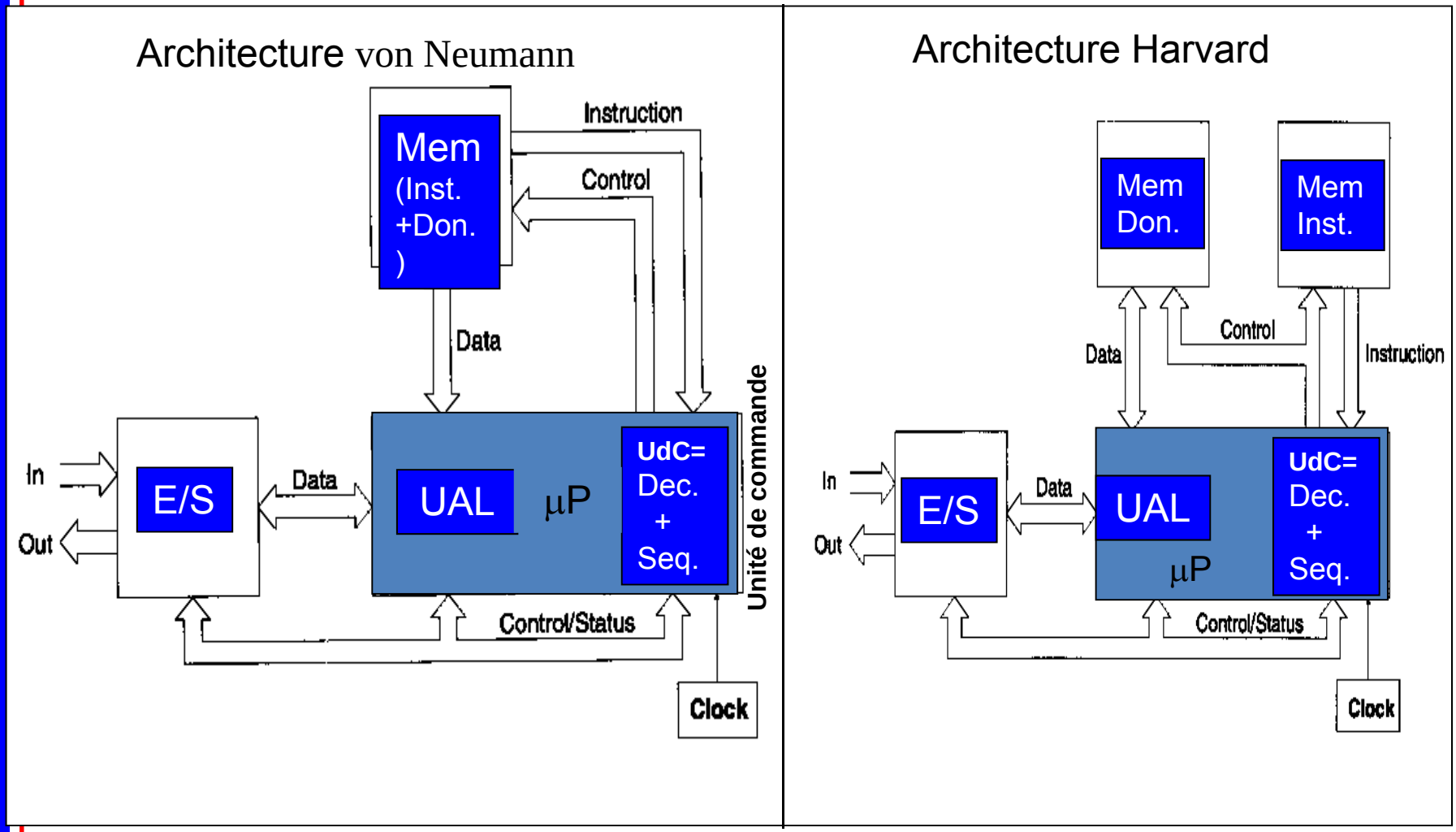
Révision : Structure d'un ordinateur

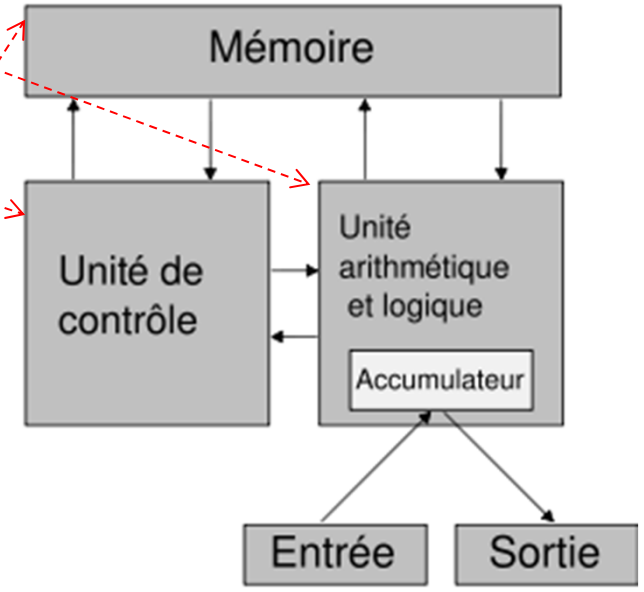
Architecture d'un (Micro)Ordinateur



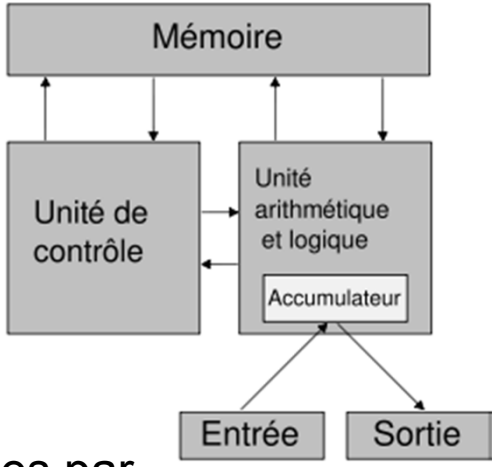
La machine de Neumann (1946)

- Cette machine est totalement différente de la précédente.
- L'architecture, dite **architecture de von Neumann**, est un modèle pour un ordinateur qui utilise une **structure de stockage unique** pour conserver à la fois **les instructions** et **les données requises ou générées par le calcul** (*ordinateurs à programme stocké en mémoire)*.
  - ***La séparation entre le stockage et le processeur est implicite dans ce modèle.***

- L'**architecture de von Neumann** décompose l'ordinateur en 4 parties distinctes :
  1. L'**unité arithmétique et logique** (UAL) ou unité de traitement : son rôle est d'effectuer les opérations de base ;
  2. **L'unité de contrôle**, chargée du séquençage des opérations ;
  3. **La mémoire** qui contient à la fois les **données** et le **programme** qui dira à l'unité de contrôle quels calculs faire sur ces données. La mémoire se divise entre **mémoire volatile (**programmes et données en cours de fonctionnement) et **mémoire permanente** (programmes et données de base de la machine).
  4. **Les dispositifs d'entrée-sortie**, qui permettent de communiquer avec le monde extérieur.



- **Ces dispositifs permettent la mise en œuvre des fonctions de base d'un ordinateur :** le stockage de données, le traitement des données, le mouvement des données et le contrôle.
- **Le fonctionnement schématique en est le suivant :**

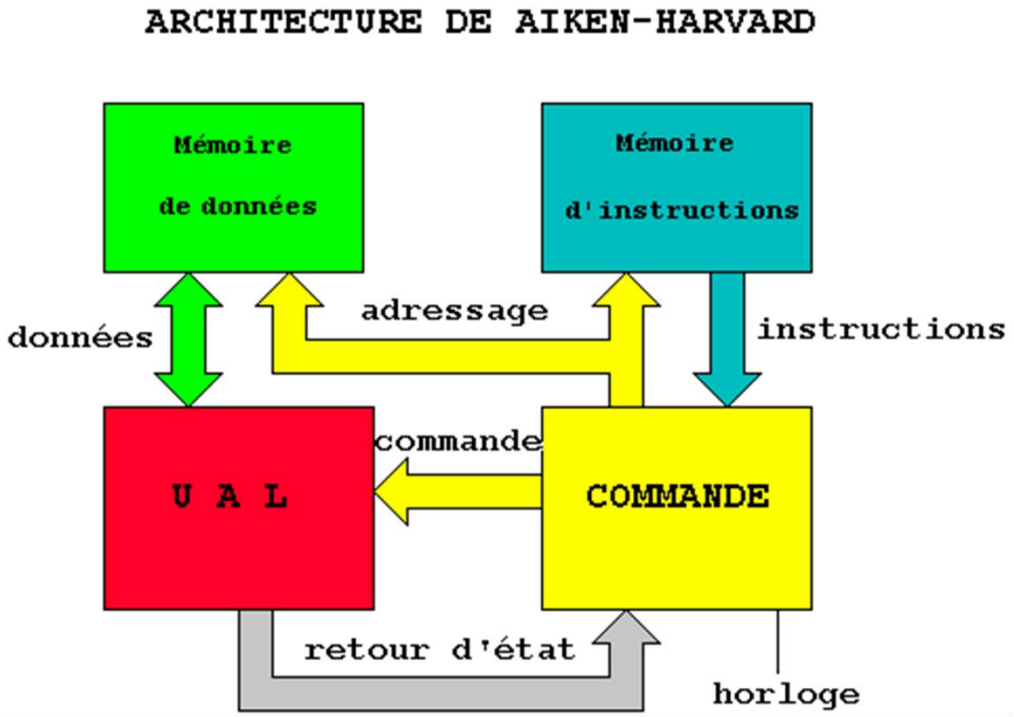


1. l'UC extrait une instruction de la mémoire,
2. analyse l'instruction,
3. Recherche dans la mémoire les données concernées par l'instruction,
4. Déclenche l'opération adéquate sur l'ALU ou l'E/S,
5. Range au besoin le résultat dans la mémoire.

La plupart des machines actuelles s'appuient sur le modèle Von Neumann.

Architecture de Aiken-Harvard :

- Proposée en 1947, elle consiste à donner des capacités **d'accès simultanées** aux **mémoire de programme** et **de données**.
- Elle est longtemps restée dans les livres en raison du **coût des mémoires**.
- Les réalisations sont apparues dans les années 90 avec des **caches séparés** sous le nom de «super Harvard architecture» **(SHARC).**



Qu'est-ce qu'un système informatique ?

- **Un système informatique est un ensemble composé d'un ou plusieurs dispositif programmable.**

- **Chacun de ces dispositifs peut communiquer avec les autres via un mécanisme dit de réseau et chacun de ces dispositifs effectue une ou plusieurs tâches par le déroulement d'instructions constituants le programme qu'il doit exécuter**

❑ Plusieurs types de systèmes informatiques :

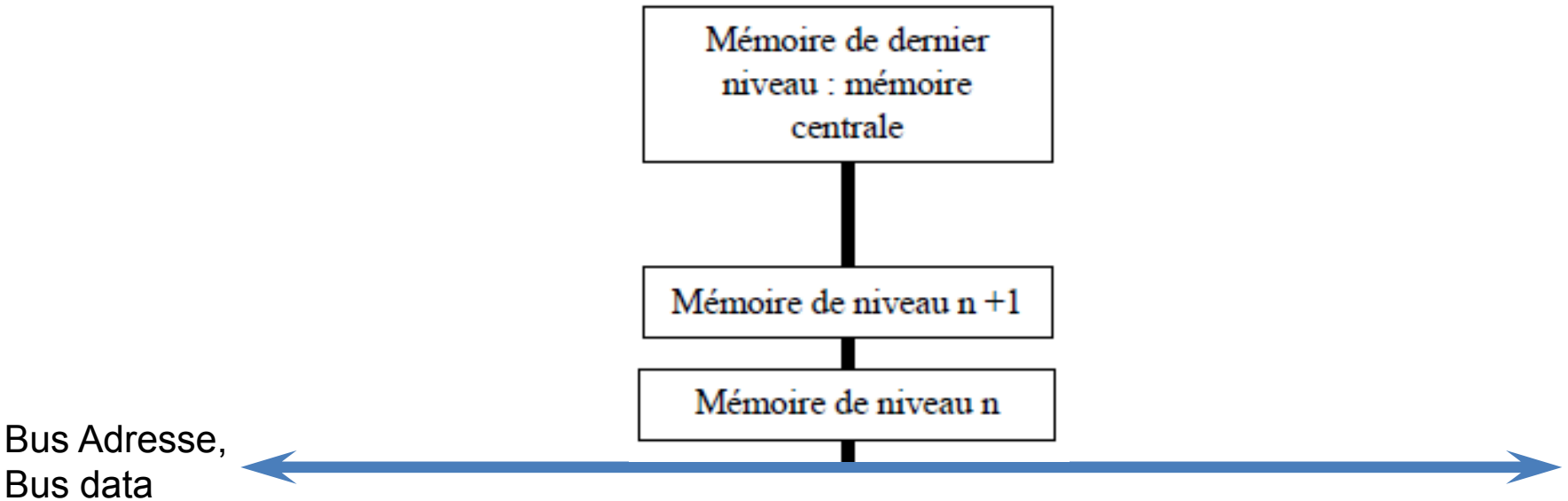
1. **Les systèmes informatiques généraux**
   - Ce sont les machines habituelles : PC, Station Unix, Mini-ordinateurs, MainFrame.
   - Ces systèmes sont adaptables par le logiciel qu'ils exécutent et donc leurs tâches peuvent être variées.
2. **Les systèmes informatiques dédiés**
   - Ce sont des machines dédiées à un type de tâches particulières, par exemple les consoles de jeux vidéo, les machines de tri du courrier.
3. **Les systèmes informatiques embarqués ou enfouis**
   - Ce sont des systèmes informatiques mis dans d'autres machines, par exemple les appareils photo numériques, les téléphones, l'informatique des véhicules de transport

- **Les bus:**
  - C'est le cœur d'un ordinateur car il véhicule les informations d'un composant à l'autre.
  - Sa composition :
    - Un ensemble de fil.
    - Un protocole de communication.
    - Un ensemble de règle régie la communication par un composant : le maître du bus.

