 2/2 = 1 (+0) ; 1/2 = 0 (+1) ➔ $(11)_{10} = (1011)_2$.

Conversion de la partie fractionnaire. Succession de multiplication du nombre décimal par 2 et on note le nombre entier obtenu et on garde la partie fractionnaire.

Exemple : **0,35** ; 0,35.2 = **0**,7 ; 0,7.2=**1**,4 ; 0,4.2 = **0**,8 ; 0,8.2 = **1**,6 ; 0,6.2 = **1**,2 ; 0,2.2 = **0**,4 on cycle ! ➔$(0{,}35)_{10} = (0{,}010110)_2$

- Hexadécimal ➔ Décimal.

1=1 ; 2=2 … 9=9 ; A=10 ; B=11 ; C=12 ; D=13 ; E=14 ; F=15.

⇨ $(8EA)_{16} = (8.16^2 + E.16^1 A.16^0)_{10} = (8.256 + 14.16 + 10.1)_{10}$
⇨ $= (2048 + 224 + 10)_{10} = (2282)_{10}$

- Binaire ➔ Hexadécimal.

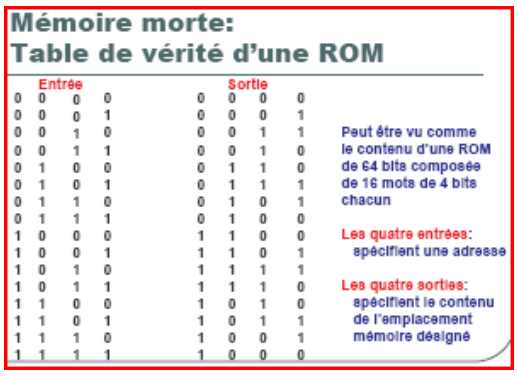
On organise le nombre binaire à convertir en groupes de 4 bits et on remplace chaque groupe par sa valeur en Hexadécimal.

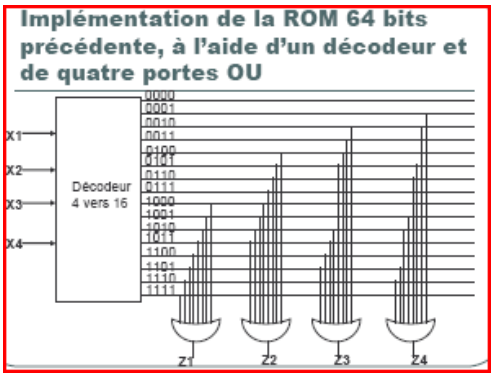
0000 = 0	1000 = 8
0001 = 1	1001 = 9
0010 = 2	1010 = A
0011 = 3	1011 = B
0100 = 4	1100 = C
0101 = 5	1101 = D
0110 = 6	1110 = E
0111 = 7	1111 = F

6. La mémoire à lecture seulement.

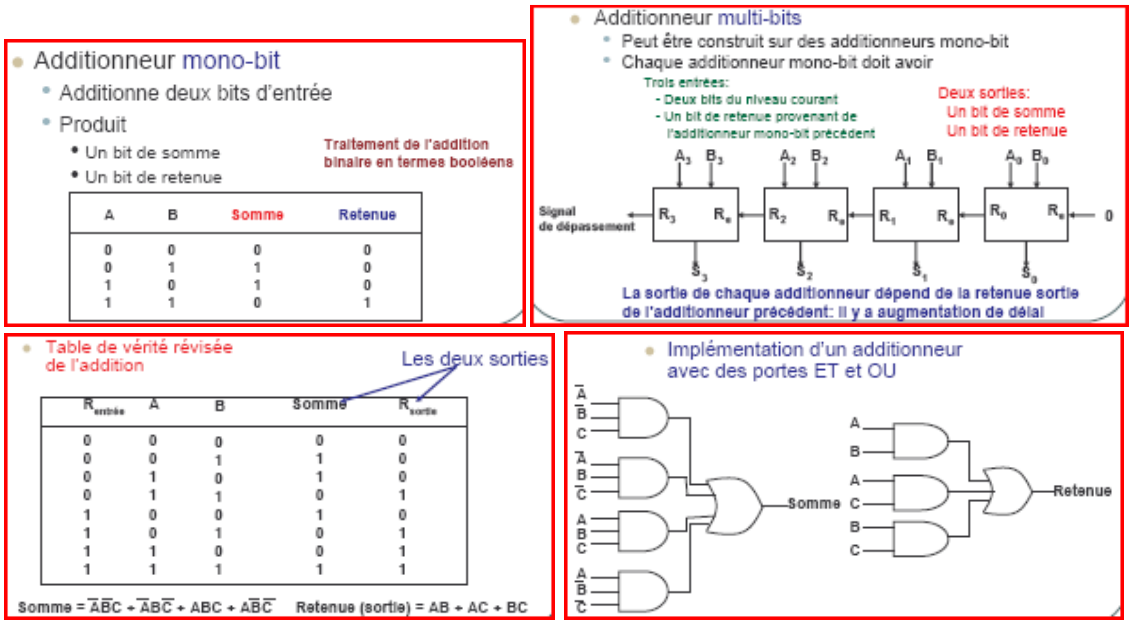
Définition

La *Read-Only Memory*, la ROM, est une mémoire implémentée par des circuits combinatoires. C'est une mémoire qui ne supporte que l'opération de lecture, les informations binaires dans la ROM sont permanentes et son stockées pendant la fabrication du circuit donc une entrée produit toujours la même sortie (donnée), elle peut être implémentée avec un décodeur et des portes OU.





Le problème des additionneurs.



La construction basée sur des additionneurs mono-bits est limitée par l'accumulation des délais de propagation. La solution est de construire des additionneurs de plus de 1 bit, par exemple construire sur des additionneurs de 8 bits.

7. Circuits séquentiels.

Les circuits combinatoires (à l'exception de la **ROM**) n'offrent pas de mémoire ou d'informations d'état, on utilise les circuits séquentiels.

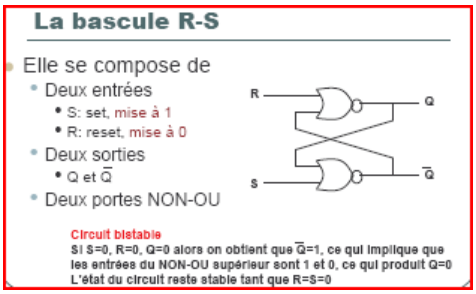
- Définition

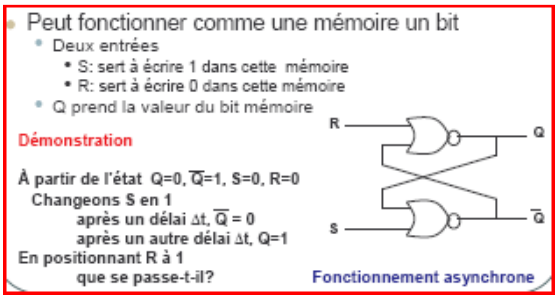
  C'est une forme plus complexe de circuit logique numérique utilisée pour servir de mémoire. La sortie d'un tel circuit dépend de l'entrée en cours et de son état courant. Exemple : Les bascules, les registres, les compteurs.

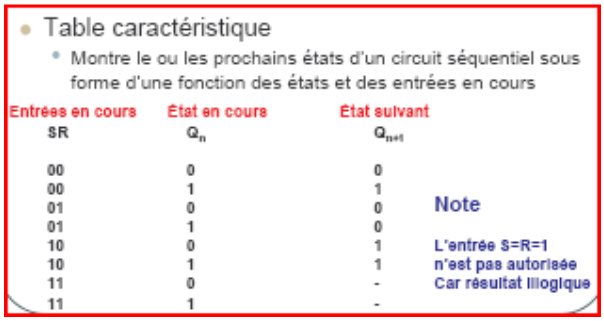
- Les bascules.

Une bascule a deux états. Elle se trouve à tout moment dans l'un ou l'autre des 2 états. En l'absence d'entrée, elle reste dans son état courant. Une bascule a 2 sorties, qui sont toujours le complément l'une de l'autre. Ces sorties sont généralement libellées Q et $\overline{Q}$. Une bascule peut servir comme mémoire à 1 bit.

- La bascule R-S

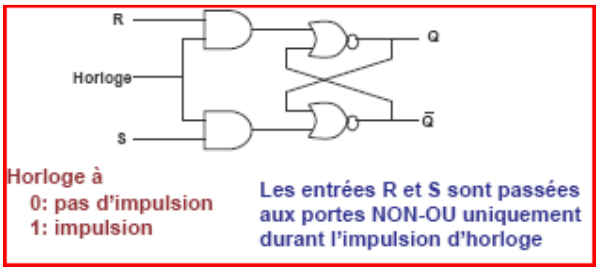






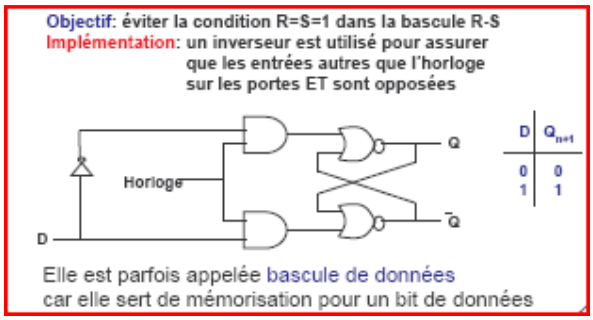
La bascule R-S a un fonctionnement asynchrone : sa sortie change, après un court délais, en réponse à un changement de l'entrée.

- La bascule R-S avec horloge

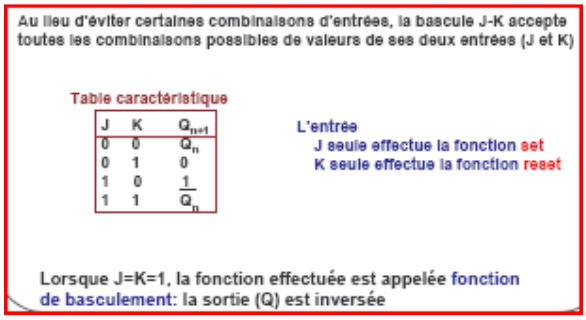


Dans un ordinateur, les évènements sont en général synchronisés par une impulsion horloge. Les changements se produisent uniquement au moment de l'impulsion d'horloge. La **bascule R-S avec horloge** permet de mettre en œuvre ce fonctionnement où les changements sont contraints par l'impulsion horloge.

- La bascule D



- La bascule J-K



- Les registres.

Définition : C'est un des éléments les plus essentiels du **CPU**, c'est un circuit digital utilisé dans le **CPU** pour stocker un ou plusieurs bits de données. Ils illustrent l'utilisation des bascules.

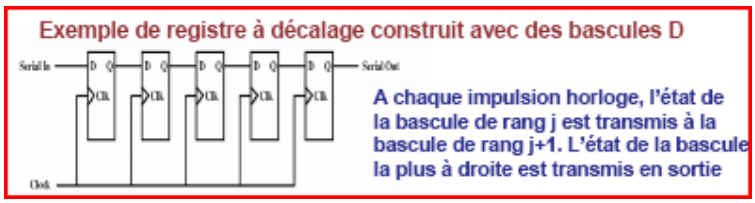
Il y a 2 types de registres :

- ✓ Les **registres parallèles.**

  Ils sont composés d'une série de mémoires 1bit (bascules) que l'on peut lire ou écrire simultanément. Ils servent à ranger (stocker) des données.

- ✓ Les **registres à décalage**.

  C'est un registre qui accepte et/ou transfère des informations en série. Ils peuvent être utilisés pour interfacer des dispositifs d'entrées/sorties sériels et dans l'**ALU** pour des opérations de décalage et de rotation logique.



- Les compteurs.

  Définition : C'est un registre (série de bascules) dont la valeur (contenu) est facilement incrémentée de 1 modulo sa capacité. Un registre compteur de *n* bits pour ainsi compter jusqu'à $2^n - 1$

  Exemple : le **PC** dans le **CPU**.

  - Compteur asynchrone (ou en cascade).

    Les bascules qui le composent changent d'état à des moments différents (propagation du changement d'une extrémité à l'autre)
  - Compteur synchrone.

    Toutes les bascules changent d'état au même moment. Il est plus rapide que le compteur asynchrone, il est utilisé dans les **CPU**.

8. Programmation matérielle et programmation logicielle.

Des composants logiques de base peuvent être interconnectés pour stocker des données binaires et pour effectuer des opérations arithmétiques et logiques sur ces données.

- **Programmation matérielle.**

  Pour un problème donné, on doit :

  - ✓ trouver une configuration des composants logiques de base spécifiques,
  - ✓ réaliser l'interconnexion de ces composants,

  Le processus de réalisation de l'interconnexion est une forme de programmation. Le programme qui en résulte est connu sous le nom de **programme câblé** *(hardwired program)*.

  ➔ Une modification du programme implique un re-câblage → Approche fort limitée.

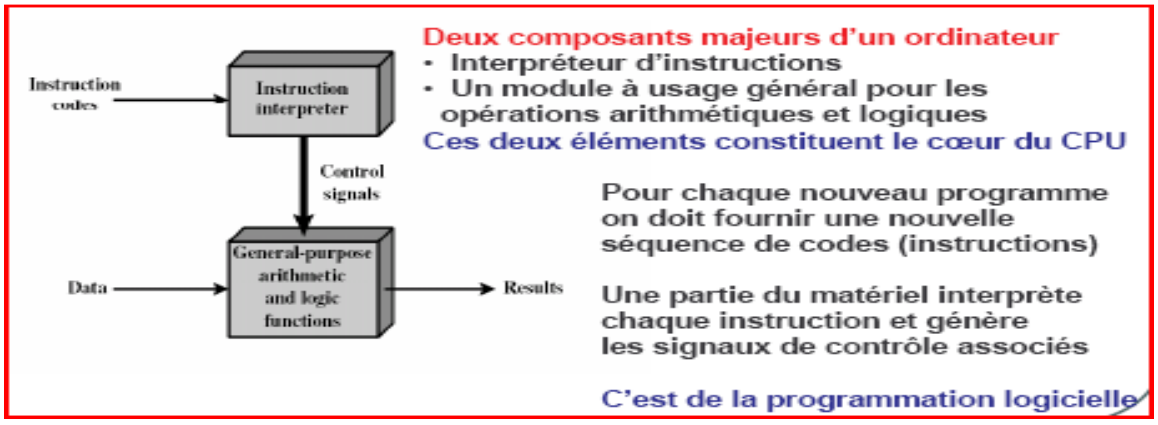
- **Programmation logicielle.**

  Se base sur l'idée d'une configuration matérielle à usage général, pouvant effectuer des opérations arithmétiques et logiques. Cette configuration opère sur base des signaux de contrôle qui lui sont envoyés. Le programmeur n'a donc plus à re-câbler pour chaque nouveau programme, il a juste besoin de fournir de nouveaux signaux de contrôle.