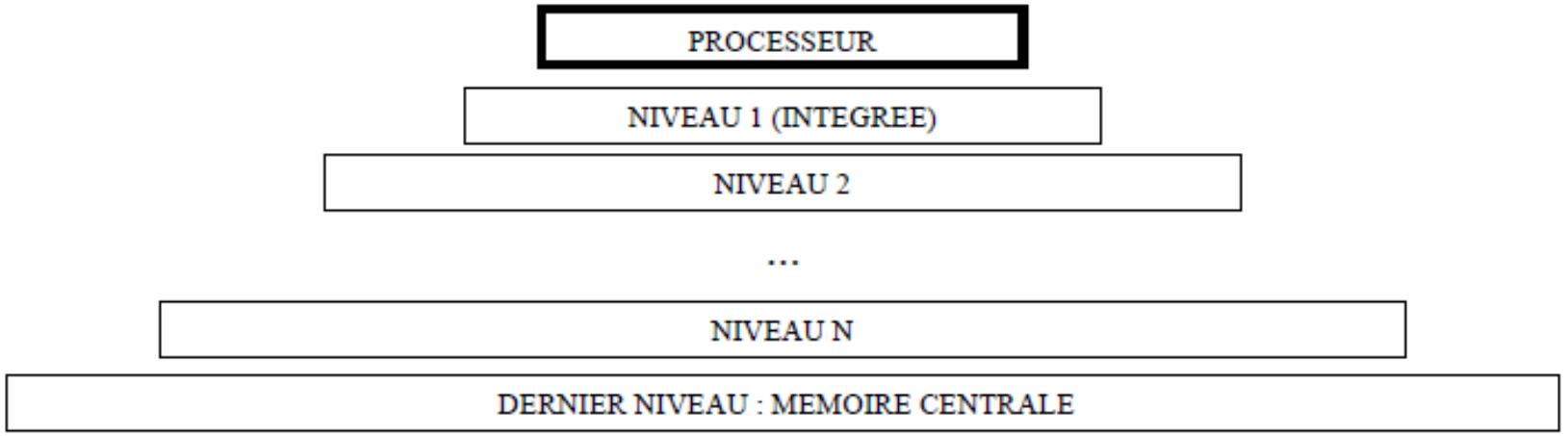
Bus Adresse,
Bus data

Les mémoires

- Elles permettent de stocker de l'information et sont réparties en plusieurs niveaux, on parle de mémoire cache :



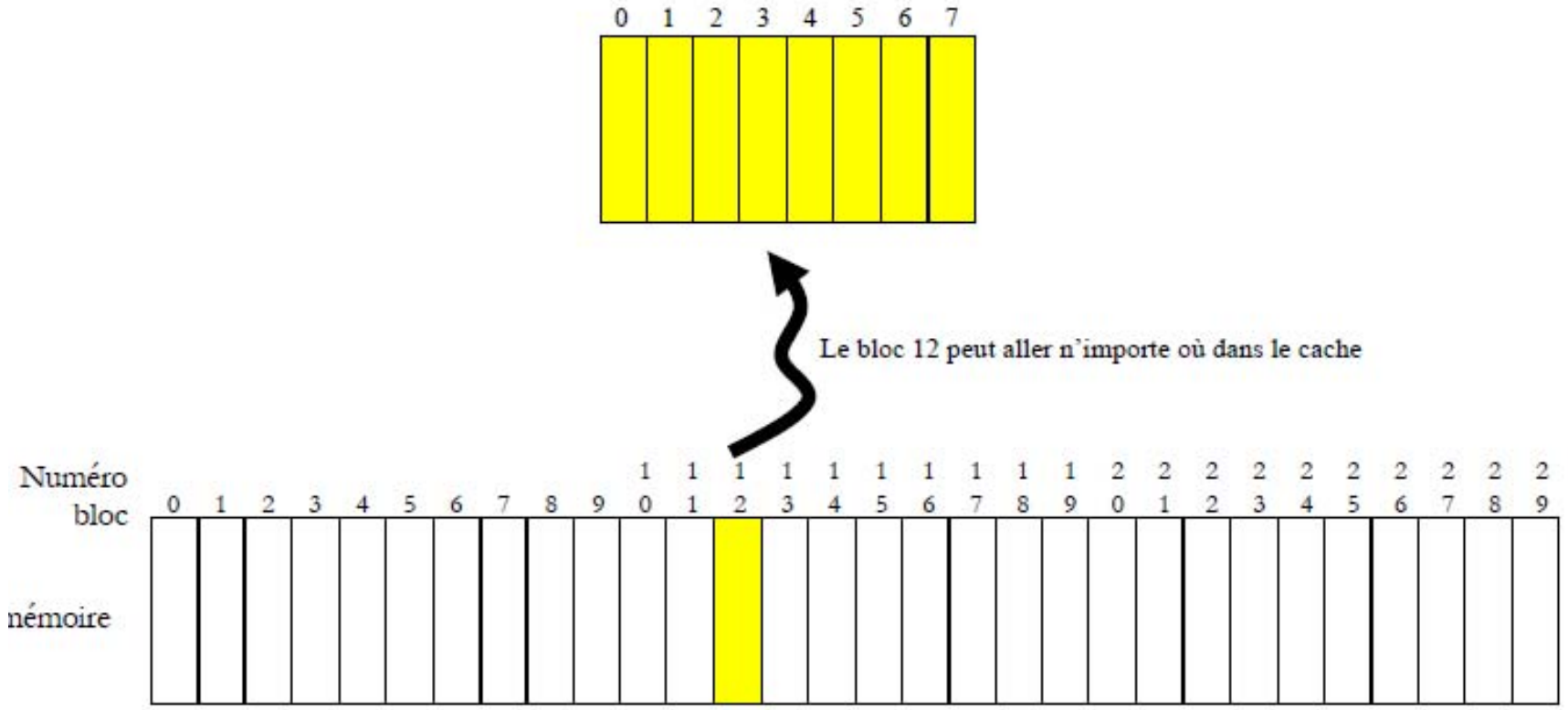
La hiérarchie de mémoire



- Le niveau n est plus petit (en taille) que le niveau (n -1).
- Une donnée présente dans le niveau (n-1) est présente dans le niveau n.
- Chaque niveau n fait correspondre une grande quantité d'adresses (en nombre) vers une quantité plus petite situé au niveau (n-1).

Ou mettre le bloc

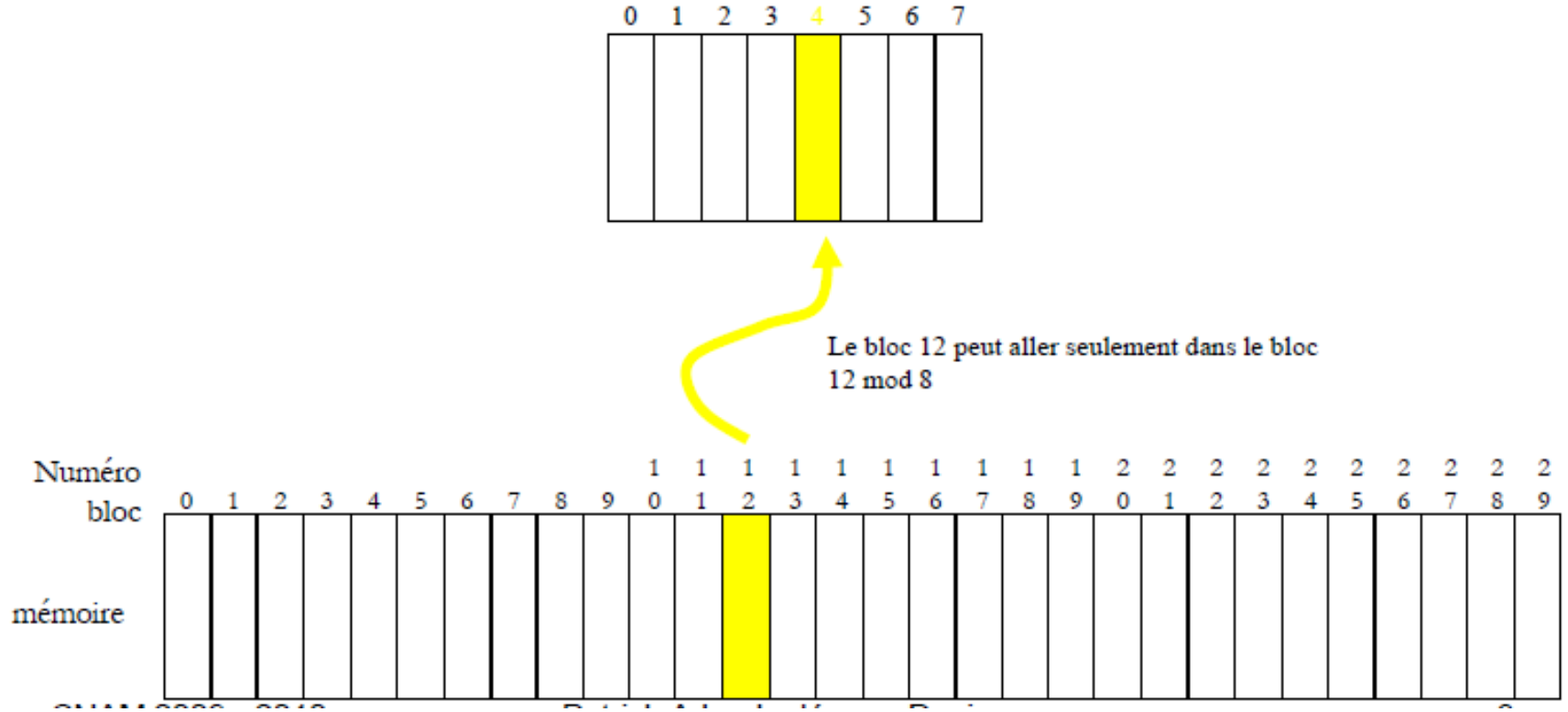
- **Cache totalement associatif** : Le bloc mémoire peut aller n'importe où dans le cache :



Ou mettre le bloc

- **Cache à correspondance direct** :Le bloc mémoire a une seule place disponible dans le cache :

(numéro bloc) mod (nombre de bloc dans le cache)

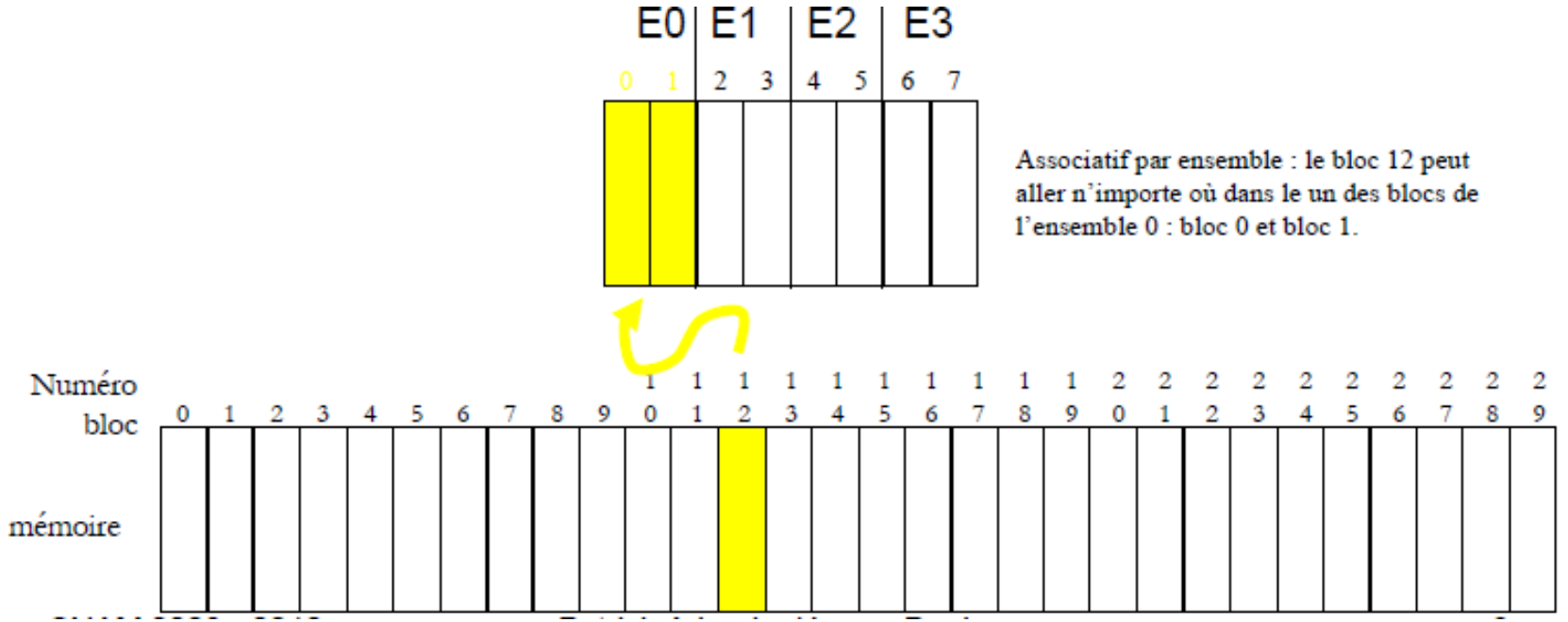


Ou mettre le bloc

- **Cache associatif par ensemble de blocs** : Le bloc mémoire peut être placé dans un ensemble de blocs :

(numéro bloc) mod (nombre d'ensemble dans le cache)

Un ensemble est un groupe de blocs, s'il y a n blocs dans un ensemble, la correspondance est appelée .**associative par ensemble de n blocs**

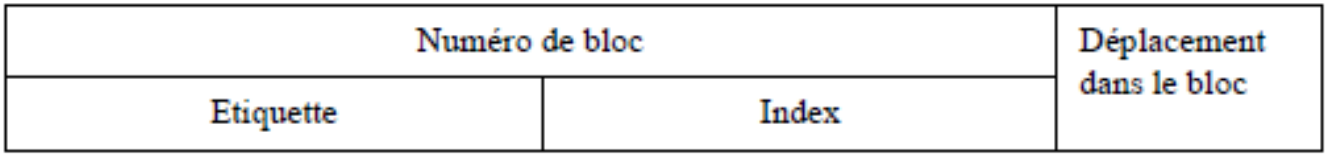


❑ Comment trouver le bloc :

- Pour chaque bloc des caches, il existe une étiquette qui donne le numéro du bloc présent :

  ⇒ Cette étiquette est examinée pour voir si elle correspond au numéro de bloc provenant de l'UC.

- Pour chaque bloc des caches, il existe un bit qui indique la validité du bloc. Si le bit n'est pas positionné, il ne peut y avoir d'accès à une adresse du bloc (non représen



Relation entre l'adresse et le cache

L'adresse est séparée en deux : le numéro de bloc et le déplacement dans ce bloc :

➢ Le numéro de bloc peut être divisé en un champ étiquette et un champ index.

L'augmentation du nombre d'ensemble diminue le champ étiquette et augmente le champ index.

❑ Remplacement d'un bloc en cas de défaut de cache

- **Correspondance directe :**
  - ➢ Pas le choix, seul le bloc testé est remplacé.
- **Associativité totale ou par ensemble :**
  - ➢ Le hasard : génération d'un numéro de bloc aléatoire.
  - ➢ Le plus ancien : LRU, le bloc remplacé est celui qui n'a pas été utilisé depuis le plus longtemps.
- **Le hasard est le plus facile à implémenter par matériel. Le coût du LRU croit avec le nombre de blocs à observer.**

Les écritures

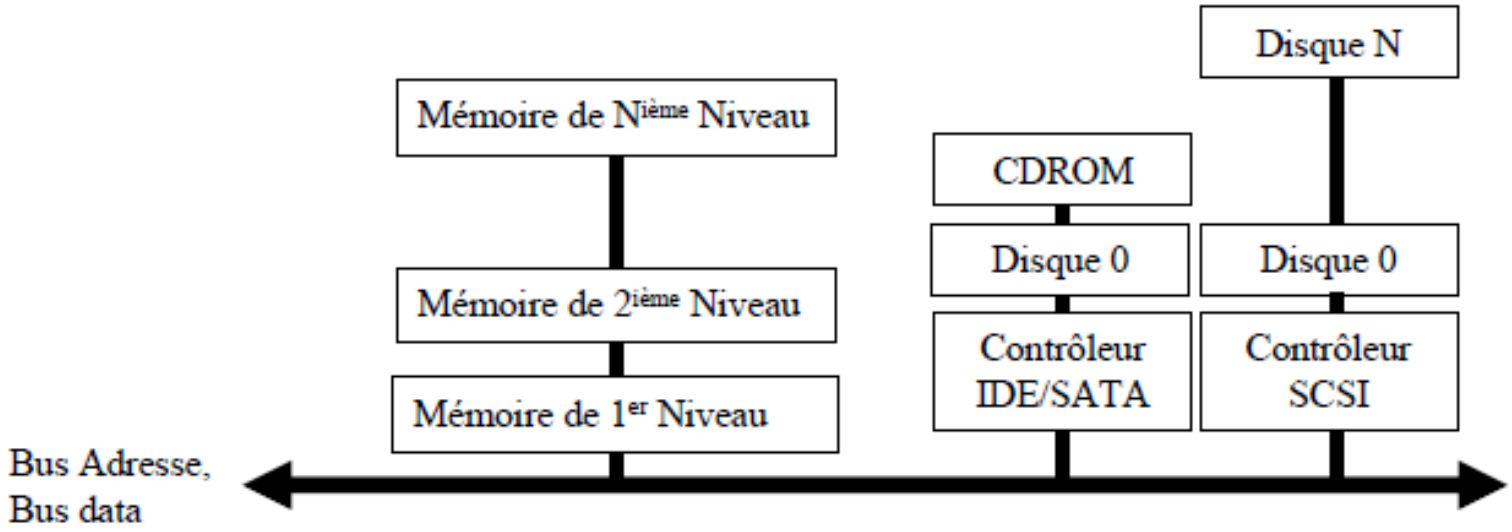
- **Les lectures dominent les accès au cache :**
  - Les accès aux instructions et à leurs opérandes s'effectuent en lecture.
  - 9% des instructions font des rangements en mémoire.

- **Deux politiques d'écriture existent :**
  1. L'écriture ou rangement simultané dans lequel l'information est écrite dans le cache et dans le niveau inférieur.

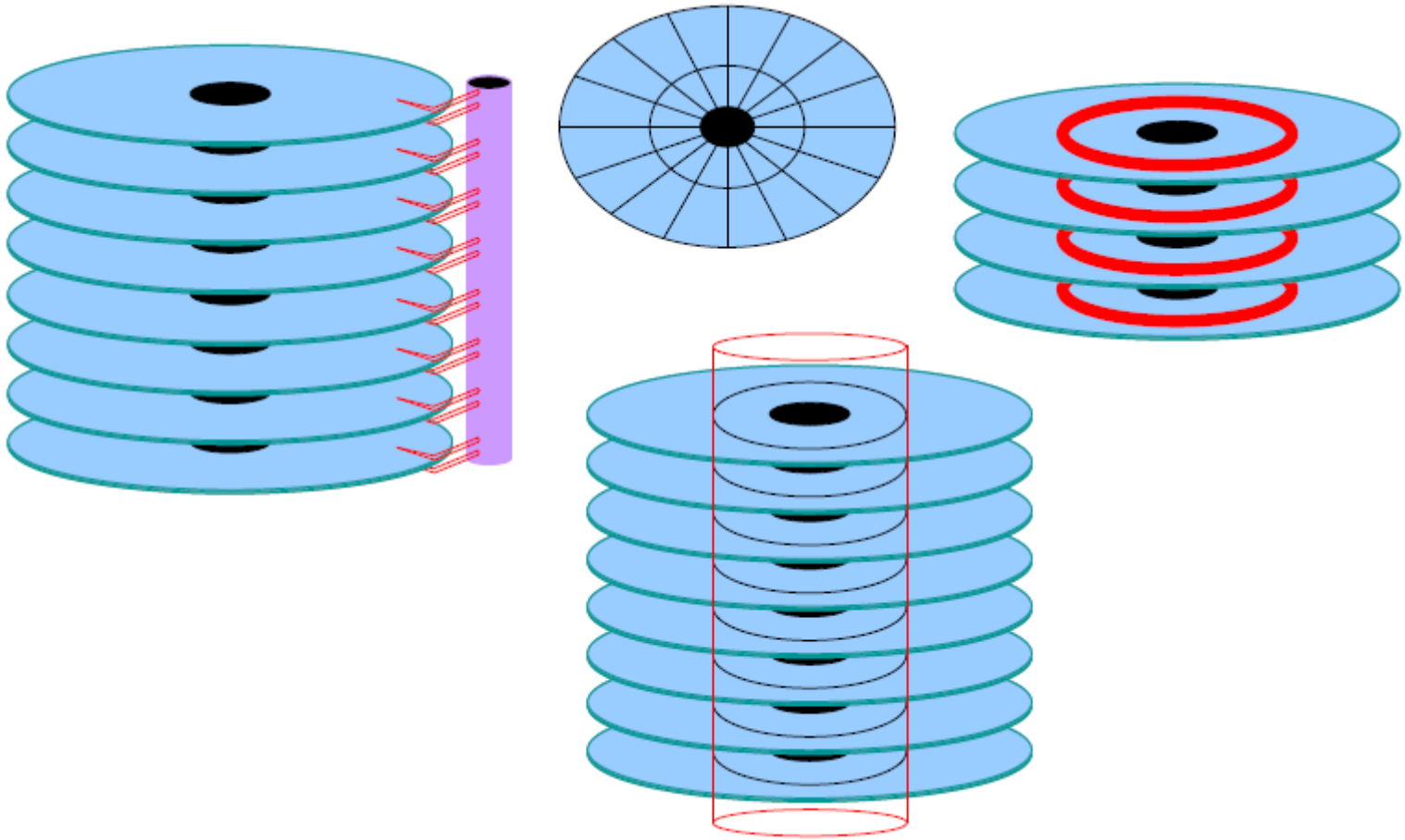
  2. La réécriture ou recopie ou encore rangement dans lequel l'information est écrite uniquement dans le bloc du cache. Le cache modifié est recopié en mémoire principale uniquement lorsqu'il est remplacé.

Les disques magnétiques

- Deux types de disques cohabitent aujourd'hui : les IDE, SATA et les SCSI.



❑ Les disques magnétiques



Le DMA

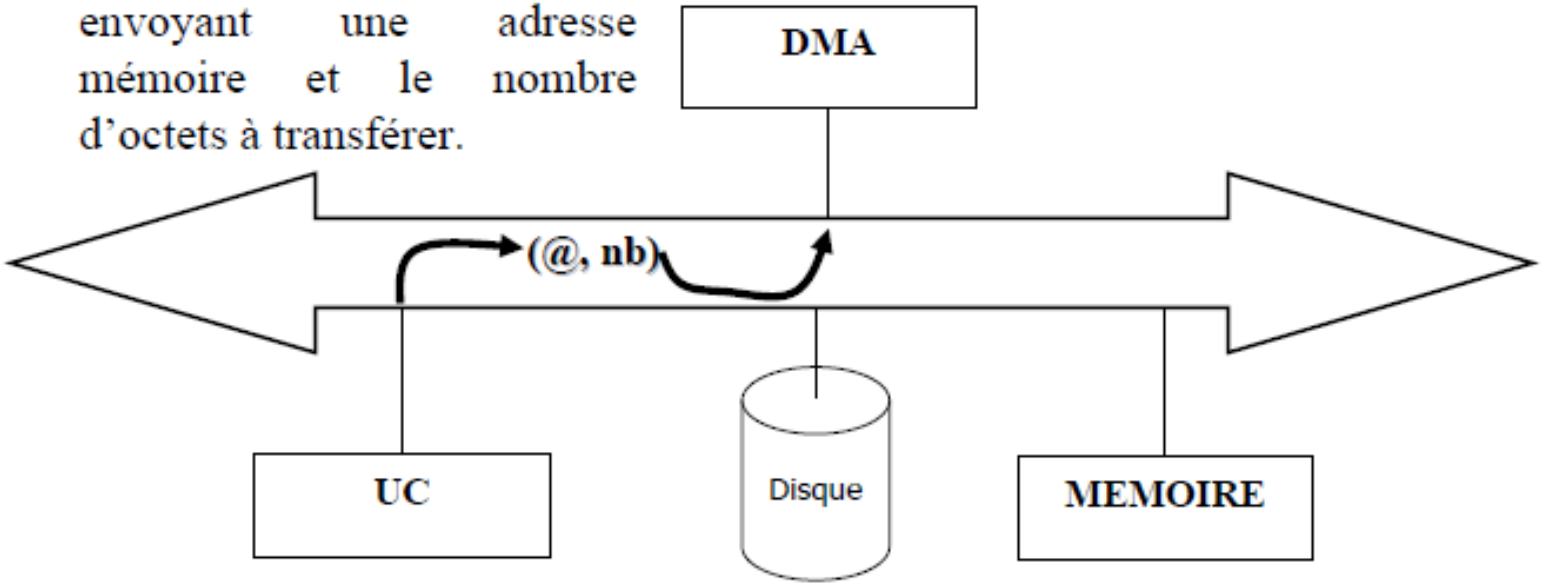
- Les entrées –sorties gérées par interruptions libèrent le processeur, mais :
  - Le processeur passe beaucoup de temps à effectuer des instructions de transfert de données.
- Beaucoup d'événements d'entrées –sorties nécessitent des transferts de blocs
- **La plupart des ordinateurs possèdent un processeur spécialisé qui gère le dispositif d'accès à la mémoire** :

LE DMA

- (Direct Memory Access : accès direct à la mémoire).