Comment fournir ces signaux à ce dispositif matériel à usage général ?

Rappelons nous, un programme est une séquence de pas (instructions), à chaque pas, une opération est effectuée sur des données. Pour chaque pas, un nouvel ensemble de signaux de contrôle est nécessaire.

L'astuce réside en un code pour chaque ensemble de signaux de contrôle possible et un dispositif capable d'accepter un code et générer des signaux de contrôle correspondants.



Cycle d'instruction/exécution/interruptions……. ➔ *Transparents.*

9. Structures d'interconnexion.

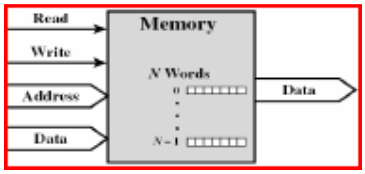
Un ordinateur consiste en un ensemble de composants ou modules de trois types de base :

- Processeur
- Mémoire
- Module d'I/O

qui communiquent entre eux.

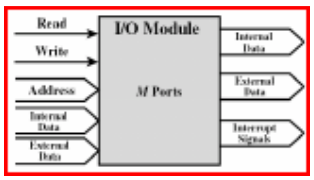
La collection des chemins qui interconnectent ces modules forme ce que l'on appelle structure d'interconnexion. La conception de cette structure dépend des échanges devant s'effectuer entre ces modules.

- Le module mémoire.



En général, un module mémoire consiste en N mots *(words)* de même longueur. Chaque mot possédant comme adresse un numéro pris dans {0,1,…,N-1}. Un mot mémoire de données peut être lu ou écrit dans la mémoire, la nature de l'opération étant indiquée par un signal de contrôle et l'emplacement de la mémoire où lire ou écrire est indiqué par une adresse.

- Le module d'I/O.



Vu de l'intérieur de l'ordinateur, un module d'I/O fonctionne comme un module mémoire *(Il y a 2 opérations : Lecture et Ecriture).* De plus un module d'I/O peut contrôler plus d'une unité périphérique. On peut désigner par un port, chacune des interfaces vers les unités périphériques, dès lors on peut donner à chaque port un numéro unique $\in \{0,1,\dots,M-1\}$ qui est son adresse. De plus, un module d'I/O possède des chemins de données externes pour les entrées/sorties avec l'unité périphérique, il peut également envoyer des signaux d'interruption au processeur.

- Le processeur.
  Il lit (ou reçoit) des instructions et des données, il écrit des données (résultats), après traitement, en sortie. Il envoie des signaux de contrôle vers d'autres modules pour contrôler le fonctionnement global du système et il reçoit des signaux d'interruptions.

Les types de transferts supportés

- Mémoire vers CPU.
  Le **CPU** lit une instruction ou une unité de donnée de la mémoire.
- CPU vers Mémoire.
  Le **CPU** écrit une unité de donnée dans la mémoire.
- Module I/O vers CPU
  Le **CPU** lit des données à partir d'une unité périphérique via un **module d'I/O**.
- CPU vers Module I/O
  Le **CPU** envoie des données à une unité périphérique, via un **module d'I/O**.
- Entre Module I/O et Mémoire.
  Un **module d'I/O** est autorisé à échanger directement des données avec la **mémoire** sans passer par le **CPU** par l'utilisation de la Direct Access Memory *(DMA)*.

- Interconnexion par bus.

Définition : Un **bus** est un chemin de communication connectant deux ou plusieurs composants, il est partagé par les composants qu'il interconnecte. Un signal envoyé par l'un des composants connectés au bus est recevable par tous les autres composants attachés au bus *(broadcast, diffusion).*

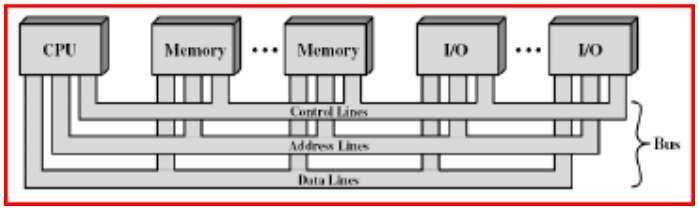
Si plusieurs composants connectés au bus émettent en même temps, les signaux sont altérés
➔ **Il faut qu'un seul composant émette à la fois pour garantir une émission avec succès.**

En général, un bus est constitué de plusieurs lignes de communication, chaque ligne étant capable de transmettre des signaux représentant le binaire 1 ou le binaire 0. En envoyant une série de signaux représentant des binaires, on peut transmettre une séquence de bits l'un après l'autre sur une ligne. Plusieurs bits peuvent transmis simultanément (en parallèle) sur les lignes d'un bus. (**Ex :** Une unité de données de 8 bits peut être transmise sur un bus 8 lignes.)

Il y a différents bus à différent niveaux de la hiérarchie d'un ordinateur. On appellera bus système, un bus qui interconnecte les composants principaux (Processeur, mémoire, module d'I/O) d'un ordinateur. (La grande majorité des structures d'interconnexion utilise un ou plusieurs bus systèmes.) Un bus système, typique, a entre 50 et des centaines de lignes, chaque ligne ayant une signification ou une fonction bien spécifique. En général, ces lignes sont classées en 3 groupes fonctionnels :

- Lignes de données *(bus de données)*
- Lignes d'adresse *(bus d'adresses)*
- Lignes de contrôle *(bus de contrôle)*

On peut aussi avoir des lignes de distribution d'électricité pour alimenter les modules connectés au bus.



- ✓ **Les bus de données** fournissent un chemin permettant de transférer des données entre les modules du système. On appelle **bus de données** l'ensemble des lignes de données, il peut contenir entre 32 et des centaines de lignes.
  La largeur du bus de données est égale au nombre de ses lignes, celle-ci détermine combien de bits peuvent être transmis en parallèle (en une fois). **La largeur du bus de donnée est un élément clé dans la performance globale du système.** (Illustration : Si un bus de données a 8 lignes et que chaque instruction a 16 bits, alors le **CPU** doit faire 2 accès mémoire pour chaque cycle d'instruction.)
- ✓ **Le bus d'adresses** est utilisé pout transmettre l'adresse de la source ou de la destination des données se trouvant sur le bus de données. (**Ex :** pour lire un mot mémoire, le **CPU** place l'adresse du mot qu'il souhaite lire dans le bus d'adresse). **La largeur du bus d'adresses détermine la capacité mémoire maximum, possible, du système.** Les lignes d'adresses sont aussi souvent utilisées pour adresser (désigner) des ports d'I/O.
- ✓ Utilisation typique du bus d'adresse : Les bits d'ordre supérieur servent à identifier un module I/O et les bits d'ordre inférieur servent à sélectionner un **emplacement mémoire** ou un **port I/O** dans le **module I/O** identifié avec les bits d'ordre supérieur.
  Exemple : Avec un bus d'adresse 8 bits, on pourrait utiliser le bus comme suit :
  - Les adresses 00000000…01111111 peuvent référencer des emplacements dans un **module mémoire** (module 0), soit 128 mots mémoires (de l'adresse 0 à 127).
  - Les adresses 10000000…11111111 peuvent servir pour identifier les dispositifs connectés à un **module I/O** (module 1).

- ✓ **Le bus de contrôle** est utilisé pour contrôler l'accès et l'utilisation des bus de données et d'adresse (qui sont partagés par tous les composants qui y sont attachés). Les signaux de contrôle transmettent des **commandes** *(opérations à exécuter)* et des **informations de timing** *(minutage)* entre les modules. Les signaux de timing indiquent la validité des informations sur les bus de données et des adresses.
- ✓ Quelques commandes typiques des bus de contrôle :

  - **Memory write**
    Indique que les données sur le bus doivent être écrites en mémoire, à l'adresse indiquée sur le bus d'adresses.
  - **Memory read**
    Indique que les données se trouvant à l'emplacement dont l'adresse est indiquée sur le bus d'adresses doivent être placées sur le bus de données.
  - **I/O write**
    Indique que les données se trouvant sur le bus de données soient écrites sur le port I/O indiqué sur le bus d'adresses.
  - **I/O read**
    Indique que les données se trouvant sur le port I/O indiqué sur le bus d'adresses doivent être placées sur le bus de données.
  - **Transfer ACK**
    Indique que des données ont été acceptées par (ou placées sur) le bus.
  - **Bus request**
    Indique qu'un module souhaite avoir le contrôle du bus.
  - **Bus grant**
    Indique que le contrôle du bus a été accordé à un module demandeur.
  - **Interup request**
    Indique qu'il y a une interruption en attente.
  - **Interup ACK**
    Indique que l'interruption en attente a été reconnue.
  - **Clock**
    Utilisé pour synchroniser les opérations.
  - **Reset**
    Initialise tous les modules.

- ## Fonctionnement du bus.

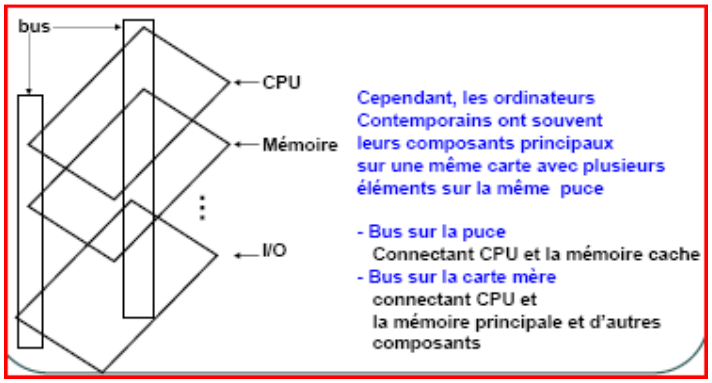
  ♠ Envoi des données.

  Lorsqu'un module M_1 souhaite envoyer des données à un autre module M_2, le module M_1 doit **obtenir le contrôle du bus** et ensuite **transférer les données** via le bus.

♠ Demande des données.

Lorsqu'un module M_1 souhaite demander des données à un autre module M_2, alors M_1 doit : **obtenir le contrôle du bus**, **transférer sa requête** (utilisation des lignes de données et d'adresses appropriées) et ensuite **se mettre en attente** des données du module M_2.

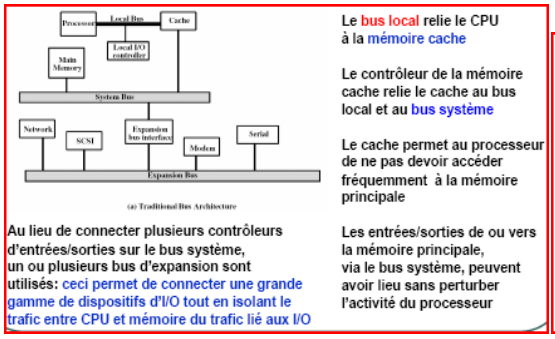
- Réalisation physique typique d'une interconnexion par bus.

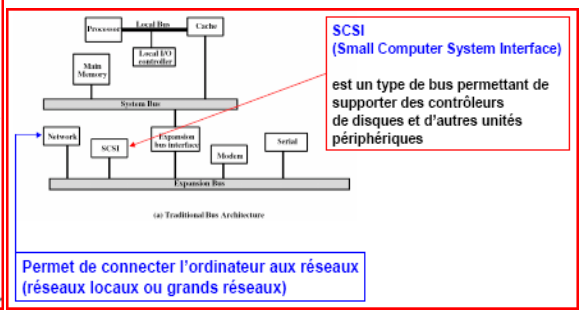


Si plusieurs composants sont interconnectés par un seul bus, la performance du système peut être fortement affectée, car plus il y a de composants sur un bus, **plus celui-ci doit être long** *(ce qui entraîne une augmentation des délais de propagation des signaux)* et **plus il y a de chance** que les demandes cumulées des transferts de données approchent la capacité du bus *(le bus devient alors un goulot d'étranglement, on parle alors de bus bouchonné)*.

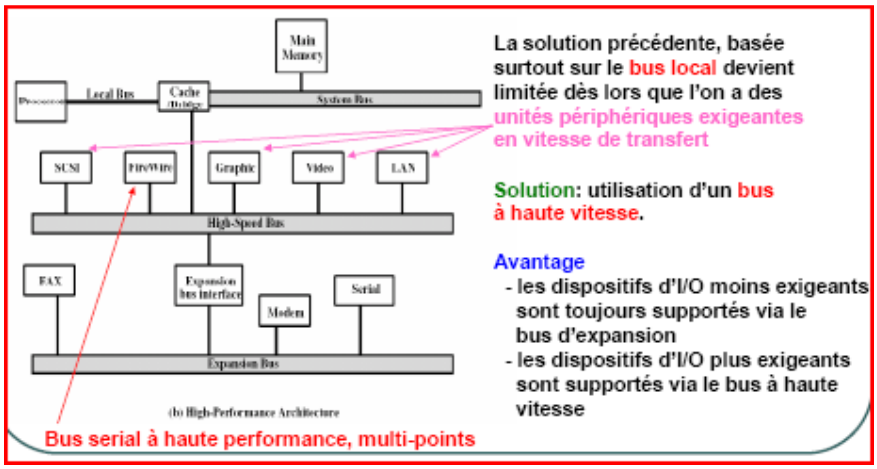
➔ Pour contourner ce problème, la majorité des ordinateurs utilise **plusieurs bus organisés** de *manière hiérarchique.*

Le bus local.





Le bus à haute vitesse.



- **Les éléments importants dans la conception de bus :**
  - **Le type de bus**

Les lignes d'un bus peuvent être classées en 2 types.