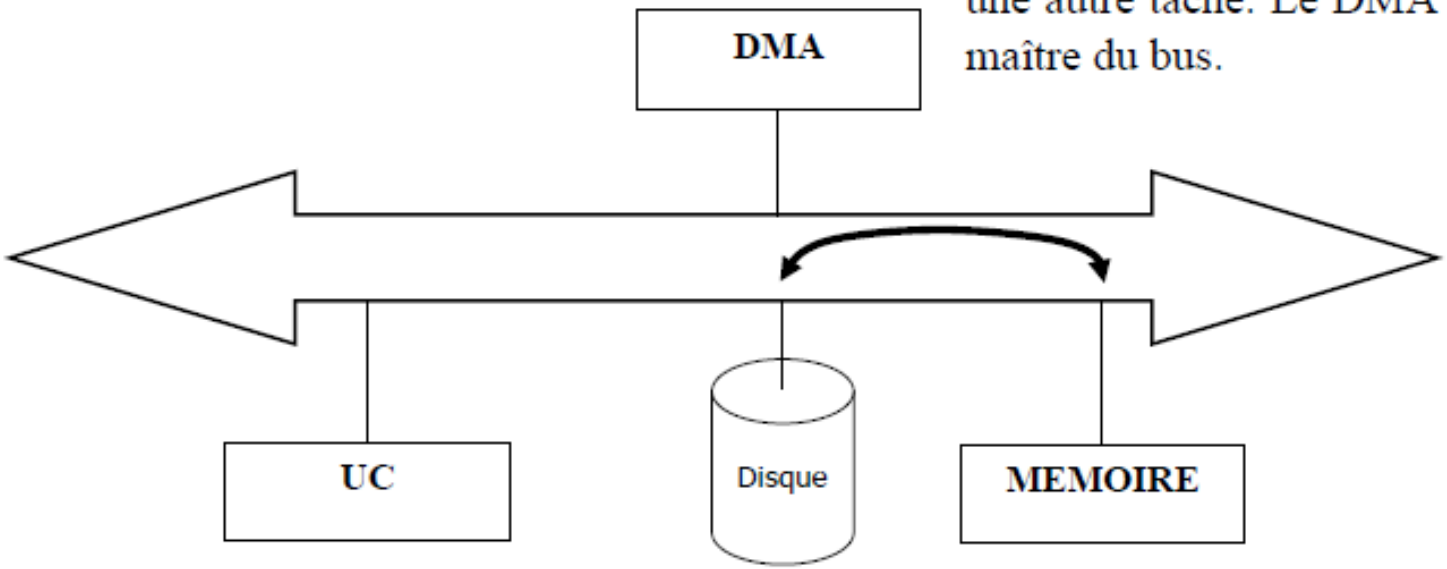
Le DMA

❶ Le processeur initialise les registres du DMA en lui envoyant une adresse mémoire et le nombre d'octets à transférer.



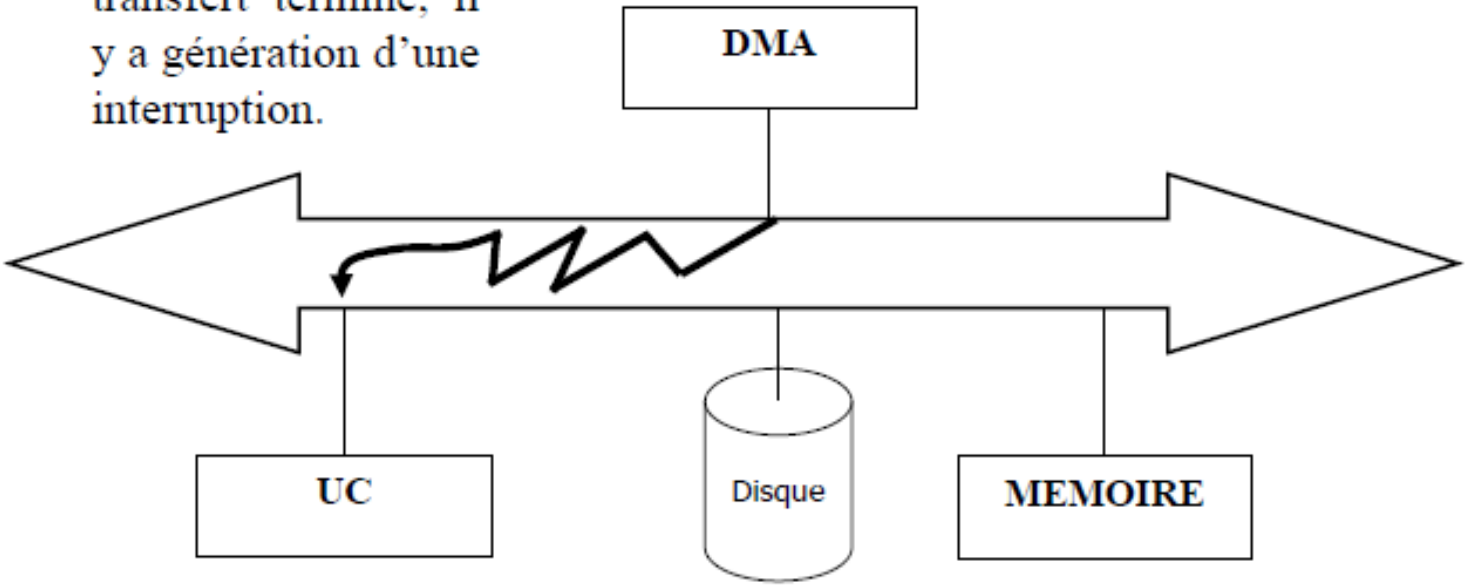
❑ Le DMA

❷ Le DMA transfert les données entre un disque et la mémoire, pendant ce temps, l'UC travaille a une autre tache. Le DMA doit être maître du bus.



Le DMA

❸ Une fois le transfert terminé, il y a génération d'une interruption.



Le DMA :

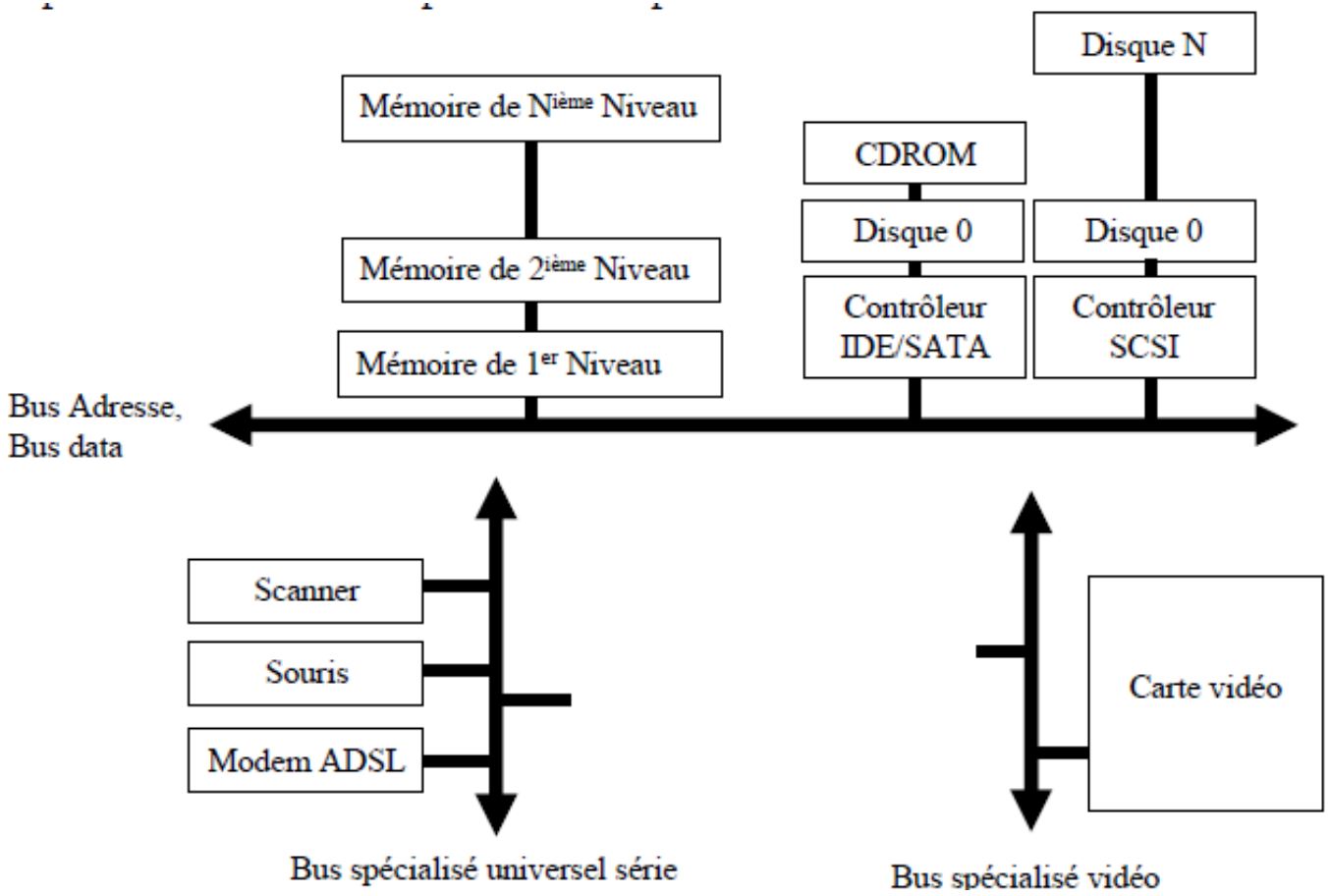
- **Il peut y avoir plusieurs DMA dans un même ordinateur :**
  - Il faut dans ce cas avoir un arbitre pour accéder au bus.

- **Les DMA peuvent être plus ou moins intelligents, ils forment alors des processeurs d'entrées –sorties :**
  - Ces processeurs opèrent à partir de programmes fixes, ou de programmes chargés par le système d'exploitation.
  - Par exemple : l'imprimante LP11 provoqueraient 4800 interruptions pour imprimer une page de 60 lignes de 80 caractères, un processeur d'entrées – sorties pourraient éviter 4799 interruptions.

- **Puisque nous parlons de processeurs d'entrées –sorties capables d'exécuter des instructions, nous sommes presque en train de parler de multiprocesseurs.**
  - Toutefois ces processeurs n'ont pas pour vocation d'effectuer que des transferts de données et non des instructions générales.

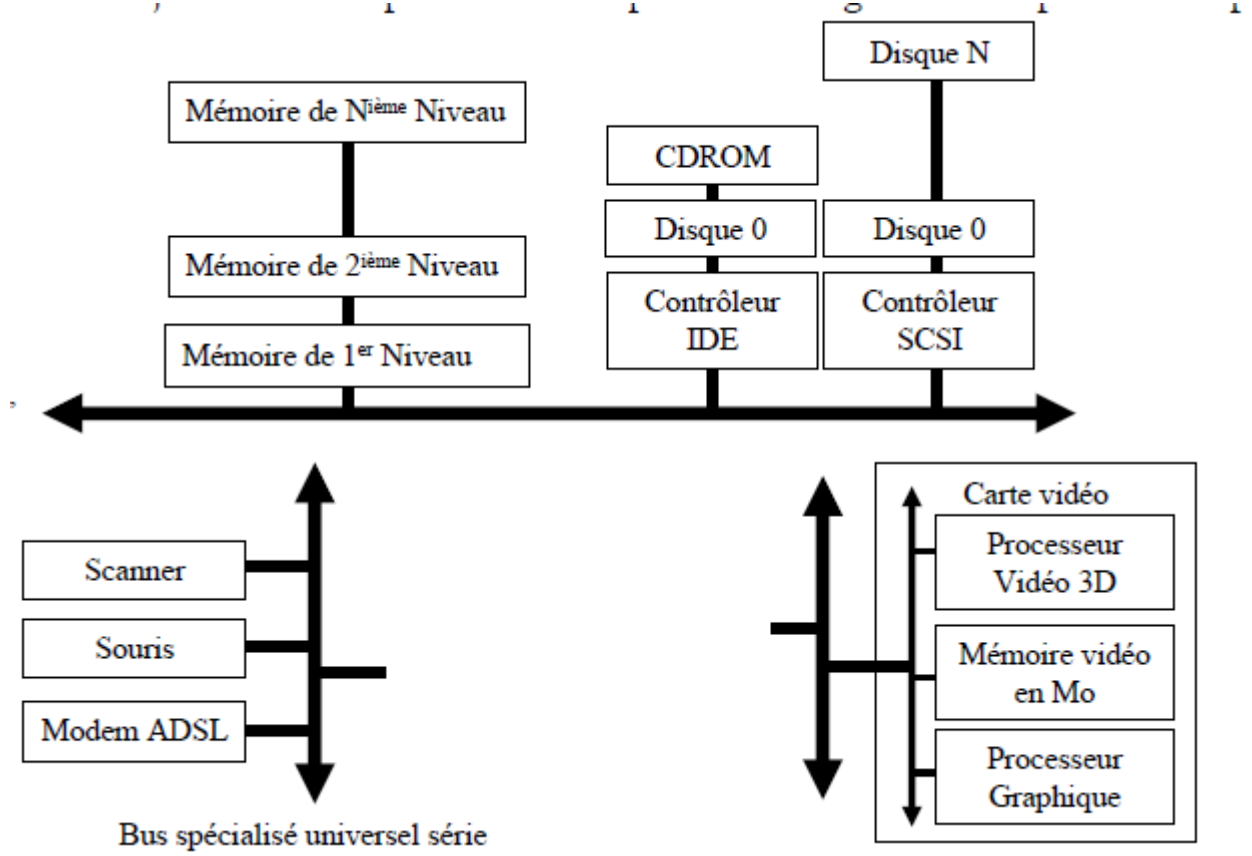
Les bus

- C'est par eux que s'acheminent les informations au travers des machines. Une machine peut en posséder plusieurs dont certains peuvent être spécialisés.

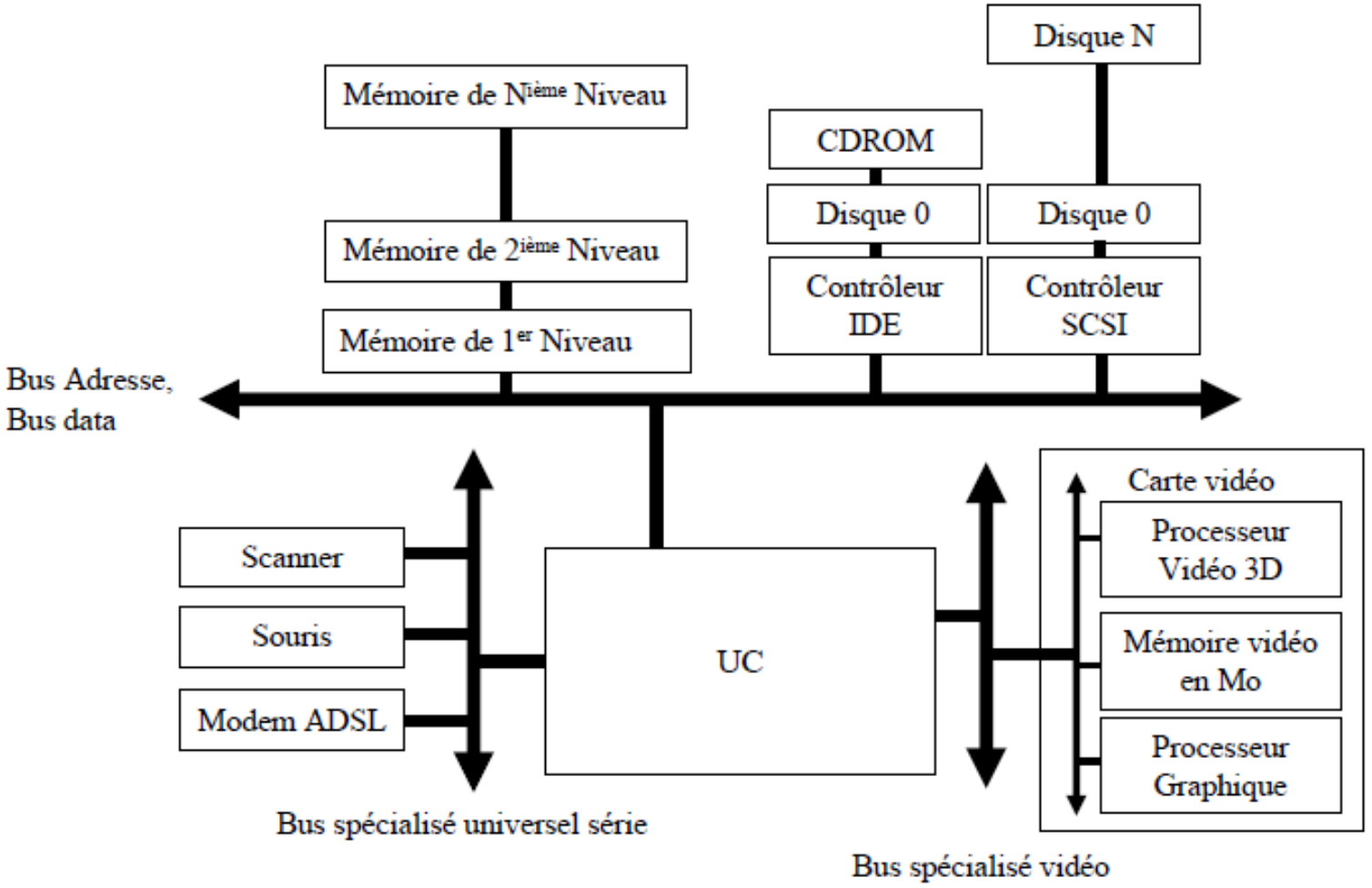


Les cartes vidéo

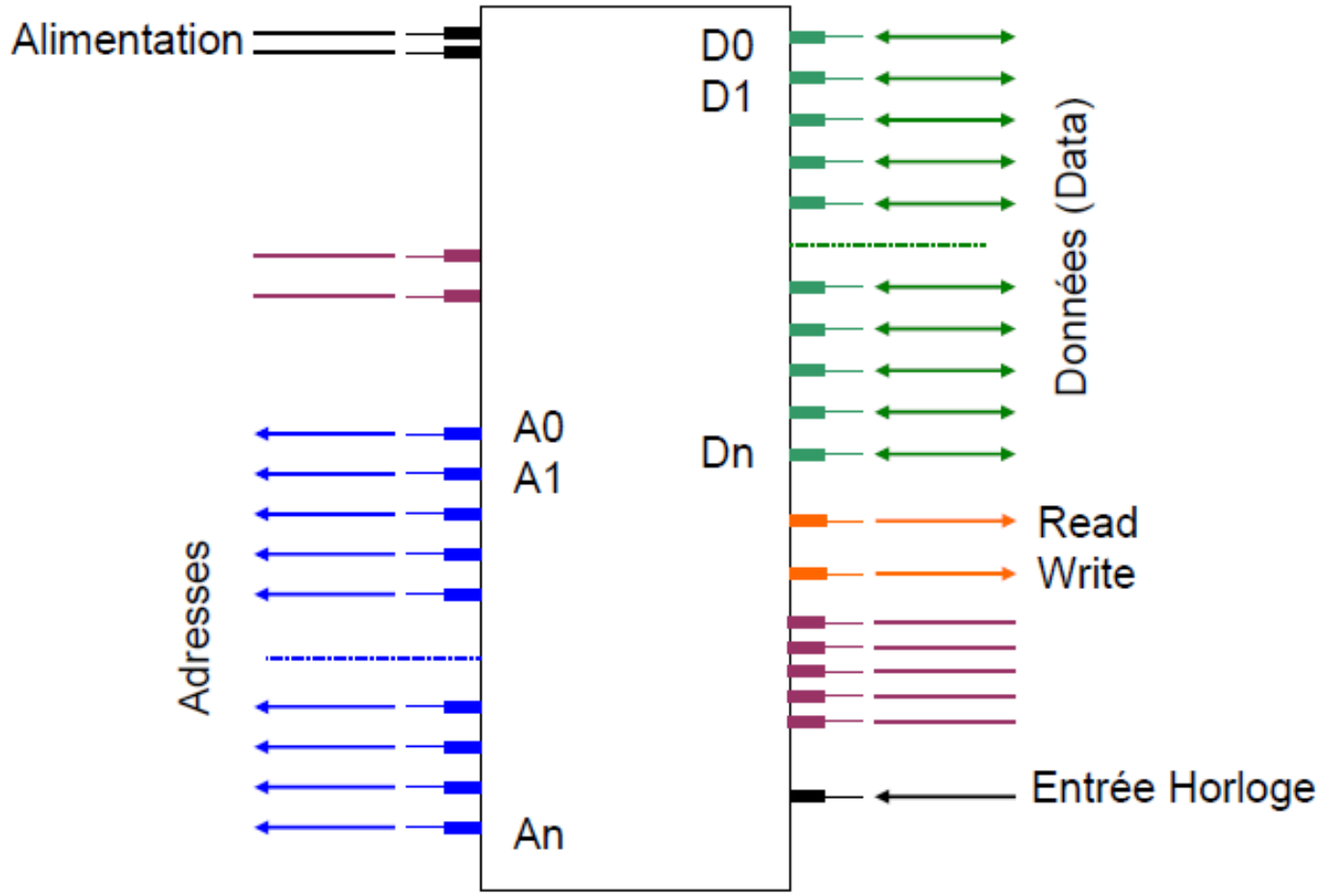
- Les cartes vidéo d'aujourd'hui incorporent des composants intelligents et des processeurs pour la 3D.