1. Les lignes dédiées

   Une ligne dédiée est affectée de façon permanente à une fonction ou à un sous-ensemble de composants de l'ordinateur. (**Ex :** *Lignes dédiées aux adresses et d'autres dédiées aux données*)
2. Les lignes multiplexées

   Utilisation des mêmes lignes pour par exemple transmettre des adresses et des données. Multiplexage temporel :
   - Avantage : Implique l'utilisation de moins de lignes ➔ moins d'espace utilisée, couts réduits.
   - Inconvénient : Circuit plus complexes dans des modules, perte de performance potentielle (absence de parallélisme).

  - **La méthode d'arbitrage**

Dans la plupart des ordinateurs, plusieurs modules peuvent vouloir le contrôle du bus au même moment mais **un seul module** peut transmettre *avec succès* **à la fois**. Il faut donc une méthode d'arbitrage. Il y en a **deux** :

- L'arbitrage centralisé

  Un dispositif matériel, appelé contrôleur de bus (*bus controler*), est chargé d'allouer le droit d'accès au bus. Ce contrôleur peut être une **partie du CPU** ou une **unité séparée**.
- L'arbitrage distribué

  Chaque module possède une logique de contrôle d'accès, ensuite les modules coopèrent afin d'allouer le bus à un seul module à la fois.

  - **Le timing**

Le timing d'un bus détermine la manière dont les évènements sont coordonnés sur le bus. Il existe 2 types de **timing** :

- Le timing synchrone

  Les occurrences des évènements sur le bus sont déterminées par une horloge. Le bus possède alors une ligne horloge via laquelle une horloge transmet une séquence régulière de 1 et de 0 alternés de durée égale et cette ligne peut être lue par tous les autres modules connectés au bus. Une transmission de 1-0 est appelé cycle horloge ou cycle de bus et détermine un intervalle de temps, tous les évènements commençant **au début** d'un cycle horloge.
- Le timing asynchrone

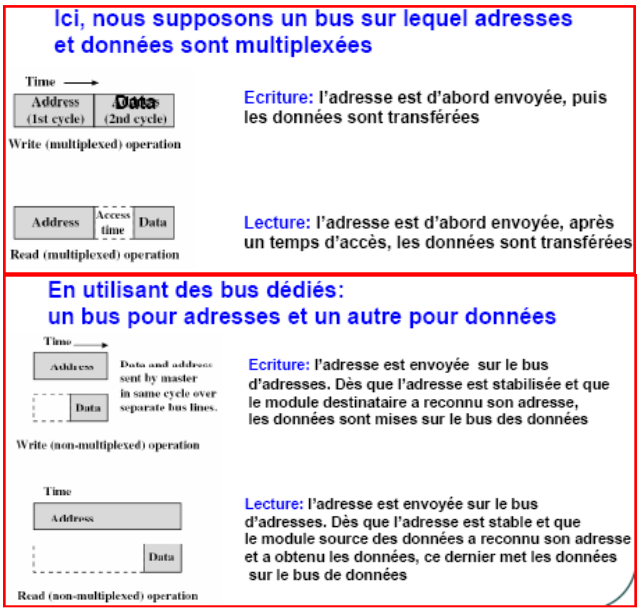
  L'occurrence d'un évènement sur le bus suit et dépend de l'occurrence d'un évènement précédent.

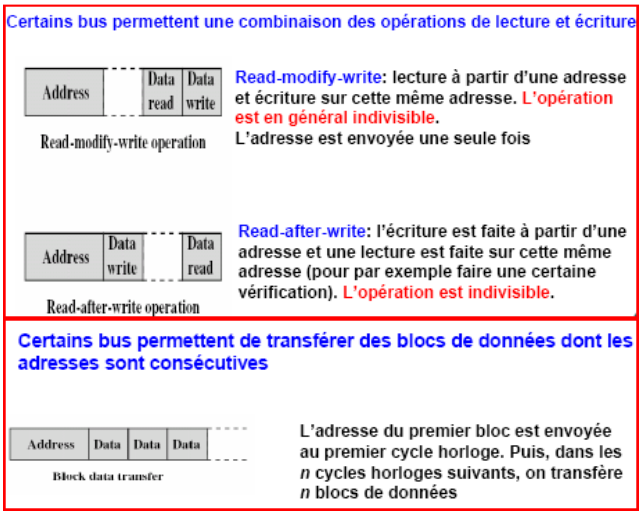
  Exemple simplifié : *Pour lire un mot mémoire, le processeur va placer l'adresse sur le bus d'adresse et après un lapse de temps il va placer une commande de lecture sur le bus. Le module mémoire va décoder l'adresse mémoire et placer les données demandées sur le bus de données. Après un lapse de temps, le processeur lit les données se trouvant sur la ligne de bus.*

- **La largeur du bus**

La **largeur du bus de données** a un **impact sur la performance globale du système** *(plus le bus de données est large, plus on peut transférer plus de bits en une fois)* et la **largeur du bus d'adresses** a un **impact sur la capacité mémoire du système.**

- **Les types de transferts de données**



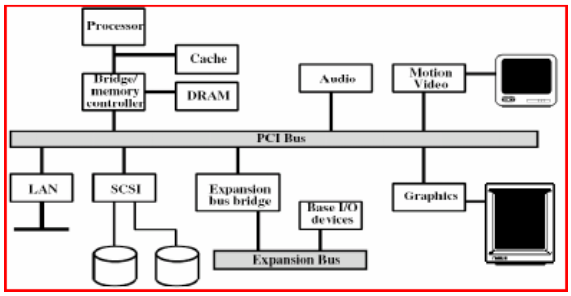


- Le PCI

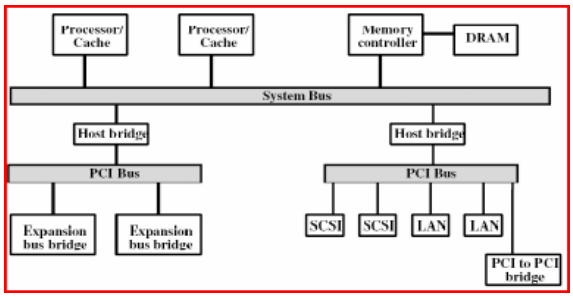
Il s'agit du *« Peripheral Component Interconnect »,* c'est un **bus** à haute bande passante, indépendant du processeur, utilisant le **timing synchrone** et la **méthode d'arbitrage centralisée**.

Usages typiques :

Monoprocesseur



Multiprocesseur



10. Les mémoires.

- Caractéristiques des mémoires.

Dans un ordinateur, on retrouve des **sous-systèmes de mémorisation** à différents niveaux, par exemple les registres dans le **CPU**, la *mémoire principale* et le *disque dur*. Chaque **sous-système de mémorisation** doit satisfaire des exigences spécifiques, par exemple le temps d'accès par le **CPU** à une donnée se trouvant dans un de ses registres doit être plus court que le temps d'accès à une donnée se trouvant en mémoire principale. En conséquence, il est difficile, si pas impossible, de construire tous les **sous-systèmes de mémorisation** en se basant sur une et une seule technologie donc dans un ordinateur typique, on va retrouver *une hiérarchie* de **sous-systèmes de mémorisation**.

Les caractéristiques des mémoires.

♠ **Emplacement**

L'emplacement d'une mémoire est soit interne, soit externe au système.

- ✓ Mémoires internes

  Ce sont des mémoires directement accessibles par le **CPU**

  Exemple : *Registres dans le **CPU,** Mémoire centrale, Mémoire cache*.
- ✓ Mémoires externes

  Ce sont des unités périphériques de stockage accessibles via des **modules d'I/O.** Exemple : *Disques durs, bandes magnétiques.*

♠ **Capacité**

C'est la quantité totale de données pouvant y être stockée

- ✓ Mémoires internes

  La capacité est typiquement mesurée en octets ou mots. *(un octet est un groupe de **8 bits** et un mot a typiquement **8, 16 ou 32 bits.**)*
- ✓ Mémoires externes

  La capacité est mesurée en octets *(bytes).*

♠ **Unité de transfert**

Deux notions de base sont liées aux **mémoires internes**,

- **Le mot**

  C'est l'unité naturelle d'organisation de la mémoire. La taille d'un mot est typiquement égale au nombre de bits utilisé pour représenter un nombre ou à la longueur d'une instruction. (*Le nombre de bits pour représenter un nombre n'étant pas toujours égal au nombre de bits pour une instruction*)
- **L'unité adressable**

  C'est la plus petite unité de mémoire directement adressable. Si les adresses sont longues de **A bits**, il y a **2^A unités adressables**. *Dans certains systèmes, l'unité adressable est le mot. Dans d'autres, c'est le byte.*

Pour la **mémoire principale**, c'est le nombre de bits lus(ou écrits) simultanément. *L'unité de transfert n'est pas forcément égale à un mot ou à une unité adressable.*

Pour une **mémoire externe**, les données sont souvent transférées en unités plus larges que l'on appelle **bloc**.

♠ **Méthode d'accès**

1. Accès séquentiel

   La mémoire est organisée en unités de données appelées **enregistrements** (*records*) qui sont séparés par des *informations d'adressage*. On utilise un **dispositif de lecture/écriture** qui doit être déplacé séquentiellement de son emplacement courant jusqu'à l'emplacement désiré. Le **temps d'accès** à un emplacement donné **dépend** de la position courante du **dispositif de lecture/écriture**, on dit que le *temps d'accès est variable.*

   (Ex : bandes magnétiques).

2. Accès direct

   Chaque unité de données (*bloc ou enregistrement*) a une adresse unique basée sur son emplacement physique. On utilise un **dispositif de lecture/écriture** afin d'accéder à un emplacement donnée. L'accès se fait en **2 étapes** :

   - ❖ Un accès direct à un voisinage de la donnée désirée.
   - ❖ Recherche séquentielle, comptage, ou attente pour atteindre la donnée désirée.

   Le temps d'accès est variable. Ex : *Disques durs.*

3. Accès aléatoire

   Chaque emplacement mémoire adressable possède un mécanisme d'adressage câblé qui est unique, pour accèdes à une donnée il suffit de connaitre son adresse. *Le temps d'accès à une donnée est constant, indépendant des accès précédents.*

   Ex : *Mémoire centrale et certains mémoires caches.*

4. Accès associatif

   Il s'agit d'un type particulier de mémoires à accès aléatoire. Elles sont caractérisées par le fait qu'un mot mémoire est accédé/lu sur base de son contenu et non de son adresse. Comme pour l'accès aléatoire, chaque emplacement adressable a son propre mécanisme d'adressage. *Le temps d'accès à une donnée est constant, indépendant des accès précedents.*

   Ex : *Mémoires caches.*

♠ **Performance**

Elle est caractérisée par 3 paramètres :

1. Le temps d'accès

   Pour des mémoires à accès aléatoire, il s'agit du **temps pour lire ou écrire une donnée.** Plus précisément, c'est **le temps qui s'écoule** entre le moment où une **adresse est présentée au module mémoire** et le moment où **la donnée correspondante est** soit **stockée dans la mémoire** (*écriture*) soit **mise à la disposition du demandeur** (*lecture*).

   Pour des mémoires à accès non-aléatoire, c'est le temps qu'il faut pour positionner le **mécanisme lecture/écriture** sur l'emplacement que l'on veut accéder.

2. Le temps de cycle

   Concept principalement appliqué aux **mémoires à accès aléatoire**.

   Le temps de cycle est la somme du **temps d'accès** et le **temps additionnel** requis pour que la mémoire soit prête pour un autre accès. Ce **temps additionnel** peut provenir de la **stabilisation des lignes de signaux** ou de la **régénération des données** *lorsque la lecture est destructive*.

   Notons que le temps de cycle mémoire est influencé par le bus système et pas par le processeur.

3. La vitesse (ou taux) de transfert

   C'est la vitesse avec laquelle les données peuvent être transférées *de/vers* la mémoire. Pour une mémoire à accès aléatoire, la vitesse est égale à $\frac{1}{temps\ de\ cycle}$.

   Pour une mémoire à accès non-aléatoire, on a la relation : $T_N = T_A + \frac{N}{R}$ où

   - T_N est le temps moyen pour lire ou écrire N bits.
   - T_A est le temps d'accès moyen.
   - N est le nombre de bits
   - R est la vitesse de transfert en bits par seconde.

♠ **Le type physique**

- Semi-conducteur. Ex : *RAM.*
- Surface magnétique. Ex : *Disques, bandes magnétiques*.
- Optique. Ex : *CD.*
- Etc.

♠ **Volatile – Non-Volatile**

1. Mémoires volatiles.

   Ce sont des mémoires qui perdent les informations en absence d'électricité. Ex : *Mémoires à semi-conducteur.*

2. Mémoires non-volatiles.

   L'information stockée y reste sans détérioration jusqu'à ce qu'elle soit délibérément modifiée, l'électricité n'est donc pas nécessaire pour maintenir celle-ci. Ex : *Mémoires à surface magnétique, ROM.*

♠ **Organisation**

Cela concerne la manière dont les bits sont physiquement arrangés pour former des mots mémoires, c'est un problème essentiel et non évident dans la construction des mémoires à accès aléatoire.

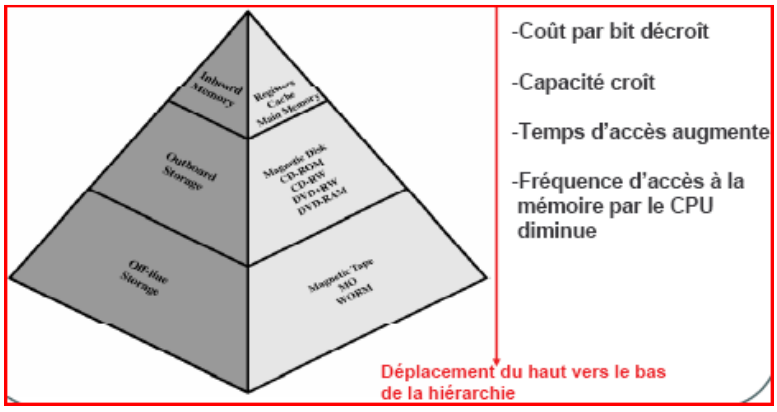
- ## Hiérarchie des mémoires.