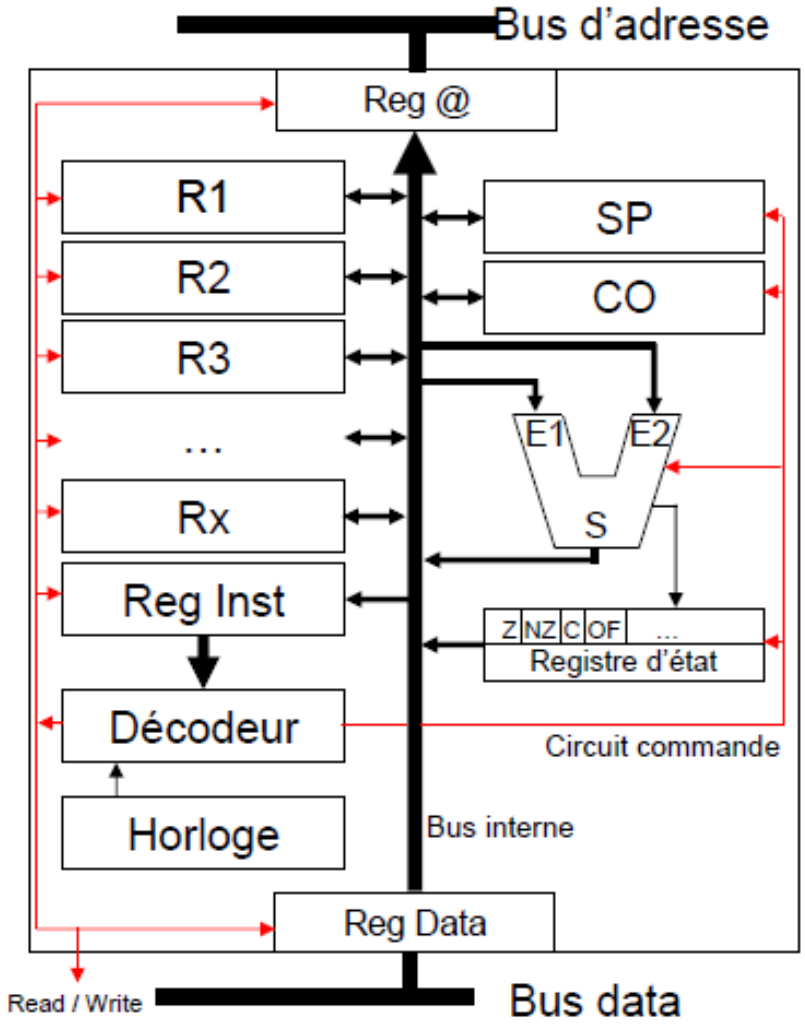
Le processeur



Le processeur

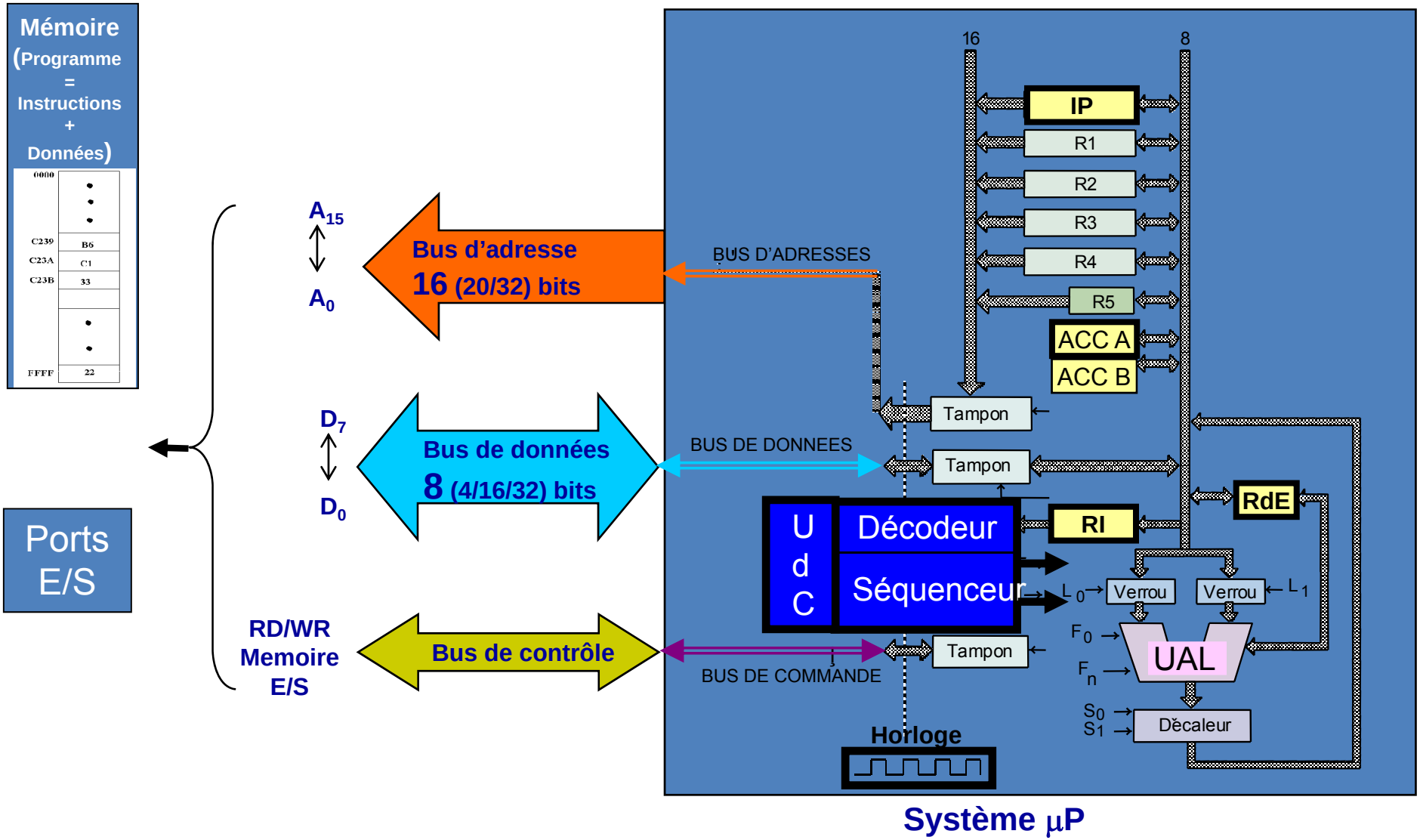


❑ Le processeur



Add R1, R2, R3	R3 ← R1 + R2
01 R1-Lec	BusInt ← R1
02 UalE1-Ecr	UalE1 ← BusInt
03 UalE1-Dec	Déconnexion UalE1
04 R1-Dec	Déconnexion R1
05 R2-Lec	BusInt ← R2
06 UalE2-Ecr	UalE2 ← BusInt
07 UalE2-Dec	Déconnexion UalE2
08 R2-Dec	Déconnexion R2
09 Add	Commande Ual ADD
10 UalS-Lec	BusInt ← UalS
11 R3-Ecr	R3 ← BusInt
12 R3-Dec	Déconnexion R3
13 UalS-Dec	Déconnexion UalS
Fetch	
14 Co-Lec	*BusInt ← CO*
15 Reg @-Ecr	*Reg @ ← BusInt*
16 Reg @-Syn-Lec	*Synchronisation Bus adresse machine - Reg @*
17 Read Mémoire	*Bus data machine ← [Mémoire]*
18 Reg @-Dec	*Déconnexion Reg @*
19 Co-Dec	*Déconnexion Co*
20 Co++	*INC CO (pointe sur instruction suivante)*
21 Reg-Data-syn-Ecr	*Synchronisation Bus Data machine – Reg Data*
22 Reg-Data-Lec	*BusInt ← RegData*
23 RI-Ecr	*RI ← BusInt*
24 RI-Dec	*Déconnexion RI*
25 Reg-Data-Dec	*Déconnexion RegData*
26 Reg-Data-NSyn	*Déconnexion Reg Data – Bus data machine*
27 Reg @-NSyn	*Déconnexion Reg @ - Bus adresse machine*

Architecture minimum d'un micro P



Architecture minimum d'un micro P.

- **Les registres**
- Petites mémoires, à accès parallèle, d'un à plusieurs octets. On trouve trois styles de registres :
- - qui stockent le résultat de l'instruction traitée;
- - qui permettent de retrouver l'adresse de l'information dans la mémoire ;
- - qui enregistrent le code de l'instruction qui vient d'être lue dans la mémoire et ramenée dans le µP.

- **Unité arithmétique et logique (UAL)**
- Elle est chargée d'exécuter les opérations arithmétiques et logiques du programme.
-
- **Unité de Commande**

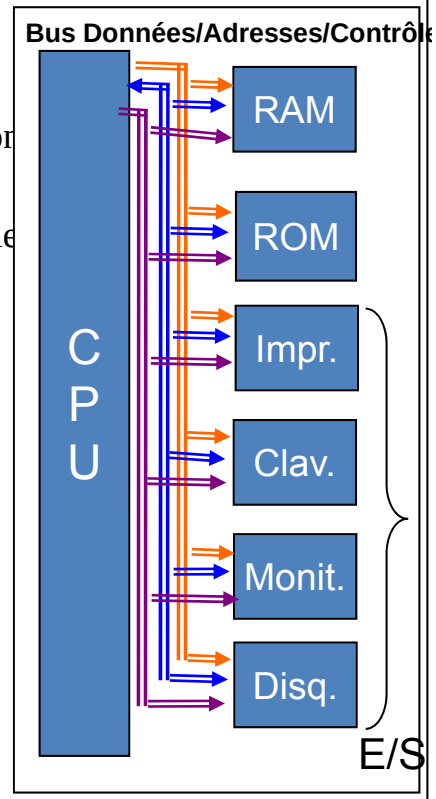
- ***Le* décodeur d'instructions**
- Le décodeur sert à animer les circuits électriques nécessaires à l'exécution de l'instructions lue . En effet, il gère la mise en place des portes logiques pour le bon déroulement de l'opération demandée.

- **Le séquenceur** (Le circuit de commande et de séquencement) :
- Le séquenceur a pour but de mettre en place chaque section du µP en service à tour de rôle.

- **Horloge**
- On peut également trouver le circuit d'une horloge à quartz intégré (ou pas) au µP.

A l'intérieur d'un micro Ordinateur

- **CPU :** Registres, UAL, Unité de Commande
- **Périphéries E/S** : Écran, Imprimante, Clavier, Disque dur/souple, Carte d'acquisition
- **ROM** → Non-volatile, Stockage permanent des programmes et informations essentie
- pour le fonctionnement de l'ordinateur (programmes d'affichage d'informations
- sur écran vidéo, …)
- **RAM** → Volatile, Stockage temporaire des programmes et données
- (logiciels Word, Calcul Taxes, …)



- <u>Quelques Terminologies:</u>
- **Bit** 0 **Octet** (Byte) 0000 0000
- **Quartet** (Nibble) 0000 **Mot (**Word) 0000 0000 0000 0000

- **KiloOctet** **(Ko)** 2^{10} = 1024 octets (et non [1000] mille octets!!!)
- **MegaOctet** **(Mo)** 2^{20} = 1048576 octets (et non [10^6] million octets!!!)
- **GigaOctet** **(Go)** 2^{30} octets (et non [10^9] billion octets!!!)
- **TiraOctet** **(To)** 2^{40} octets (et non [1012] trillion octets!!!)

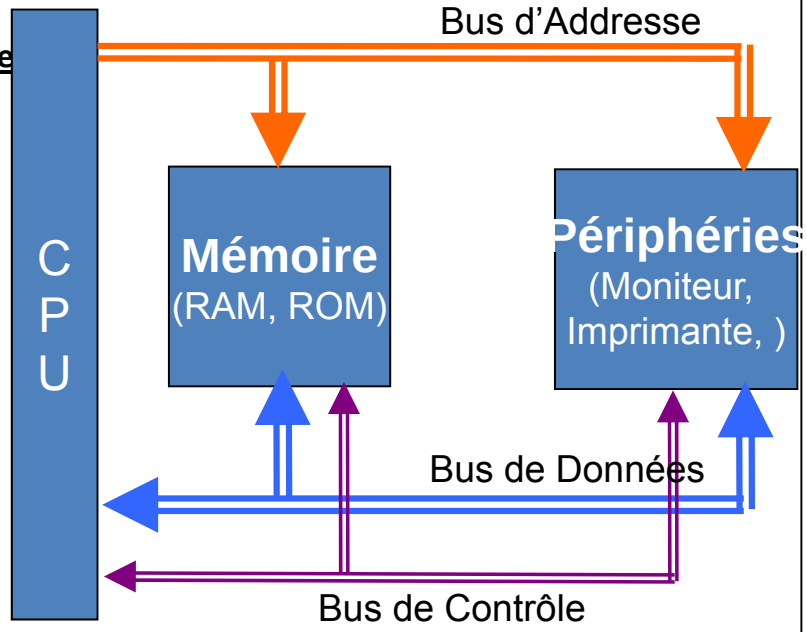
- **<u>Exemple</u>**: Un micro-ordinateur possède 16 Mo de mémoire → 16 x 2^{20} = 2^{24} octets

A l'intérieur d'un µOrdinateur

- Chaque unité (Mémoire, E/S) possède sa propre adresse.
  - La CPU met l'adresse (en binaire) dans le **Bus d'Adresse**
  - Un circuit de décodage repère l'unité Adressée.

- Ensuite la CPU utilise le **Bus de Données** pour envoyer
  - la donnée a l'unité adressée ou acquérir la donnée a
  - partir de cette unité.

- Le **Bus de Contrôle** génère (*entre autres*) des signaux
  - de lecture/écriture envers l'unité adressée pour
  - indiquer la direction de transmission des données:
  - si la CPU lui demande une information
  - ou plutôt lui envoie une information.



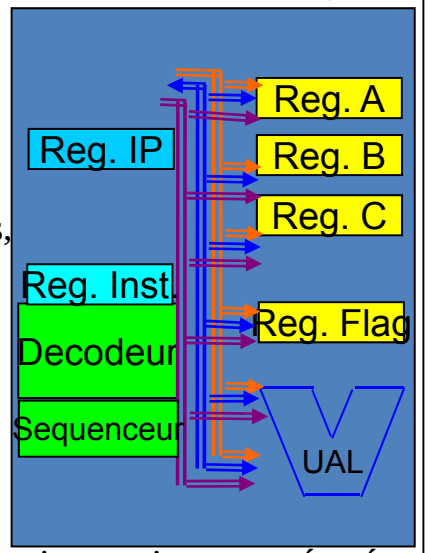
- IMPORTANT:

- **Bus de Données** : Bidirectionnel. Plus la taille est importante, la CPU est meilleure au détriment du coût de construction. La moyenne de la taille varie entre 8 et 64. La puissance du traitement est liée a cette taille. Un bus 8-bit peut envoyer 1 byte, alors qu'un bus 16-bit en envoie 2 bytes, donc deux fois plus rapide.

- **Bus d'Adresse** : Unidirectionnel. Plus la taille est importante, le nombre d'unités (locations mémoire) a adresser est grand au détriment du coût de construction. Le nombre de locations est 2^x, ou **x** est la taille du bus d'adresse. La moyenne de la taille varie entre 16 et 32. Exemple: une CPU a 16 lignes d'adresses, peut cibler une mémoire a 64K locations, ou chaque location possède un maximum de 1 byte (*Byte Adressable CPUs*).

A l'intérieur d'un µP (CPU)

- Un programme stocké dans une mémoire est un ensemble d'instructions pour la CPU pour performer une action donnée (contrôle d'un robot). C'est le rôle de la CPU de chercher ces instructions et les exécuter. Pour y parvenir, la CPU est équipée de:



- **Registres**
- C'est des petites mémoires, à accès parallèle, pour stocker une information temporairement. Celle-ci peut être une valeur a traiter ou l'adresse d'une valeur a chercher d'une mémoire. Ces registres peuvent être 8, 16, 32 ou 64 bits, suivant la CPU. En générale, plus les registres sont nombreux et de grande taille, meilleure est la CPU. Ceci est au détriment du prix qui croit.

- **Unité arithmétique et logique (UAL)**
- Elle est chargée d'exécuter les opérations arithmétiques et logiques du programme.

- **Compteur Programme**
- Pointe sur l'adresse de la prochaine instruction a exécuter. A chaque fois qu'une instruction est exécutée, le compteur programme est incrémenté. C'est son contenu qui est placé dans le bus d'adresse pour trouver et chercher l'instruction désirée. Pour l'IBM PC, ce compteur programme est un registre qui s'intitule IP (pointeur d'instructions).

- **Décodeur d'instructions**
- Son rôle est d'interpréter le code instruction recherché dans la CPU. Il l'analyse et transmet au séquenceur le type d'opération a exécuter.

- **Séquenceur**
- Organise l'exécution de l'instruction en distribuant les µ-commandes aux divers blocs du µP suivant le rythme d'une horloge pilote.

Principe de fonctionnement d'un µP(µOrdinateur)

Unite de commande

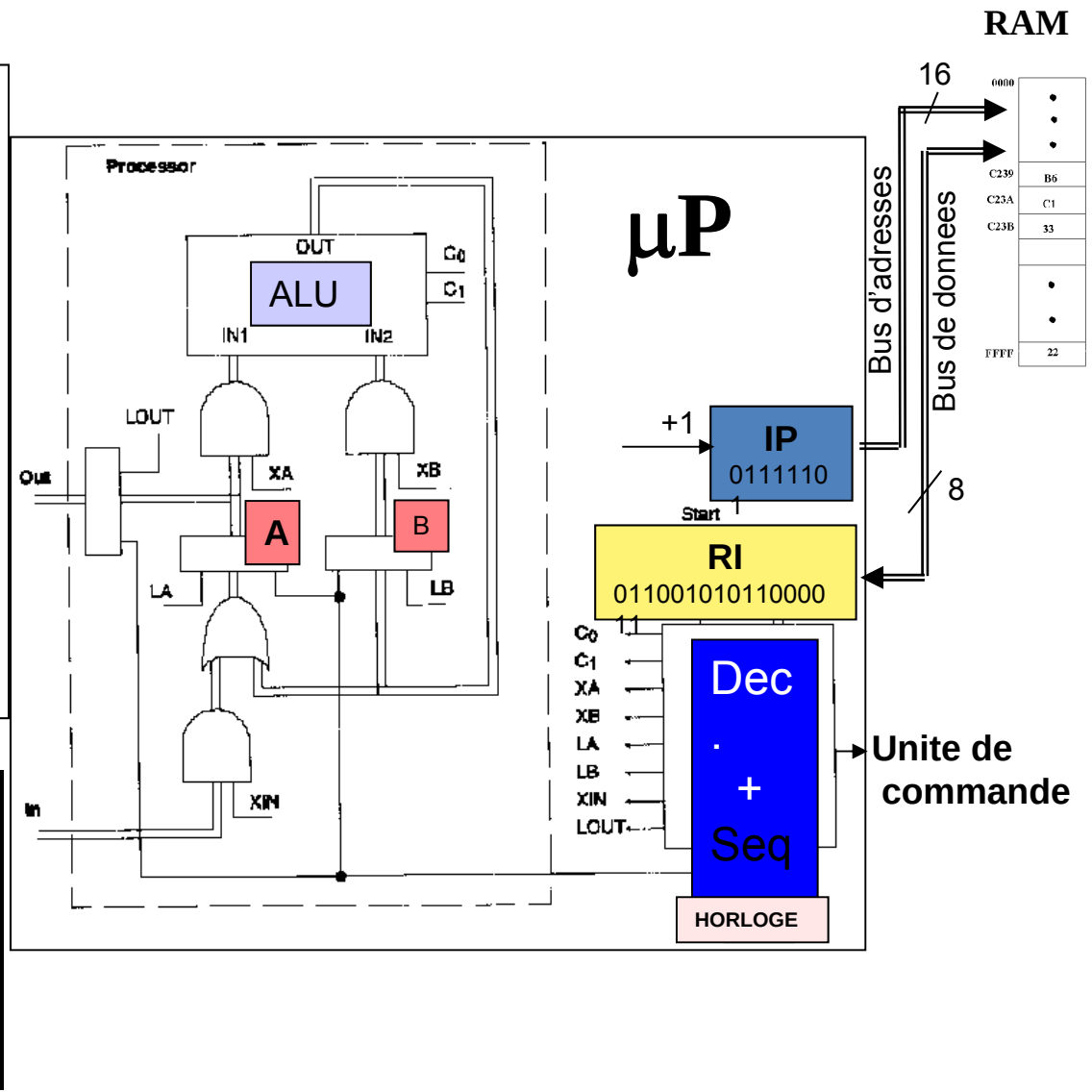
- **Une opération instruction peut prendre plusieurs cycles d'horloge**
- **Ceux-ci peuvent être en 2 phases: fetch et exécuté**

- **FETCH**
  - **Instruction est ramené de RAM vers l'unité de commande via un RI**
  - **l'unité de commande prépare pour la prochaine opération de fetch**

- **EXECUTE**
  - **l'unité de commande décode l'instruction**
  - **Envoie des signaux de commande vers le le processeur**

Operation ALU

Signaux de Commande C0C1	Operation
00	IN1 +IN2 →OUT
01	IN1 +IN2+1 →OUT
10	IN1 +IN2' →OUT
11	IN1 +IN2'+1 →OUT



⌦ **Aspect quantitatif :** La **mesure de capacité d'une unité centrale** peut, entre autres, être caractérisée par **trois facteurs multiplicatifs** :

1. le **nombre d'instructions** à exécuter pour une même tâche, **NI;**
2. le **nombre de cycles d'horloge** par instruction, **NH;**
3. la **durée du cycle** d'horloge, **DH.**

- La plus grande rapidité sera obtenue pour le produit {**NI x NH x DH**} **minimal :**
  - **DH** est fixé par la technique;
  - ➢ on peut **diminuer NI** en faisant exécuter des actions ***plus complexes*** par chaque instruction, en contre partie, **NH** augmente, **on tend vers le CISC;**
  - ➢ on peut **diminuer NI** en faisant exécuter *plusieurs* ***actions simultanément*** par chaque instruction, **NH** augmente peu, **on tend vers le VLIW;**
  - ➢ on peut **diminuer NH** en construisant des ***instructions exécutables*** *en* ***un seul cycle*** d'horloge, elles apportent en plus une **simplification du décodeur** et du **séquenceur**, mais alors **NI** augmente, **on tend vers le RISC**.

CISC et RISC : Une première comparaison peut être faite :

Caractère	CISC	RISC
Taille des instructions	variées	fixe
Durées d'exécution des instructions	variées	fixe sauf les accès à la mémoire
Nombre de formats	grand	petit
Modification implicite des codes de condition par les instructions	oui	non
Utilisation de la microprogrammation	presque toujours	presque jamais
Alignement des instructions et des données	**non,** mises bout à bout	**oui**, sur des adresses multiples de leur longueur
Nombre de registres	petit, au plus 16	beaucoup, couramment plus de 100

- **Pour illustrer quelques points précédents :**

Comparaison des processeurs	CISC *VAX de DEC*	RISC *SPARC de SUN*
Nombre de tailles différentes d'instructions	**54**	**1**
Longueurs des instructions en bits	**de 16 à 456**	**32**
Nombre d'instructions du jeu d'instructions	**244**	**72**
Nombre de modes d'adressage	**22**	**2**
Opérations combinées : chargement, stockage en mémoire et calcul	**OUI**	**NON**

- Le **NI (Nombre d'Instructions)** d'un RISC est couramment 1,3 fois plus grand que celui d'un CISC.
- Le **format fixe** des instructions d'un RISC induit un **décodage simple** réalisable par un **séquenceur câblé** et non plus microprogrammé. Le séquencement occupe 10 à 20% de la puce au lieu de 50% environ, hors cache.
- Les seules instructions d'accès à la mémoire sont LOAD et STORE **directs** ou par **indirection sur registres**. En plus, la fenêtre de registres limite le nombre d'accès à la mémoire pour passer les paramètres des procédures.

Le processeur CISC a beaucoup de modes d'adressage.
Le processeur RISC a peu de modes d'adressage

Le VLIW (Very Long instruction Word) est récent (relativement).

- Cette technique naît avec une première publication en 1983
- Il va s'agir d'utiliser au mieux les **unités fonctionnelles multiples**. par exemple :
  - **Deux unités entières**,
  - Une **unité flottante**,
  - Un **décaleur**,
  - **Deux organes de lecture**
  - Et **un d'écriture**.

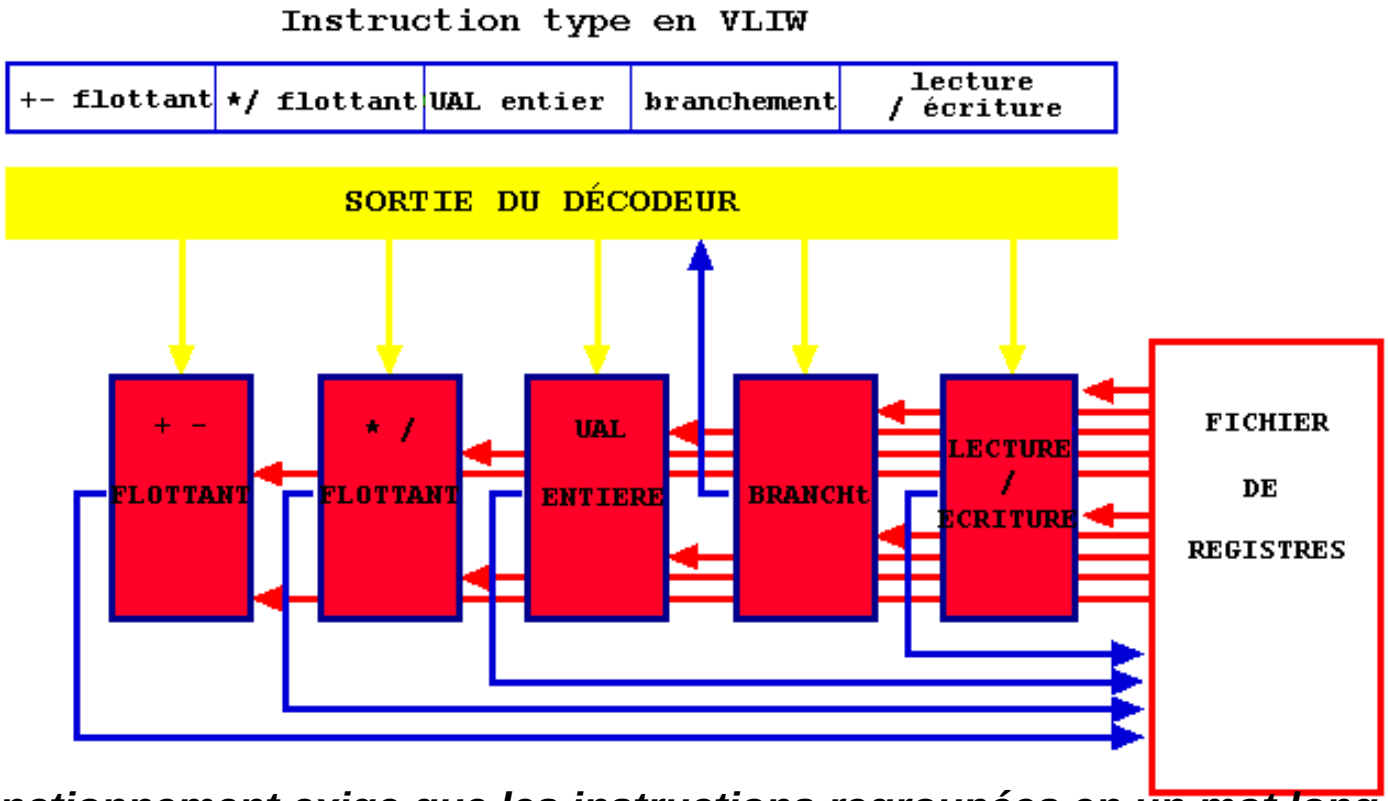
☞ **Principe : Puisque ces opérations peuvent être exécutées en même temps**, autant grouper **plusieurs instructions** en un seul mot qui **devient très long,** d'où le nom de la technique.