3 paramètres entre en compte, **la capacité**, **la rapidité** et le **coût**. Souvent, un compromis doit être trouvé en exploitant diverses technologies. Le compromis se fonde sur les relations suivantes.

- Plus la mémoire est rapide *(temps d'accès petit)* plus le coup par bit est élevé.
- Plus la capacité de la mémoire est grande, moins le coût par bit est élevé, plus le temps d'accès est long (*mémoire moins rapide*).

La solution à ce dilemme est d'utiliser une **hiérarchie des mémoires**.

Une hiérarchie typique :



Impact :

Le succès de cette hiérarchie repose sur l'observation que plus une mémoire se situe loin (plus bas dans la hiérarchie des mémoires) du processeur, moins le processeur y accède souvent. Avec une hiérarchie des mémoires il est possible de réduire le temps d'accès moyen.

Illustration :

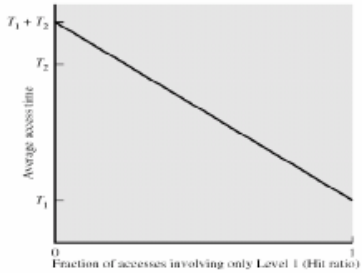
Supposons que le CPU a accès à deux niveaux de mémoire que nous désignons par
- L1
  - Ayant 1000 mots et un temps d'accès de 0.01 µs
- L2
  - Ayant 100000 mots avec un temps d'accès de 0.1 µs

Supposons
- Que le CPU souhaite accéder à un mot de notre mémoire à deux niveaux
- Si le mot mémoire à accéder se trouve en L1, le CPU accède directement à ce mot.
- Si le mot à accéder est dans L2,
  - Le mot mémoire est d'abord transféré de L2 vers L1
  - Puis le CPU accède au mot à partir de L1

Note: nous ignorons ici le temps qu'il faut au CPU pour déterminer dans quelle mémoire se trouve le mot

Nous voulons avoir une idée de l'évolution du temps d'accès moyen à cette mémoire à deux niveaux

- Évolution du temps d'accès moyen en fonction du nombre d'accès qui sont satisfaits par la mémoire L1



H = la fraction des accès, de tous les accès, qui sont satisfaits par le mémoire de niveau 1 (L1)

T_1 = temps d'accès à la mémoire L1 = 0.01µs

T_2 = temps d'accès à la mémoire L2 = 0.1µs

Avec H = 95%, on a que le temps moyen pour accéder à un mot mémoire pour notre exemple est: 0.95(0.01µs) + (0.05)(0.01µs + 0.1µs) = 0.015 µs, plus proche de T_1

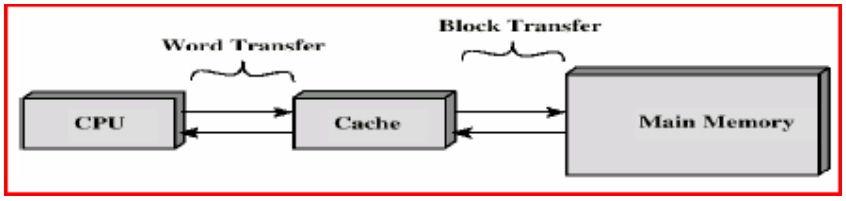
Localité des références :

Nous avons vu que plus on descend dans la hiérarchie des mémoires, moins le processeur accède souvent à l'unité mémoire, cette observation est basée sur le **principe de localité des références.** *Lors de l'exécution d'un programme, les références mémoires faites par le* ***CPU*** *tendent à se regrouper pour les raisons suivantes :*

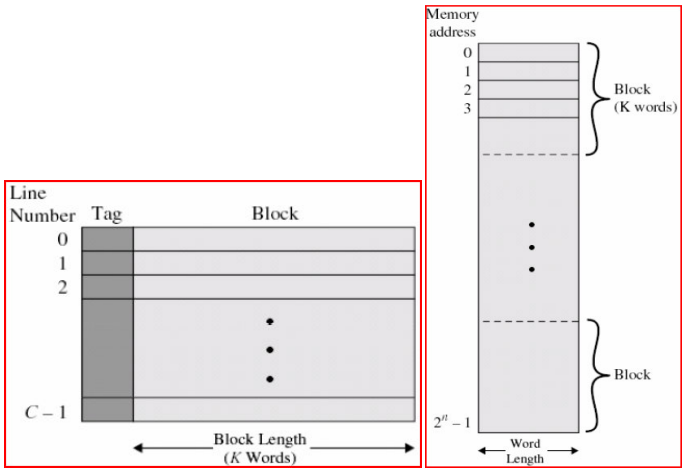
- Les boucles dans les programmes, sous-programmes, etc.
- L'utilisation des tableaux dans des programmes.

- ## La mémoire cache.

Il s'agit d'une mémoire à faible capacité mais très rapide, interposée souvent entre le **CPU** et la mémoire principale, elle contient une copie des portions de la mémoire centrale. Son utilisation est motivée par le principe de la **localité des références**.



Ex : Voici un cache et la mémoire principale.



Sur cet exemple, chaque mot mémoire a une adresse unique de *n* bits. Pour faciliter les transferts vers le cache, la mémoire principale est organisée en blocs de K mots, il y a au total M blocs. Le cache a quant à lui C lignes de K mots chacune. « *Tag* » indique le bloc courant stocké dans une ligne (ce « *Tag* » est nécessaire car C est très petit par rapport à M).

$$C \ll M = \frac{2^n}{K}$$

Principe : Lorsque le **CPU** veut lire un mot mémoire, un test est effectué pour déterminer si le mot est dans le cache,

- Si le mot est dans le cache, celui-ci est délivré au **CPU**. ➔ Pas d'accès à la mémoire principale.
- Si le mot n'est pas dans le cache, un bloc de la mémoire principale, contenant le mot désiré, est transféré dans le cache et le mot désiré est fourni au **CPU**.

Grâce au principe de la localité des références, il y a des fortes chances que le prochain mot mémoire que le **CPU** voudra lire sera dans la mémoire cache.

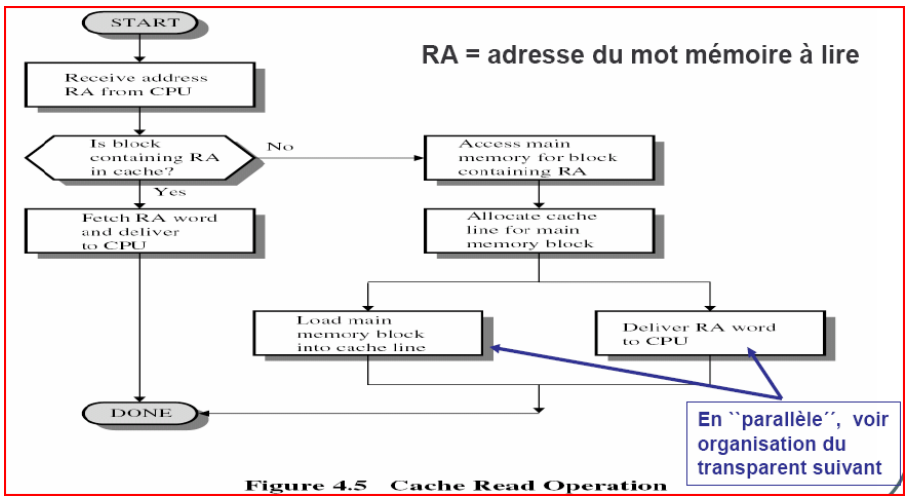
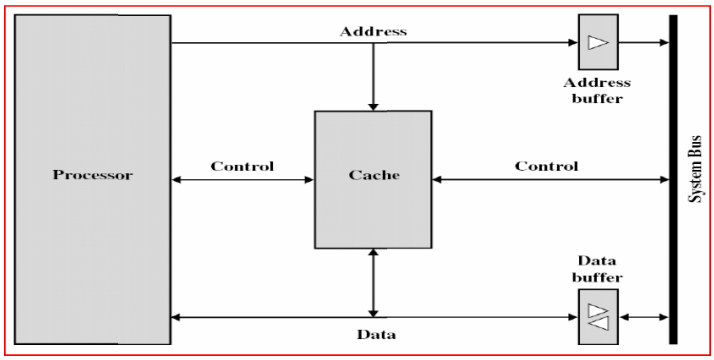


Figure 4.5 Cache Read Operation

Une organisation typique de la mémoire cache :



Quelques éléments dans la conception du cache.

Taille du cache.

La taille du cache doit être **suffisamment petite** pour que le coût total (*cache+mémoire principale*) par bit soit comparable au coût de la mémoire principale seule et **suffisamment grande** pour que le temps d'accès de l'ensemble soit proche du temps d'accès du cache.

Plus le cache est grand plus il faut de portes logiques pour son adressage.

Les caches de grande taille tendent à être plus lents que les caches de petite taille.

Fonctions de correspondance.

Voir transparents.

Algorithmes de remplacement.

Voir transparents.

Techniques d'écriture : *Voir transparents.*

11. Les mémoires internes.

- Mémoires principales à semi-conducteurs.

Les 2 formes de base pour les mémoires à semi-conducteur à accès aléatoire son la **RAM dynamique (*DRAM*)** et la **RAM statique (*SRAM*).** La **SRAM** est plus rapide, plus coûteuse et moins dense que la **DRAM** et elle est utilisée pour la mémoire cache alors que **DRAM** est utilisée pour la mémoire principale.

Des techniques de correction d'erreurs sont souvent utilisées dans des systèmes de mémorisation pour améliorer la fiabilité de ces systèmes. Ces techniques sont basées sur l'utilisation de la *redondance*, à chaque séquence de bits de données, on associe un code correcteur d'erreur qui consiste en des bits additionnels qui sont fonction des bits de données et lorsqu'un bit de données devient erroné, cela est détecté à l'aide du code et bien souvent corrigé.

Les premiers ordinateurs utilisaient des mémoires principales à tores magnétiques. Avec la microélectronique, l'utilisation des puces à semi-conducteurs est devenue quasi-universelle pour les mémoires principales, on parle alors de **mémoires à semi-conducteurs.**

L'élément de base d'une mémoire à semi-conducteurs est la cellule mémoire.

- Propriété des cellules mémoires :

2 états stables, qui peuvent être utilisés pour représenter les binaires 1 et 0. On peut y écrire (au moins une fois), pour positionner l'état et on peut les lire pour obtenir l'état.

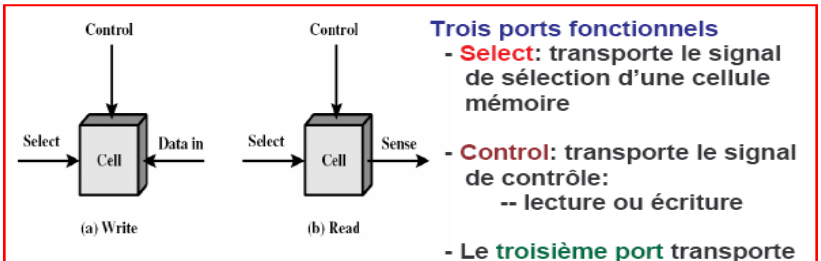


Figure 5.1 Memory Cell Operation

Trois ports fonctionnels
- **Select**: transporte le signal de sélection d'une cellule mémoire
- **Control**: transporte le signal de contrôle:
  -- lecture ou écriture
- Le **troisième port** transporte le signal de positionnement de l'état (écriture) ou le signal de livraison de l'état (lecture)

- Types de mémoire à semi-conducteurs. (toutes des mémoires à accès aléatoire)

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

- La RAM

Il s'agit de la mémoire à accès aléatoire à semi-conducteur la plus courante, le terme **RAM** est souvent utilisé pour faire référence à la mémoire principale. Ce terme est un peu trompeur car toutes les mémoires à semi-conducteurs listées au transparent précédent sont à accès aléatoire.

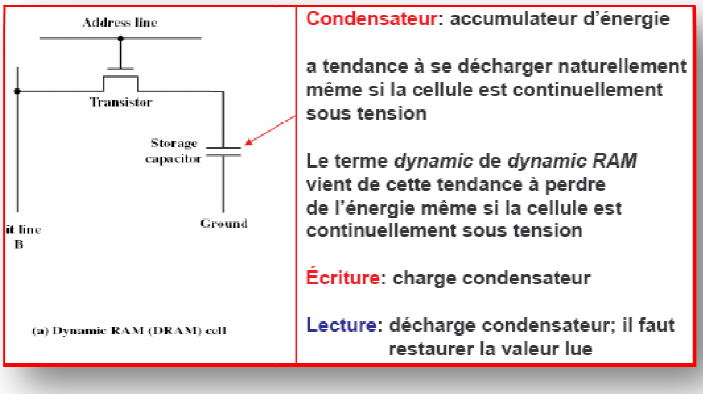
  - Caractéristiques principales :

Ecriture et lecture assez simple et rapide et réalisés au moyen de signaux électriques. La **RAM** est une mémoire volatile et peut donc être utilisée que pour un stockage temporaire d'informations.

Les 2 formes traditionnelles de **RAM** utilisées dans des ordinateurs sont la **DRAM** et la **SRAM**.

1. La **DRAM**.
   Elle est constituée de cellules de mémoire qui stockent des données sous forme de charge électrique dans des condensateurs (*accumulateurs d'énergie*). La présence ou l'absence de charge est interprétée comme un 1 ou un 0. Les condensateurs ont cependant une tendance naturelle à se décharger, en conséquence la DRAM nécessite un rafraîchissement périodique pour maintenir les données stockées.

   - Structure typique d'une cellule **DRAM :**

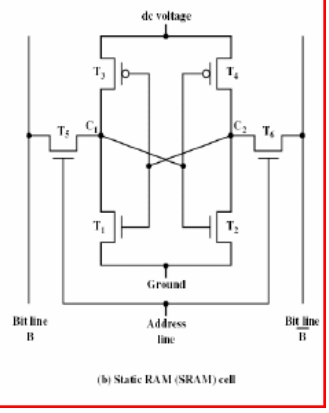


   - Caractéristiques :
     - Plus facile à construire
     - Plus dense
     - Moins coûteuse
     - Plus lente
     - Souvent utilisée pour les mémoires principales.
     - Une cellule **DRAM** est essentiellement analogique,

     Un condensateur peut stocker une valeur d'une charge électrique contenue dans une plage de valeurs, une valeur seuil permettra de déterminer la valeur binaire stockée.

2. La **SRAM.**

- Une mémoire RAM statique (SRAM) est un composant numérique utilisant les mêmes éléments logiques que l'on trouve dans le processeur
- Les valeurs binaires sont stockées au moyen des configurations des bascules
  - Pas de décharge comme dans le cas des DRAM
    - Pas besoin de rafraîchissement
- Toute donnée stockée y reste tant qu'il y a du courant
- Plus complexe
- Plus coûteuse
- Plus rapide
- Souvent utilisée pour des mémoires caches



- <u>Comparaison entre la DRAM et la SRAM.</u>

Elles sont toutes les deux **volatiles**.

Une cellule de mémoire dynamique est plus simple et plus petite qu'une cellule de mémoire statique ➔ la **DRAM** est plus dense.

La **DRAM** est moins coûteuse que la **SRAM** mais elle nécessite un ensemble de circuits de rafraichissement et est du coup plus lente. Les cellules de la **SRAM** sont faites à l'aide de bascules. La **DRAM** est utilisée pour les mémoires principales et la **SRAM** pour la mémoire cache.

- <u>Les types de ROM</u>
  - <u>La ROM</u>

    C'est une puce mémoire dans laquelle les données sont enregistrées lors de la fabrication, on peut lire à partir d'une **ROM** mais on ne peut pas y écrire. C'est une **mémoire non-volatile**. Elle est utile pour stocker des programmes critiques, tels que le BIOS (*Basic Input/Output System)* qui détermine ce que l'ordinateur peut faire sans accéder aux programmes stockés sur le disque.
  - <u>La PROM</u>

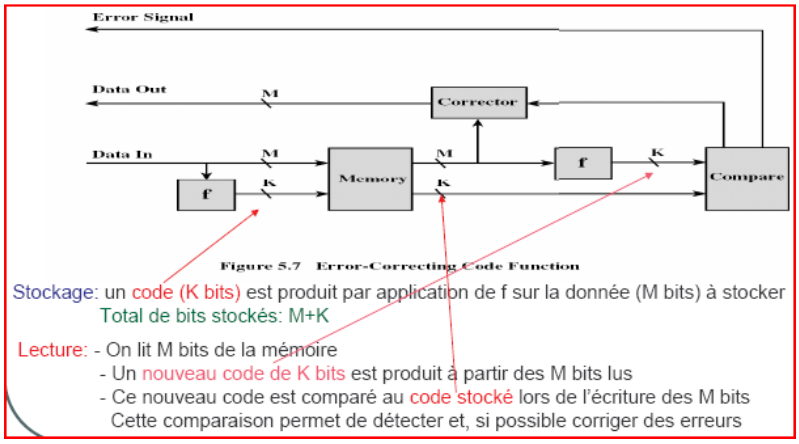
    C'est une puce mémoire dans laquelle on ne peut écrire qu'une seule fois, donc lorsqu'un programme y est stocké, il y reste (virtuellement) pour toujours. La **PROM** est une mémoire non-volatile, et elle est construite comme une mémoire vide qui est programmable électriquement par le constructeur ou le client, pas nécessairement au moment de la fabrication de la puce, ce qui la rend différente de la **ROM**.
  - <u>La mémoire à lecture majoritaire</u>

    Il s'agit d'une variante de la **ROM**, elle est utile pour les applications où les opérations de lecture sont plus fréquentes et où une mémoire non volatile est nécessaire. Il en existe 3 formes courantes :

- EPROM : *Erasable Programmable Read-Only Memory*
  C'est un type spécial de **PROM** qui peut être reprogrammée, pour ce faire il faut tout d'abord l'effacer entièrement en exposant la puce à des rayons **UV**. La lecture et l'écriture se font à l'aide de signaux électriques. L'unique différence avec la **PROM** c'est que l'**EPROM** est **modifiable**.
- EEPROM : *Electrically Erasable Programmable Read-Only Memory*
  C'est une **PROM** spéciale qui peut être effacée électriquement. On peut y écrire à tout moment sans avoir à effacer toutes les cellules au préalable, les données étant écrites ou effacées un octet à la fois et une écriture prenant plus de temps qu'une lecture. Elle coûte plus chère qu'une mémoire **EPROM** mais elle est moins dense, quelques bits par puce.
- Flash Memory : *Mémoire à lecture majoritaire avec effacement très rapide.*
  Le nom de flash memory vient du fait qu'une telle mémoire peut être reprogrammée très vite. C'est une mémoire intermédiaire, en termes de coût et fonctionnalité, entre la **EPROM** et la **EEPROM**. Elle est effaçable électriquement comme la **EEPROM** mais plus rapidement, en quelques secondes. Contrairement à une **EEPROM**, la mémoire flash permet d'écrire ou d'effacer uniquement des blocs à la fois, ce qui implique une écriture plus rapide que l'**EEPROM.**

- Détection et correction d'erreurs.
  - Erreurs permanentes
    Défectuosités physique affectant une ou plusieurs cellules mémoires, celles-ci une fois affectées ne peuvent plus stocker l'information de manière fiable. Une telle cellule est soit bloquée sur la valeur 1 ou 0, soit elle change arbitrairement entre 1 et 0. De telles erreurs peuvent êtres causés par des défauts de fabrication ou un mauvais environnement.
  - Erreurs intermittentes
    Ce sont des évènements aléatoires, non destructives causant le changement du contenu de une ou plusieurs cellules mémoires, sans endommager la mémoire. Les cellules mémoires affectées ne sont pas détruites, seul leur contenu est changé. Elles peuvent être causées par une coupure de courant ou un petit dérangement temporaire.

- La logique illustrée



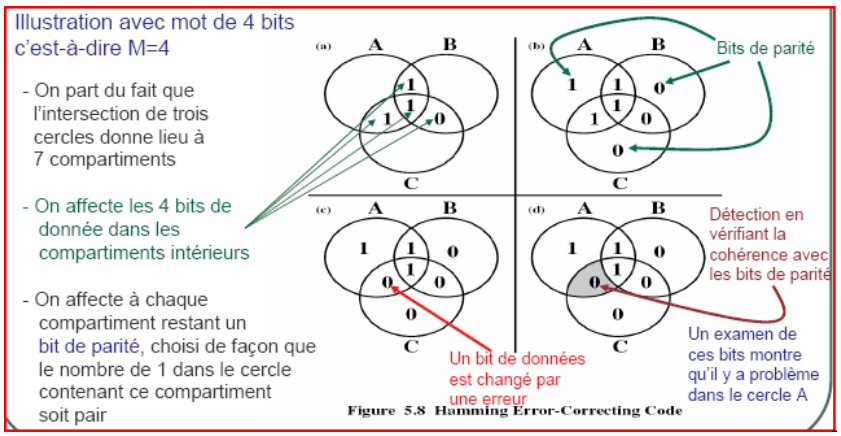
Résultat de la comparaison :

- Aucune erreur n'est détectée ➔ les bits lus sont envoyés au module demandeur.
- Une erreur corrigible est détectée ➔ Les M bits et le code stockés sont passés au correcteur, qui corrige l'erreur et envoie la donnée corrigée au module demandeur.
- Une erreur non corrigible est détectée ➔ Un rapport est retourné au module demandeur.

Les codes opérants de cette façon sont appelés **codes correcteurs d'erreurs**, ces codes sont caractérisés par le nombre de bits erronés, dans un mot, qu'ils permettent de détecter et de corriger.

- Le code de Hamming

C'est l'un des codes les plus simples. Illustration :



Pour mettre en évidence les concepts sous-jacents au code de **Hamming**, développons un code de **Hamming** capable de détecter et corriger un seul bit erroné sur des mots de 8 bits (c à d M=8).

Etape 1: détermination de la longueur du code

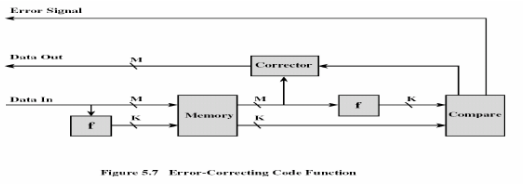


Figure 5.7 Error-Correcting Code Function

- Le comparateur reçoit deux entrées de K bits
- La comparaison bit par bit se fait à l'aide du OU-EXCLUSIF des deux entrées
- Le résultat de cette comparaison est appelé mot syndrome

- Exemple de comparaison bit par bit à l'aide du OU-EXCLUSIF (noté ⊕)

0 1 1 1
⊕ 0 0 0 1
Mot syndrome = 0 1 1 0

Le bit en position *i* du mot syndrome est
1: si les deux bits comparés en position *i* sont différents

0: si les deux bits comparés en position *i* sont égaux

Cet exemple montre que le mot syndrome aura le même nombre de bits que chacune des chaînes de bits comparées

Le mot syndrome a **K bits** de long, sa valeur décimale peut donc être une des valeurs $\in \{0,1,2,\ldots 2^k-1\}$. On va donc utiliser ces valeurs de la façon suivante :

- La valeur **0** indique qu'aucune erreur n'a été détectée
- Les valeurs **1,2,…,2^k-1** servent à indiquer en cas d'erreur, que bit est erroné et puisque le bit érroné peut être un des M+K bits, nous devons avoir $2^K-1 \geq M+K$ (**).

L'inégalité (**) nous donne le nombre de bits nécessaires pour corriger un seul bit erroné dans un mot de données contenant M bits. Ex : pour M=8, nous avons :

- Pour K=3, $2^3-1 = 7 \not\geq 8+3$, la relation (**) n'est pas satisfaite. On ne peut donc pas prendre K=3.
- Pour K=4, $2^4-1 = 15 \geq 8+4$, la relation (**) est satisfaite, donc avec 8 bits de données (M) nous avons besoin de 4 bits de code (K).

	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
Data Bits	Check Bits	% Increase	Check Bits	% Increase
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

- Caractéristiques du mot syndrome.

Pour notre exemple, générons le mot syndrome de 4 bits pour 8 bits de données avec les contraintes suivantes :

- Si le mot syndrome ne contient que des 0, alors aucune erreur n'est détectée.

- Si le mot syndrome contient un et un seul bit mis à 1, alors il y a une erreur dans un des 4 bits de contrôle (code). *(Aucune correction n'est requise)*
- Si le mot syndrome contient plus d'un bit mis à 1, alors la valeur numérique du mot syndrome indique la position du bit de donnée erronée

✓ Pour générer le mot syndrome de 4 bits qui satisfait les contraintes présentes, on arrange les bits de données et les bits du code sur un mot de 12 bits.

✓ Les bits de ce mot de 12 bits sont numérotés comme indiqué sur le tableau ci-dessous.

✓ Tout bit dont la position est une puissance de 2 est considéré comme bit de contrôle et tout autre bit est considéré comme bit de donnée.

Représentation binaire des positions de bits

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check Bit					C8				C4		C2	C1

Il nous reste maintenant à calculer la valeur de chaque bit de contrôle

Calcul des bits de contrôle

Chaque bit de contrôle C_i est le OU-EXCLUSIF de tous les bits de données de D_j qui sont tels que le *« position number »* (représentation binaire de la position) de D_j contient un 1 à la même position que le *« position number »* de C_i. Pour notre exemple, on obtient C_i, comme suit :

$$C1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7$$
$$C2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7$$
$$C4 = D_2 \oplus D_3 \oplus D_4 \oplus D_8$$
$$C8 = D_5 \oplus D_6 \oplus D_7 \oplus D_8$$

Supposons que la donnée de 8 bits en entrée est *00111001, vu comme* $D_8D_7D_6D_5D_4D_3D_2D_1$. Les bits de contrôle sont :

$$C_1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C_2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C_4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$
$$C_8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

Code de Hamming: $C_8C_4C_2C_1 = 0111$

Supposons maintenant que lors de l'écriture de la donnée, une erreur change le bit de donnée D_3 de 0 en 1. Avec ce flip de D_3 de 0 en 1, la donnée devient : 00111101 vu comme $D'_8D'_7D'_6D'_5D'_4D'_3D'_2D'_1$. Lorsque l'on calcule les nouveaux bits de contrôle au moment de la lecture, on obtient :

$C_{1\_new} = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$
$C_{2\_new} = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$
$C_{4\_new} = 0 \oplus 1 \oplus 1 \oplus 0 = 0$
$C_{8\_new} = 1 \oplus 1 \oplus 0 \oplus 0 = 0$

$C_{1\_new} = D'_1 \oplus D'_2 \oplus D'_4 \oplus D'_5 \oplus D'_7$
$C_{2\_new} = D'_1 \oplus D'_3 \oplus D'_4 \oplus D'_6 \oplus D'_7$
$C_{4\_new} = D'_2 \oplus D'_3 \oplus D'_4 \oplus D'_8$
$C_{8\_new} = D'_5 \oplus D'_6 \oplus D'_7 \oplus D'_8$

Une comparaison avec OU-EXCLUSIF des anciens bits de contrôle avec les nouveaux bits de contrôle avec les nouveaux bits de contrôle donne le mot syndrome qui est différent de 0000 indiquant qu'il y a un bit erroné.

	C_8	C_4	C_2	C_1
⊕	C_{8_new}	C_{4_new}	C_{2_new}	C_{1_new}
	0	1	1	0

La position du bit erroné, est donnée par la valeur entière du mot syndrome. 0110 est la valeur binaire de 6. Donc le bit en position 6 sur le mot de 12 bits (8bits de données + 4 bits de code) est celui qui est erroné. Ce qui est correcte car le bit en position 6 correspond au bit de donnée D_3.

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word stored as	0	0	1	1	0	1	0	0	1	1	1	1
Word fetched as	0	0	1	1	0	1	1	0	1	1	1	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Check Bit					0				0		0	1

Le code que nous venons de décrire est connu sous le nom de code de correction d'erreur simple (Single-Error-Correcting Code, SEC). Bien souvent, les mémoires à semi-conducteur sont équipées de code correction d'erreur simple et de détection d'erreurs doubles (SEC-DED) (*Double Error Detecting Code*).

12. Les mémoires externes.

- Les disques magnétiques.
  - Qu'est-ce que c'est ?

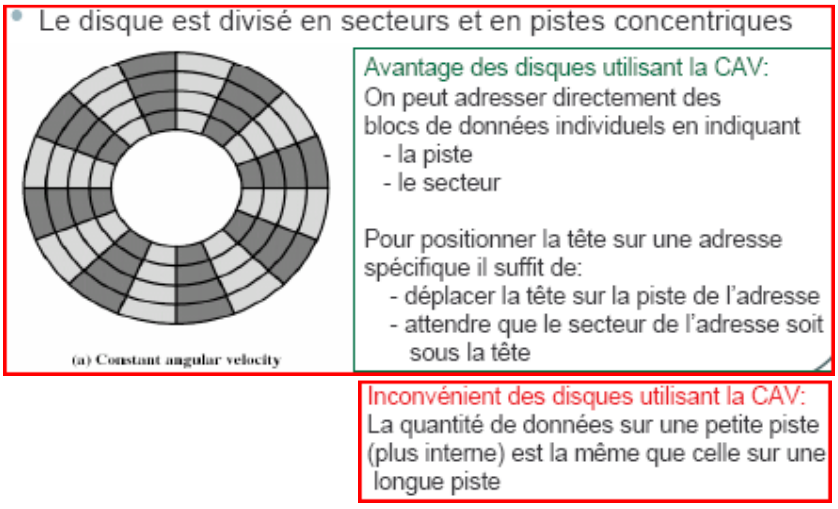
    Un disque magnétique est un plateau rond composé de matériau non magnétique, appelé **substrat** recouvert de matériau magnétisable. Dans le passé, le **substrat** était en *aluminium* ce qui le rendait peu fiable, il est remplacé actuellement par du *verre*.
  - Lecture et écriture.

    Les **données** sont **enregistrées et lues** via un dispositif appelé *tête*. Pendant une écriture ou une lecture, la tête reste immobile tandis que le plateau tourne. Dans de nombreux systèmes il existe 2 têtes une pour *lire* et une autre pour *écrire*.
  - Organisation.

    Transparents 5->8.

    Une façon de permettre à la tête de lire tous les bits à la même vitesse consiste à accroître l'espacement entre les bits de données enregistrés dans des segments de disques. Ainsi on peut alors scanner les informations en faisant tourner le disque à une vitesse constante que l'on appelle vitesse angulaire constante (*Constant Angular Velocity, **CAV***).

    Avec l'utilisation de cette **CAV** on a que :

    Dans un disque à vitesse angulaire constante, la densité de bits par unité de longueur augment quand on se déplace des pistes extérieurs vers des pistes intérieurs, en conséquence, la densité de base des disques avec CAV est limité à la densité maximum de la disque la plus interne. Il faut donc trouver un moyen pour accroître la densité des pistes (*le nombre de bits par piste*).

    Transparents 13->16.

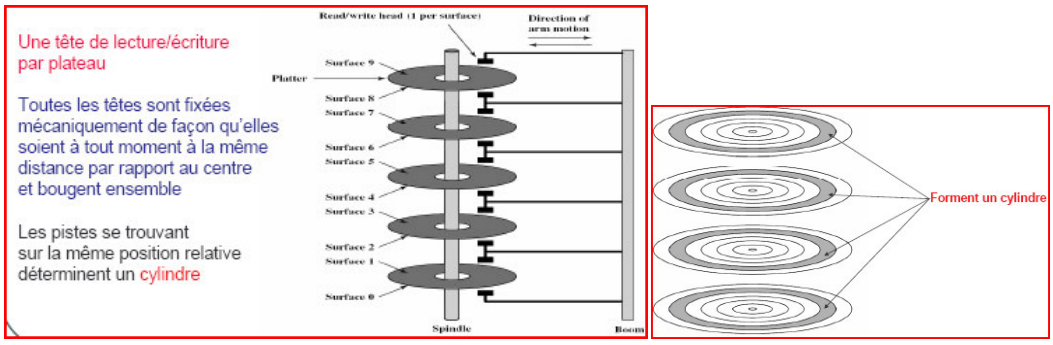
- Caractéristiques physiques.

  Les disques magnétiques peuvent être classés sur base de divers critères :

Head Motion	**Platters**
Fixed Head (one per track)	Single-Platter
Movable Head (one per surface)	Multiple-Platter
Disk Portability	**Head Mechanism**
Nonremovable disk	Contact (floppy)
Removable disk	Fixed gap
	Aerodynamic gap (Winchester)
Sides	
Single-Sided	
Double-Sided	

  - Les disques à tête fixe (par rapport à la direction radiale du plateau) ont une tête de lecture par piste, chacune de ces têtes sont montées sur un bras rigide qui s'étend sur toutes les pistes. (*De tels systèmes sont rares de nos jours)*
  - Les disques à tête déplaçable possèdent une seule tête de lecture pour toutes les pistes du plateau, celle-ci est montée sur un bras qui peut s'allonger ou se rétracter pour permettre à la tête d'être positionnée sur n'importe quelle piste.
  - Le disque lui-même est monté dans une unité de disque comportant un bras, un dispositif (*bras*) qui fait tourner le disque et l'électronique nécessaire pour la **lecture** et l'**écriture**.
  - Le disque non amovible est fixé « virtuellement » en permanence dans l'unité de disque. (Ex : Disque dur d'un ordinateur.)
  - Le disque amovible peut être retiré de l'unité de disque et remplacé par un autre. (Ex : Disquette.)
  - Le disque à double face possède une couche magnétisable sur 2 surfaces. (*C'est le cas de la plupart des disques durs*)
  - Le disque à simple face possède une seule couche magnétisable, ces disques sont souvent moins coûteux.
  - Le disque à plusieurs plateaux possède plusieurs plateaux empilés au lieu d'un.

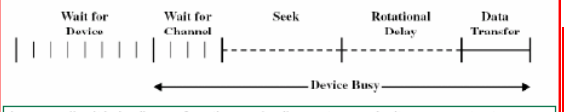
    Illustration.



- Paramètres de performance.

  Le fonctionnement des entrées/sorties du disque dépend de *l'ordinateur, du système d'exploitation, de la nature des canaux d'entrées/sorties, du matériel du contrôleur du disque.*

- Diagramme temporel typique des transferts des entrées/sorties



Lorsque l'unité de disque fonctionne, le disque tourne à vitesse constante

Pour lire ou écrire, il faut alors sélectionner la piste désirée et se placer au début du secteur désiré de cette piste

Sélection d'une piste consiste en:
- un déplacement de la tête dans le cas de disque à tête déplaçable
- une sélection électronique d'une tête dans le cas de disque à tête fixe

Seek time: pour une unité de disque à tête amovible, c'est le temps qu'il faut pour positionner le tête sur la piste désirée

Rotational delay: Que ce soit pour une unité de disque à tête fixe ou à tête amovible, une fois que la bonne piste est sélectionnée, il faut attendre que la tête soit positionnée sur le début bon secteur désiré. Ce temps d'attente est appelé temps de rotation ou rotational delay.

Access time = seek time + rotationnal delay

Transfer time: une fois la tête positionnée en début de secteur, c'est le temps qu'il faut pour que *tout le secteur* passe en dessous de la tête (dépend de la vitesse de rotation du disque)

Paramètres typiques :

Characteristics	Seagate Barracuda 180	Seagate Cheetah X15-36LP	Seagate Barracuda 36ES	Toshiba HDD1242	IBM Microdrive
Application	High-capacity server	High-performance server	Entry-level desktop	Portable	Handheld devices
Capacity	181.6 GB	36.7 GB	18.4 GB	5 GB	1 GB
Minimum track-to-track seek time	0.8 ms	0.3 ms	1.0 ms	—	1.0 ms
Average seek time	7.4 ms	3.6 ms	9.5 ms	15 ms	12 ms
Spindle speed	7200 rpm	15K rpm	7200	4200 rpm	3600 rpm
Average rotational delay	4.17 ms	2 ms	4.17 ms	7.14 ms	8.33 ms
Maximum transfer rate	160 MB/s	522 to 709 MB/s	25 MB/s	66 MB/s	13.3 MB/s
Bytes per sector	512	512	512	512	512
Sectors per track	793	485	600	63	—
Tracks per cylinder (number of platter surfaces)	24	8	2	2	2
Cylinders (number of tracks on one side of platter)	24,247	18,479	29,851	10,350	—

- RAID
  - Qu'est-ce que c'est ?

    Il s'agit d'une organisation d'unités de disques dans laquelle un ensemble de disques est vu par le système d'exploitation comme un seul disque logique, les données étant distribuées sur les différents disques.
  - Objectifs
    - Fournir une grande capacité de stockage à partir de disques peu coûteux.
    - Augmenter la performance par des accès parallèles aux données.
    - Fournir une haute disponibilité des données par la *tolérance aux pannes de disques*.

⇨ *Transparents 29->41.*

- Bandes Magnétiques

⇨ *Transparents 42 ->48.*

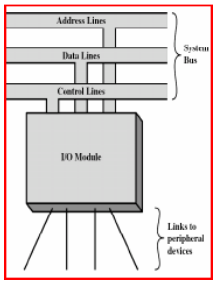
13. Les modules d'entrées/sorties.

- Interface entre l'ordinateur et le monde extérieur

  L'architecture des entrées/sorties d'un ordinateur constitue son interface vis-à-vis du monde extérieur. Celle-ci *permet de contrôler* systématiquement les interactions entre l'ordinateur et le monde extérieur et elle fournir au *système d'exploitation* **toutes les informations nécessaires** à la gestion effective des **entrées/sorties**.

- Modèle générique de module d'I/O.

  Chaque module d'I/O **est** typiquement connecté au bus système, **contrôle** une ou plusieurs unités périphériques et **possède** une certaine *« intelligence »* qui lui permet d'accomplir la fonction de communication entre le **bus système** et les **unités périphériques**.



- Nécessité

  Les unités périphériques ne sont pas connectées directement au bus système car :

  - Il existe plusieurs **classes** d'*unités périphériques* et donc chaque unité périphérique a **un mode opératoire** qui est propre aux unités de sa classe. *Il ne serait pas pratique d'incorporer tous ces modes opératoires au processeur.*
  - **La vitesse de transfert** de certaines *unités périphériques* est souvent plus lente que celle de la mémoire principale ou du processeur. *Une connexion directe des périphériques conduirait donc à une utilisation non effective du bus système qui est, lui, à haut débit.*
  - Certaines *unités périphériques* sont plus **rapides** que la mémoire principale ou le processeur. *De nouveau une connexion directe des périphériques au bus système conduirait à une mauvaise utilisation de ce dernier.*
  - Les *unités périphériques* ont souvent des **formats de données** et des **longueurs de mots mémoires** différents de ceux de l'ordinateur sur lequel elles sont connectées. *Une connexion directe des périphériques conduirait donc à l'incorporation, dans le processeur, de la capacité à gérer ces différences.*

- Fonctions

  Un module d'I/O a *essentiellement* **2 fonctions** :
  - Servir d'interface au **processeur** et à la **mémoire principale** via le *bus système*.
  - Servir d'interface à une ou plusieurs **unités périphériques** vie des lignes spécifiques de données vers ces unités périphériques.
- Unités Périphériques
  - Qu'est-ce que c'est ?

    Il s'agit d'un *dispositif* pouvant **échanger des données** avec le **CPU** ou la **mémoire principale** via une *ligne de communication* et un *module d'I/O*.

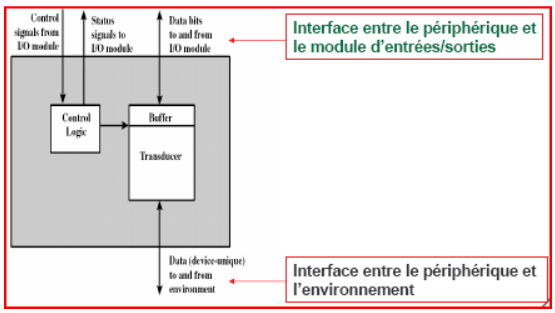
    La *ligne de communication* est utilisée pour **transporter** des informations de **contrôle** de ou vers **l'unité périphérique,** des informations *d'état du périphérique* ou **des données**.
  - Classification

    Les **unités périphériques** peuvent être classées suivant **3 catégories** :
    - ✓ Les unités périphériques **lisibles par des humains**, appropriées pour communiquer avec l'utilisateur de l'ordinateur.

      Exemple : *écran, imprimante, clavier.*
    - ✓ Les unités périphériques **lisibles par des machines**, appropriées communiquer avec des *équipements*.

      Exemple : *disques et bandes magnétiques.*
    - ✓ Les unités périphériques **de communication distante**, appropriées pour communiquer avec un appareil distant, qui peut lui-même être lisible par des humains ou par des machines.

      Exemple : *terminal, modem, carte réseau.*
  - Remarque
    - D'un point de vue *fonctionnel*, les disques et bandes magnétiques sont des **sous-systèmes de mémorisation.**
    - D'un point de vue *structurel*, les unités de disques et bandes magnétiques sont **contrôlées par des modules d'entrées/sorties**. *Par conséquent, d'un point de vue structurel, les disques et bandes magnétiques sont des unités périphériques.*
  - Schéma général



- Caractéristiques fondamentales
  - Les **signaux de contrôle** sont envoyés par le *module d'I/O* et déterminent la tâche que l'unité périphérique doit accomplir.
    Exemples de tâches :
    - *Envoyer des données au module d'I/O (INPUT ou READ).*
    - *Accepter des données venant du module d'I/O (OUTPUT ou WRITE).*
    - *Reporter son état.*
    - *Effectuer une opération de contrôle spécifique (comme positionner une piste du disque).*
  - Les **signaux d'état** (*status signal*) sont envoyés par *l'unité périphérique* au **module d'I/O**. Ils permettent à l'unité périphérique de signifier son état au module d'I/O (Ex : *READY, NOT-READY*).
  - Les **signaux de données** (*data*) sont des ensembles de bits à transférer entre le *module d'I/O* et le *périphérique*.
  - La **logique de contrôle** (*control logic*) supervise le fonctionnement de l'unité périphérique en réponse aux commandes venant du module d'I/O.
  - Le **transducer** convertit les données :
    - De la forme électrique vers d'autres formes lors des opérations de sortie (*WRITE*).
    - D'une autre forme vers la forme électrique lors des opérations d'entrée (*READ*).
  - Le **buffer** est un *tampon* servant à stocker temporairement des données transférées entre le module d'I/O et l'environnement (*capacité typique : 8 à 16 bits*)
- Clavier/Ecran

  Combinaison la plus utilisée pour des interactions entre un ordinateur et son utilisateur. L'utilisateur **fournit** des entrées via le **clavier**, ces données sont transmises à l'ordinateur et peuvent être **affichées** à l'**écran**.
  L'unité de base utilisée lors d'échange de données est le **caractère** et chaque caractère introduit au clavier est typiquement codé sur 7 ou 8 bits.
  Le codage de texte le plus courant est **IRA** (***I**nternational **R**eference **A**lphabet)* et le code **ASCII** *(**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)* est la version américaine de l'**IRA**.
  - L'IRA.

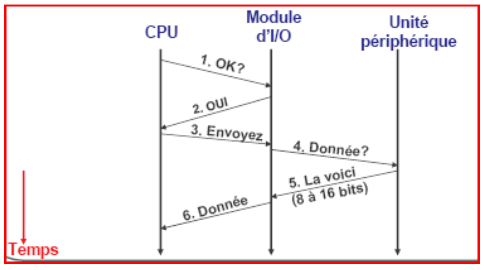
    ➔ *Transparents 17->24.*

- Retour aux fonctions des modules d'I/O

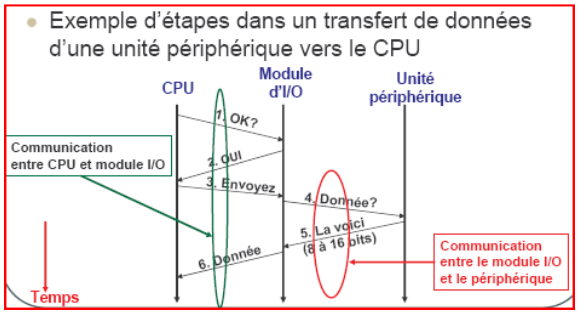
  Les *fonctions principales* d'un module d'I/O peuvent être regroupées suivant les catégories :

  - Contrôle et timing

    Pendant une période de temps quelconque, le processeur peut communiquer avec plusieurs unités périphériques suivant un schéma non prédictible en fonction des demandes du programme exécuté en entrées/sorties. Plusieurs activités, y compris celles concernant l'entrée/sortie des données, partagent des ressources internes telles que la **mémoire principale** et le **bus système**. *Un module d'I/O doit donc remplir une fonction de **contrôle** et de **timing** pour contrôler le flot du trafic entre les unités périphériques et les ressources internes.* Exemple (lors d'un transfert vers le **CPU**):

  - Communication avec le processeur et l'unité périphérique

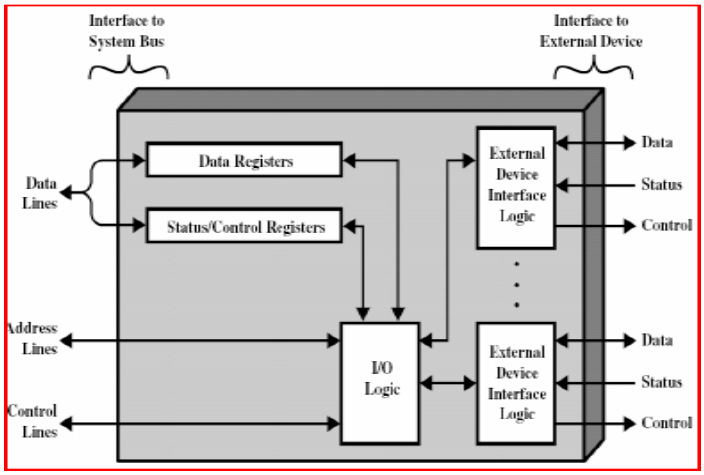
  - Mémorisation temporaire des données

    Les vitesses de transfert de/vers la **mémoire principale** ou le **CPU** sont beaucoup plus grandes que les vitesses de transferts offertes par bon nombre d'*unités périphériques*. *Le module d'I/O doit donc remplir une fonction de **mémorisation temporaire** des données pour permettre une certaine « synchronisation », c à d compenser les différences entre la vitesse de transfert de l'ordinateur (mémoire ou CPU) et les vitesses de transferts des unités périphériques.*

- Détection d'erreurs

  Le module d'I/O est dans bien des cas chargé de détecter et de reporter des erreurs au processeur. Exemples d'erreurs : L'unité périphérique peut reporter une erreur mécanique ou électronique au module d'I/O, le module d'I/O peut donc faire suivre cette information au processeur. (Ex : fourrage de papier dans une imprimante). Des erreurs de corruption de données lors de leur transmission par la ligne reliant le périphérique au module d'I/O sont souvent détectées par le module d'I/O, grâce par exemple aux bits de parité.

- Structure du module d'I/O



  - Data registers

    Les données transitant dans le module d'I/O sont mémorisées temporairement dans un ou plusieurs registres de données.
  - Status/Control registers

    Stocke l'état courant de ou des unités périphériques controlées par le module d'I/O.
  - Un module d'I/O peut contrôler plusieurs unités périphériques

    Il y a une adresse unique pour chacune des unités périphériques contrôlées et il y a une logique spécifique à l'interface de chacune des unités périphériques contrôlées.
  - Address lines

    Elles sont nécessaires car un module d'I/O doit reconnaître et fournir l'adresse ou les adresses de chacune des unités qu'il contrôle.
  - I/O logic

    Interagit avec le processeur vie les lignes de contrôle (*control lines*).

- Grâce au module d'I/O
  - Les **détails** (*tels que les formats et le timing*) d'une grande variété d'unités périphériques sont masquées.
  - Le **processeur** peut alors interagir avec un **module d'I/O** avec quelques commandes simples comme *Lecture, Ecriture, Ouvrir/Fermer un fichier.*

- Terminologie

  On parle de :
  - **Canal d'I/O** *(I/O Channel)* ou **processeur d'I/O** *(I/O processor)* pour désigner un module d'I/O qui prend en charge presque tous les détails du traitement d'I/O et de ce fait, offre une interface de haut niveau au **CPU** (*On trouve ce genre de module d'I/O dans des mainframes*).
  - **Contrôleur d'I/O ou de périphérique** *(I/O Controler)* pour désigner un module d'I/O primitif qui nécessite un contrôle détaillé par le **CPU** (*On retrouve ce genre de module d'I/O dans les micro-ordinateurs)*.
- Techniques d'entrées/sorties et DMA

  Transparents : 33->53

14. Systèmes d'exploitation.

- Points clés.
  - Le système d'exploitation (**OS**, *Operating System*) est le logiciel qui contrôle l'exécution des programmes sur un processeur et gère les ressources du processeur.
  - Certaines fonctions assurées par l'**OS**, telles que le *scheduling des processus* et la *gestion de la mémoire* ne peuvent être accomplies efficacement et rapidement que si le matériel du processeur possède certaines aptitudes pour supporter l'**OS**.
  - Presque tous les processeurs possèdent ces aptitudes :
    - Matériel pour la gestion de la mémoire virtuelle
    - Matériel pour la gestion des processus
  - Le matériel inclut : des *registres spéciaux*, des *buffers* et des *circuits* pour effectuer des tâches de base de la gestion des ressources.
  - Une des fonctions les plus importantes de l'**OS** est le *scheduling* des processus :
    - L'**OS** détermine quel processus devrait s'exécuter à un instant donné.
    - En général, le matériel va de temps en temps interrompre un processus en cours d'exécution afin de permettre à l'**OS** de prendre de nouvelles décisions de *scheduling* de façon à permettre un partage, plus ou moins équitable, du temps du processeur entre un certain nombre de processus.

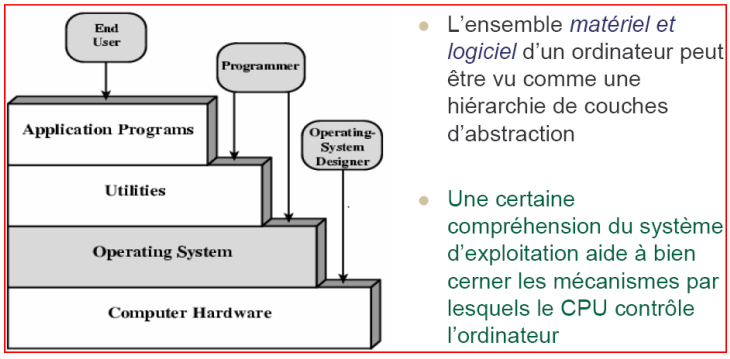
- Une autre fonction important de l'**OS** est la gestion de la mémoire, la plupart des **OS** actuels incluent la capacité de *mémoire virtuelle* qui comporte deux avantages.
  - Un programme peut s'exécuter en mémoire principale sans que l'entièreté du programme et des données soient présents en mémoire principale.
  - La mémoire totale disponible à un programme peut de très loin dépasser la mémoire réelle de l'ordinateur.
- Bien que la gestion de la mémoire soit faite par un composant logiciel, l'**OS** dépend du matériel du processeur :
  - Matériel du processeur pour la *pagination*
  - Matériel du processeur pour la *segmentation*

- <u>Qu'est-ce que c'est qu'un **OS** ?</u>

  En somme, un ordinateur est un système complexe constitué d'un certain nombre de ressources, beaucoup de détails physiques et électroniques et la capacité d'exécuter des programmes.

  Un système d'exploitation est un **programme** qui **gère** les ressources de l'ordinateur pour permettre une utilisation effective de celles-ci, qui **fournit** des services aux programmeurs (*Il sert d'interface entre les programmes utilisateurs d'un ordinateur et le matériel de ce dernier*) et qui **contrôle** l'exécution des programmes des utilisateurs.

- <u>Situation dans la hiérarchie des couches d'abstraction</u>



- <u>Interface entre utilisateurs et ordinateur</u>

  Un utilisateur final s'intéresse surtout aux applications disponibles sur l'ordinateur (*applications bancaires, web*). Les **applications** sont écrites dans des langages de programmation et développées par des programmeurs. Celles-ci, pendant leurs exécutions, et les programmeurs pendant le développement utilisent souvent des programmes utilitaires (éditeurs de texte, compilateur, gestionnaire de fichier, interpréteur de commandes). Le **système d'exploitation** masque les détails du matériel et fournit aux programmeurs/programmes utilitaires, une interface conviviale pour utiliser le système.

- Quelques fonctions

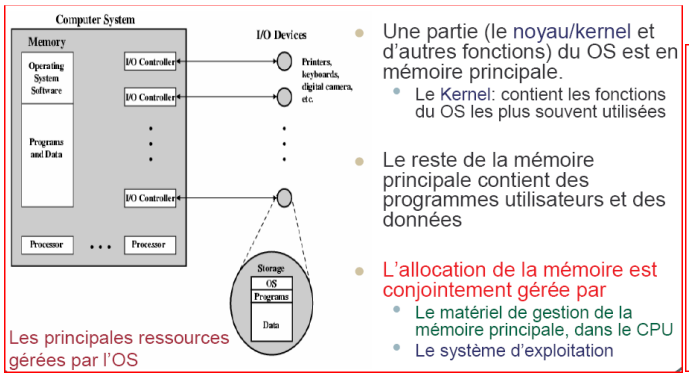
  En général, le système d'exploitation fournit des services dans les domaines suivants :

  - Création de programmes
  - Exécution de programmes
  - Accès aux unités périphériques
  - Contrôle d'accès aux fichiers
  - Contrôle d'accès au système/ressources spécifiques
  - Détection et réponde aux défaillances
  - Comptabilité dans l'utilisation des diverses ressources

- Gestionnaire des ressources

  La particularité des instructions d'un système d'exploitation d'un ordinateur est que celles-ci dirigent le processeur dans l'utilisation des autres ressources de l'ordinateur et l'ordonnancement de l'exécution des autres programmes.

  Pour que des programmes utilisateurs puissent être exécutés, le système d'exploitation doit céder le contrôle du **CPU**, puis le reprendre à un certain moment.



- Type – classification suivant deux dimensions

  - **Première dimension** *(se base sur l'existence ou non d'interactions entre l'utilisateur (ou le programmeur) et l'ordinateur)*

    - **Interactif** – le programmeur/utilisateur interagit directement avec l'ordinateur pour, par exemple, demander l'exécution d'un programme
    - **Batch** – Le programme d'un utilisateur est mis ensemble avec d'autres programmes utilisateurs et soumis à un opérateur de l'ordinateur. Après exécution, les résultats sont imprimés pour l'utilisateur. Les systèmes purement **batch** sont rares de nos jours.

- **Seconde dimension** *(se base sur la présence ou non de la multiprogrammation)*
  - **Multiprogrammation** – Le système d'exploitation essaie de maintenir le **CPU** occupé au maximum en le faisant travailler sur plus d'un programme à la fois. Plusieurs programmes sont chargés en mémoire et le **CPU** commute rapidement entre ces programmes.
  - **Monoprogrammation** – Le système d'exploitation fait travailler le **CPU** sur un seul programme à la fois.

- Les tous premiers ordinateurs

  (des années 1940 aux années 1950) Le programmeur interagit directement avec le matériel de l'ordinateur ➔ pas de système d'exploitation. Le **CPU** était contrôlé à partir d'une console. Les programmes étaient en langage du processeur et devaient être chargés via un dispositif d'entrée (*ex : lecteur de cartes perforées)*.

  2 grosses limitations :

  - Première limitation : **Scheduling** (*ordonnancement/planification des jobs*) – Un utilisateur devait réserver un intervalle de temps pour l'utilisation de l'ordinateur. Si le programme se terminait avant la fin du temps réservé, pendant le temps restant le **CPU** est inexploité. Si le temps réservé s'écoule et que le programme n'est pas terminé, le programme est forcé de se terminer.
  - Seconde limitation : **Setup time** *(préparation à l'exécution).*

    Pour exécuter un programme (appelé *job*) il faut :
    - Charger en mémoire le programme source
    - Charger en mémoire le compilateur
    - Sauvegarder le programme compilé (*programme objet*)
    - Charger en mémoire et lier ensemble le programme objet avec les fonctions de librairie (*édition des liens*).

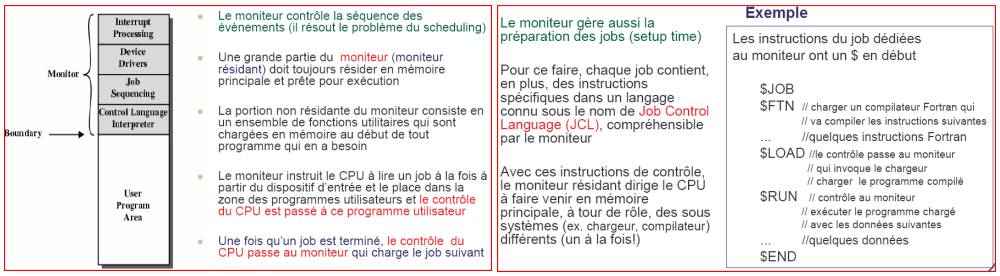
    Chacune de ces étapes pouvait impliquer le montage et le démontage de bandes magnétiques. ➔ Pertes de temps considérables !

- Les systèmes batch simples

  Dès les années 50-60, les pertes de temps dues à la planification et à la préparation des exécutions des programmes deviennent insupportables ➔ Des grosses machines, les mainframes, sont produites, ces machines sont très coûteuses, il faut donc maximiser leur temps d'utilisation. Les premiers systèmes d'exploitation de type batch, appelés **moniteurs** (*ancêtres des systèmes d'exploitation d'aujourd'hui*) sont développés.

  Avec **les moniteurs**, l'**utilisateur** n'a plus un accès direct au **CPU**, il soumet un job *(i.e. un programme)* sur cartes perforées ou sur bandes magnétiques à un

opérateur. L'**opérateur** collecte séquentiellement les jobs et les place sur un dispositif d'entrée utilisé par le moniteur.



On voit qu'un **moniteur** (*système d'exploitation batch simple*) n'est rien d'autre qu'un programme qui, pour atteindre sa mission, s'appuie sur le fait qu'un **CPU** peut lire des instructions à partir de la mémoire principale et pour que le tout fonctionne bien, certains appuis matériels sont désirables.

Les appuis matériels désirables.

- Protection mémoire

  Un programme utilisateur en exécution ne doit pas altérer la zone mémoire occupée par le **moniteur**. Tout tentative, par un programme utilisateur, d'accéder à l'espace mémoire occupée par le moniteur devrait être détectée par le matériel du **CPU** et le contrôle est retourné au **moniteur**. Le **moniteur** se charge alors de terminer immédiatement le job courant avec un message d'erreur et de faire rentrer en mémoire le job suivant s'il y en a un.

- Interruptions

  Pour, par exemple, permettre au système d'exploitation de céder/retirer le contrôle à un programme utilisateur.

- Minuteur *(timer)*

  Un **minuteur** est utilisé pour empêcher qu'un job, monopolise le système. Au début de chaque job, un minuteur est positionné, si le minuteur sonne alors que le job n'est pas terminé, une *interruption* se produit et le contrôle retourne au **moniteur**.

- Instructions privilégiées

  Certaines instructions sont considérées comme *privilégiées* et ne peuvent être exécutées que par le moniteur. Exemple : *Instructions d'I/O, le moniteur a le contrôle de toutes les unités périphériques.* Si le **CPU** rencontre une telle instruction lorsqu'il exécute un programme utilisateur, une *interruption d'erreur*, générée par le matériel du **CPU**, se produit et le contrôle est retourné au **moniteur**. Si un programme utilisateur doit effectuer une I/O, il doit faire une demande au **moniteur**.

- **Observation**

  Le temps **CPU** alterne entre exécution des programmes utilisateurs et exécution du **moniteur**.

2 **sacrifices** : Une partie de la mémoire principale est maintenant allouée au **moniteur** ; et un certain temps **CPU** est consommé par le **moniteur**.

- **Mauvaise exploitation du CPU illustrée**

  Considérons un programme qui traite un fichier d'enregistrements. Ce programme exécute en moyenne 100 instructions par enregistrement.

Read one record from file	0.0015 seconds
Execute 100 instructions	0.0001 seconds
Write one record to file	0.0015 seconds
TOTAL	0.0031 seconds

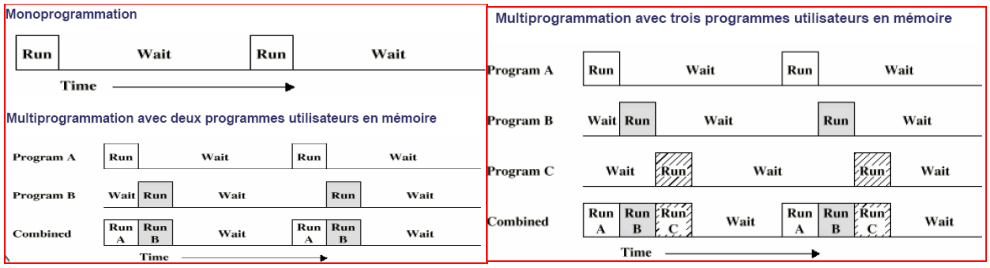
$$\text{Percent CPU Utilization} = \frac{0.0001}{0.0031} = 0.032 = 3.2\%$$

- Les systèmes batch multiprogrammés

  Les systèmes batch simples ont amélioré l'utilisation du **CPU** mais le **CPU** reste tout de même inexploité à de nombreuses occasions lors des entrées/sorties car les unités d'I/O sont très lentes comparé au **CPU**.

  L'idée de base des **systèmes batch multiprogrammés** est d'exécuter un programme utilisateur B lorsqu'un programme utilisateur A attend l'accomplissement d'une opération d'I/O. L'objectif principal d'un système d'exploitation batch multiprogrammé est alors de maximiser l'utilisation du **CPU.**

  - **Illustration**



  - **Support matériel**

    Comme pour les systèmes d'exploitation batch simples, les systèmes d'exploitation batch multiprogrammés ont besoin d'un support matériel pour l'accomplissement de leur **mission.** ***(Maximiser l'utilisation du CPU)***

    Il faut par exemple du matériel qui supporte des entrées/sorties par interruptions et par **DMA**.

  - **Note**

    Avec les systèmes d'exploitation batch multiprogrammés on a que plusieurs programmes sont en mémoire principale (*Ceci nécessite une certaine gestion de la mémoire)* et si plusieurs jobs sont prêts à être exécutés, le processeur doit décider lequel exécuter (*Ceci nécessite des algorithmes de scheduling des jobs*)

- Temps partagés *(time sharing)*

  Permettent à plusieurs utilisateurs d'interagir directement avec l'ordinateur, via des terminaux
  - Exemple: dans le traitement des transactions

  Ici, l'objectif principal du système d'exploitation est de minimiser *les temps de réponse* perçus par les utilisateurs

  La technique de multiprogrammation est utilisée pour gérer plusieurs jobs interactifs
  - Dans ce cas, la technique de multiprgrammation prend le nom de temps partagé (time sharing), puisque le temps CPU est partagé entre plusieurs utilisateurs qui accèdent simultanément au système via des terminaux

  Le système d'exploitation entrelace alors les exécutions des différents programmes utilisateurs
  - Chaque programme utilisateur est exécuté pendant un quantum

- Différences entre batch multiprogrammés et temps partagés

  Les systèmes batch multiprogrammé et les systèmes *time sharing* utilisent tous 2 la multiprogrammation. Les principales **différences** entre les 2 sont indiquées dans cette table :

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

- Concept de processus
  - Définition

    Plusieurs définitions sont souvent utilisées pour désigner ce qu'est un processus :
    - Programme en exécution (*en réalité une représentation de l'état d'une instance d'un programme en exécution*)
    - Esprit animé d'un programme
    - L'entité à laquelle le processeur est alloué

    Un processus est exécuté *séquentiellement*, au plus une instruction du processus est exécutée à la fois. Tout processus a une durée de vie, pendant laquelle il passe par différents états.

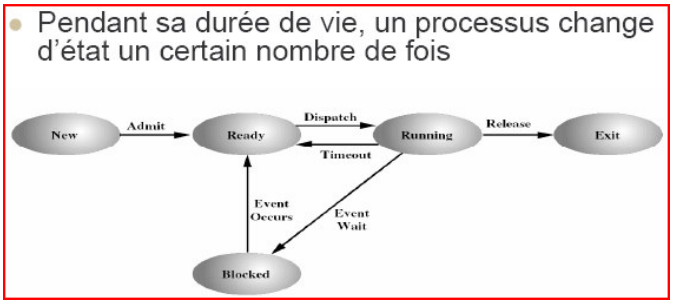
  - Etat d'un processus

    L'état d'un processus est défini en partie par son *activité courante*, un processus peut être dans un des états suivants :
    - New : le processus est en train d'être créé. Un programme est admis dans le système mais n'est pas encore prêt pour l'exécution.
    - Ready : le processus est prêt à être exécuter et attend d'avoir le contrôle du **CPU**.
    - Running : le processus est en train d'être exécuté.
    - Waiting/blocked : le processus est suspendu et mis en attente d'une ressource telle que l'**I/O**.

- Halted/Terminated/Exit : le processus a terminé et attend d'être détruit par le système d'exploitation.

  Note : *A tout instant, un seul processus peut être dans l'état Running. Mais, plusieurs processus peuvent être dans les états Ready, Waiting.*

- Concept de processus



- Progression des processus à l'intérieur de l'ordinateur

  Transparents 58-59.

- Le scheduling
  - Introduction
    - Lorsqu'un ordinateur utilise la multiprogrammation, plusieurs jobs ou programmes rentrent en compétition pour leur admission dans le système pour l'exécution et pour l'utilisation du **CPU**. Il faut alors décider quel est le prochain job ou programme à admettre ou à allouer le **CPU**.
    - Un composant du système d'exploitation en charge d'une telle décision est appelé **scheduler** dont la fonction est d'établir un *scheduling*.
  - Définition

    Le scheduling est l'élément clé de la multiprogrammation.

    Quatres types de scheduling sont souvent utilisés pour supporter la *multiprogrammation (voir la suite)*.
  - Le long-term scheduling

Dans les systèmes (batch) multiprogrammés, il arrive souvent que le nombre de jobs soumis soit plus grand que le nombre de jobs que l'ordinateur peut exécuter immédiatement

Ces jobs sont alors placés dans un pool (long-term queue), typiquement sur le disque dur

Le long-term sheduler détermine quels sont les programmes du pool qui doivent être (admis dans le système ou) chargés en mémoire pour exécution

- Il contrôle alors le degré de multiprogrammation
  - Le nombre de processus en mémoire principale

Une fois qu'un job ou programme est admis dans le système, il devient un processus et est ajouté à la file pour le short-term scheduler

- Le medium-term scheduling

  Le medium-term scheduling fait partie du mécanisme de swapping

  Lorsque tous les processus actuellement en mémoire sont dans l'état blocked
  - Le swapping-out consiste à ramener un de ces processus sur disque pour faire de la place pour un autre processus susceptible de prendre immédiatement le contrôle du CPU
  - Le swapping-in consiste à ramener le nouveau processus en mémoire

  Le medium-term scheduler est donc impliqué dans la décision de quel processus suspendre et quel autre ramener en mémoire (ça peut être la réactivation d'un processus qui a été swapped-out ou un processus du long-term queue)

- Le short-term scheduling

  Le short-term scheduling consiste en l'allocation du CPU

  Le short-term scheduler, détermine quel est le *prochain* processus, dans l'état Ready, qui sera exécuté par le CPU, et pour combien de temps

  La caractéristique principale du short-term scheduler est qu'il s'exécute plus fréquemment que le long-term scheduler

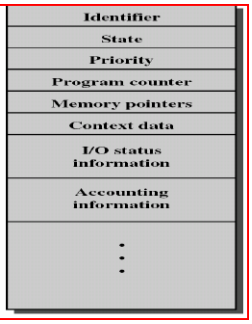
- Le I/O scheduling
- Techniques de scheduling

  Transparents 53->55.
- Eléments clés pour la multiprogrammation et le scheduling.

  Transparents 56-57

- Le Process Control Bloc

A l'intérieur du système d'exploitation, chaque processus est représenté par un bloc d'informations appelé process control bloc
- Identifier: Identificateur unique du processus
- State: New, Ready, Running, Waiting, ou Halted
- Priority: priorité relative du processus
- Program counter: contient l'adresse de la prochaine instruction à exécuter pour ce processus



- Memory pointers: le début et la fin en mémoire du programme de ce processus
- Context data: contenu de certains registres du CPU, pendant l'exécution de ce processus
- I/O status information: liste d'unités périphériques allouées à ce processus, liste de fichiers ouverts pour ce processus
- Accounting information: temps CPU alloué et utilisé par ce processus, numéros de comptes

A l'intérieur du système d'exploitation, chaque processus est représenté par un bloc d'informations appelé process control bloc

Lorsque le scheduler accepte qu'un job soit admis dans le système pour exécution
- Il crée un nouveau processus
- Un PCB vide est créé pour ce processus avec New comme état
- Dès que ce PCB est proprement initialisé, l'état de ce processus passe à Ready

- Gestion de la mémoire

  La multiprogrammation permet d'avoir simultanément en mémoire le noyau du système d'exploitation et plusieurs programmes utilisateurs. La partition de la mémoire dédiée aux programmes utilisateurs doit être subdivisée dynamiquement par le système d'exploitation. La partie du système d'exploitation qui s'occupe de cette tâche est appelée **gestionnaire de mémoire**, son but est de gérer efficacement la mémoire de façon à permettre un maximum de processus en mémoire.

  Transparents 61->72

- Mémoire virtuelle
  - Idée de base

    La pagination rend la multiprogrammation effective, l'idée de partitionner un processus en pages a conduit au développement du concept de ***mémoire virtuelle***.
  - Avantages
    - Plus de programmes (*morceaux de programmes*) en mémoire en même temps.
    - Economie de temps (*Les pages non-utilisées ne sont pas amenées et déchargées de la mémoire.*)
    - Il est possible pour un processus d'être plus large que toute la mémoire principale réelle.

    Mais l'**OS** doit alors être plus « *intelligent* » pour gérer tout cela, quand l'**OS** doit charger une page demandée, il faut en retirer une autre. ➔ **Nécessité d'algorithmes de remplacement de page pour éviter le problème de *trashing.***

15. Arithmétiques des ordinateurs.

Voir Transparents.