- **L'instruction longue** commande plusieurs opérations dans un même cycle.
- Elle est chargée à partir d'un **cache d'instructions** pour éviter d'avoir un bus de mémoire **exagérément large**.

- Chaque unité fonctionnelle est alimentée en instructions par un **pipeline particulier** qui suit une unité de **dégroupage et décodage** des champs de l'instruction très longue.

Le processeur

- ***Le premier VLIW a été l'Advanced flexible processor de Control Data à mot d'instructions de 210 bits en 1982***

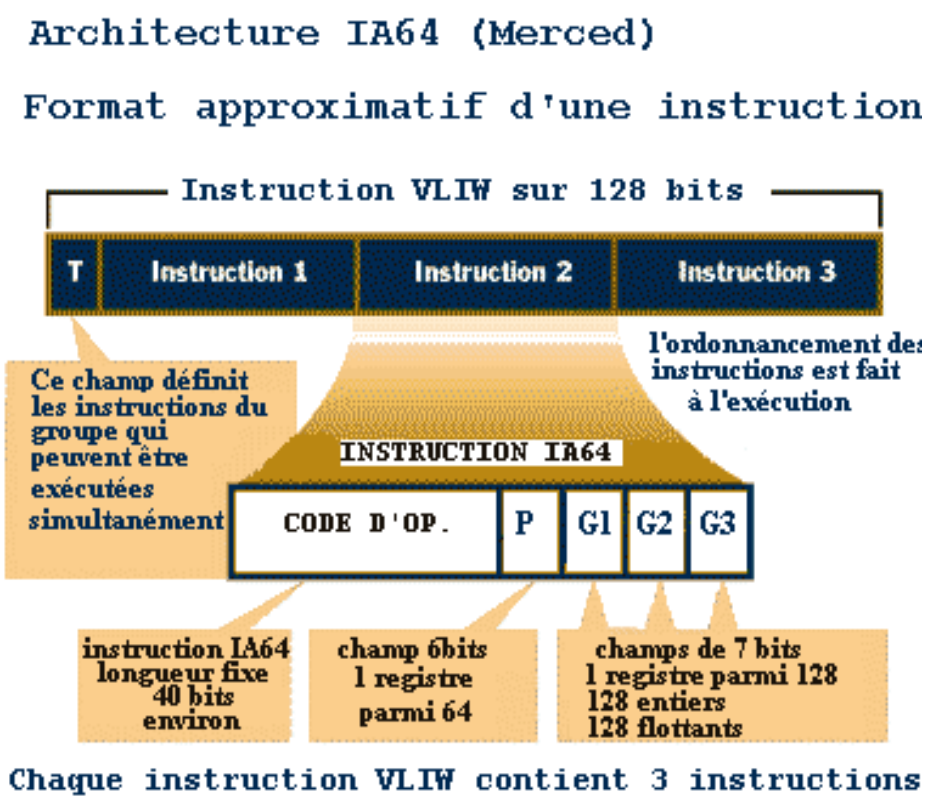


- ***Ce fonctionnement exige que les instructions regroupées en un mot long soient indépendantes l'une de l'autre, c'est à dire qu'aucune n'utilise le résultat d'une autre, en d'autres termes, qu'il n'y ait pas dépendance des données . Les conflits sont prévus et réglés par le compilateur.***

- La figure ci-contre est une vue simplifiée de **l'IA64 ou Merced**, nom employé pendant la phase d'étude de l'architecture **à 64 bits**, commune à Intel et à Hewlett-Packard. Il a été mis en production sous le nom **d'Itanium.**

- On remarquera ci-contre le très grand nombre de registres.

- **Note** : l'Itanium à mots de 64 bits et les Pentium à mots de 32 bits sont deux familles différentes.



(**IA** - **I**ntel **A**rchitecture 64 bits)

2.2 : Classification par les opérandes :

- La distinction entre CISC et RISC a **un succès mérité** mais elle n'est pas la seule pertinente. Toujours en considérant le seul **jeu de codes d'opérations**, on peut utiliser l'origine des opérandes pour distinguer les processeurs.

  - **Registre à registre,** les deux ou trois opérandes sont dans des registres. **Le processeur est simple, l'exécution rapide, le programme contient un grand nombre d'instructions load/store. C'est la situation des RISC (**: *SPARC, MIPS, HP-PA, PowerPC, Alpha)*

  - **Registre à mémoire,** un des opérandes est en mémoire. **Le programme contient moins d'instructions load/store, le code est plus compact. C'est déjà du CISC (**: *Motorola 68000, PDP-10, IBM 360)*.

  - **Mémoire à mémoire,** les deux opérandes sont en mémoire : **Le code du programme est très compact, le processeur est complexe. C'est toujours du CISC** : *PDP-11, NS32x32, IBM 360* (jeu nommé SS set).

  - **Pile,** le premier opérande est placé dans la pile, le second opérande l'est ensuite, l'opération dépile les deux opérandes et place le résultat dans la pile. auxquels on peut ajouter les mélanges des précédents.

Introduction :

- La présentation d'architectures à travers le **jeu d'instructions** (**ISA** ou Instruction **S**et **A**rchitecture) est ***la plus fréquente dans les ouvrages et dans les enseignements***
  - Le jeu d'instructions **définit une machine**.
  - Dans les années 1940 et 1950, ***c'était la machine réelle*.**
  - Aujourd'hui, ce n'est plus la machine physique, c'est la machine **vue par le programmeur** (en langage d'assemblage).
  - Présenter l'architecture d'un système informatique par le jeu d'instructions, c'est présenter une **machine émulée** sur la ***machine matérielle concrète***, celle que l'on voudrait connaître et à laquelle la programmation ne **donne plus accès.**

La conception d'un produit.

- Le processus normal de conception d'un produit ou d'une ligne de produits consiste à définir **un jeu d'instructions ;**
  - Contraintes à prendre en compte :
    - la longueur des mots,
    - les types de données,
    - les langages, …etc.
- Lancer une architecture nouvelle **incompatible avec les précédentes** est plus que risqué :
  - Intel432, IBM 8000, Zilog Z8000, National Semiconductor 3032, Motorola 88000 ont été des échecs.
  - ***Le PowerPC a été un succès***.

Un jeu d'instructions a cinq dimensions principales :

1. Le **nombre d'opérandes explicites** : 0, 1, 2, 3;
2. Les **modes d'accès à la mémoire** ou comment l'adresse en mémoire est-elle spécifiée;
3. **Où les données sont elles chargées ?** registres généraux ou non, pile, etc.
4. **Les types et tailles des opérandes** : octet, entier, etc. et la façon dont ils sont spécifiés;
5. **Le jeu d'opérations** : addition, multiplication, etc.

Parmi les autres aspects, on doit examiner :

- Comment est défini **le successeur** de l'instruction et de la donnée;
- Les **codes de conditions**;
- Le **parallélisme**.

4.3 Fonctions dans un processeur :

1. PRÉRECHERCHE D'INSTRUCTION.

Reçoit	les limites de validité des adresses, l'adresse du mot, l'instruction.
Fait	les tests de limite d'adressage (par rapport au segment et à la page). un stockage de plusieurs octets de code.
Émet	l'adresse vers le bus, les **codes** vers le prédécodage

2. PRÉDÉCODAGE D'INSTRUCTION.

Reçoit	les codes de la prérecherche
Fait	une première transformation du code en ***adressage intermédiaire*** un stockage de deux instructions décodées.
Émet	les instructions décodées vers l***a commande***, les valeurs de ***déplacement vers la segmentation.***

3. COMMANDE ou SÉQUENCEUR ou PIPELINE

Reçoit	les instructions prédécodées, les drapeaux d'état fournis par l'UAL.
Fait	le calcul ***des commandes*** ou celui de ***l'adresse de la microinstruction à rechercher***
Émet	les commandes de tous les organes via le bus interne, les ordres d'opération et les opérandes immédiats vers l'UAL

4. UAL : UNITÉ ARITHMÉTIQUE ET LOGIQUE OU UNITÉS FONCTIONNELLES

Reçoit	Les ordres et opérandes immédiats de la COMMANDE, les données du bus externe.
Fait	Les lectures et écritures des registres, les opérations arithmétiques et logiques.
Émet	Les adresses sur bus internes vers **la SEGMENTATION**, les drapeaux d'état, ***les résultats des opérations vers un bus interne.***

5. SEGMENTATION

Reçoit	Les adresses calculées dans l'UAL, les indications de déplacement issues du décodage.
Fait	Les combinaisons-additions d'adresses, ***entretient les registres descripteurs de segments.***
Émet	Les adresses calculées vers la pagination et la prérecherche

6. PAGINATION

Reçoit	Les adresses dans les segments, les adresses physiques
Fait	Les opérations sur les adresses, entretient la table des pages
Émet	Les adresses physiques vers le ***bus et la prérecherche***

7. COMMANDE DU BUS

Reçoit	les adresses des données à lire et à écrire, les adresses des instructions, les signaux de la commande.
Fait	les multiplexages et démultiplexages, le codage des priorités sur les demandes
Émet	les données, les instructions, les commandes d'interruption

❑ **On distingue ainsi clairement :**

- Les **fonctions de transport** et les **fonctions de stockage**, globalement la logistique, *indispensables à la bonne gestion du système;*
- Les **fonctions de calcul**, essentielles dans la conception des les années 1940 et 1950, secondaires aujourd'hui d'un point de vue architectural car bien maîtrisées.
  - Ceci n'empêche pas des recherches permanentes sur les opérations arithmétiques et logiques, à preuve, les «***Symposium on Computer Arithmetic»*** tenus tous les deux ans.

Révision numérique

- Les ordinateurs utilisent des états électriques que nous représentons par des 0 et 1.
  - Tout est représenté par du binaire, les entiers sont en binaire :

    0000 1010 représente le nombre décimale 10.

- Habituellement, nous regroupons les bits par 8 pour former un octet.
- Cela permet de stocker un caractère car 8 bits permettent 256 combinaisons différentes ce qui est assez pour les jeux de caractères.

Révision numérique

- Les bits sont numérotés de droite à gauche.
- On commence la numérotation par la valeur 0
- Pour passer du binaire au décimal, on multiplie la valeur du bit (0 ou 1) par la puissance de 2 donnée par son numéro.
- Le bit le plus à droite (bit 0) a pour poids : 2^0
- Le bit 1 a pour poids : 2^1
- Le bit 2 a pour poids : 2^2
- Et ainsi de suite…
- Nous avons donc successivement les valeurs suivantes :

$$2^X\ 2^{x-1}\ \ldots\ 2^8\ 2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$$

Révision numérique

- Les ordinateurs utilisent des états électriques que nous représentons par des 0 et 1.
  - Tout est représenté par du binaire, les entiers sont en binaire :

    0000 1010 représente le nombre décimale 10.

- Habituellement, nous regroupons les bits par 8 pour former un octet.
- Cela permet de stocker un caractère car 8 bits permettent 256 combinaisons différentes ce qui est assez pour les jeux de caractères.

Révision numérique

- Les ordinateurs utilisent des états électriques que nous représentons par des 0 et 1.
  - Tout est représenté par du binaire, les entiers sont en binaire :

    0000 1010 représente le nombre décimale 10.

- Habituellement, nous regroupons les bits par 8 pour former un octet.
- Cela permet de stocker un caractère car 8 bits permettent 256 combinaisons différentes ce qui est assez pour les jeux de caractères.

Révision numérique

- Pour obtenir la valeur décimale d'une représentation binaire, il suffit de faire l'addition des différents bits multipliés par leurs poids (puissance de 2).

- Exemple :

$$1011\ 0110\ 1100\ 1010 =$$
$$0*2^0+1*2^1+0*2^2+1*2^3+0*2^4+0*2^5+1*2^6+1*2^7+$$
$$0*2^8+1*2^9+1*2^{10}+0*2^{11}+1*2^{12}+1*2^{13}+0*2^{14}+1*2^{15} =$$
$$1*0+1*2+0*4+1*8+0*16+0*32+1*64+1*128+0*256+$$
$$1*512+1*1024+0*2048+1*4096+1*8192+0*16384+1*32768$$
$$= \mathbf{46794}$$

Révision numérique

- Les ordinateurs utilisent des états électriques que nous représentons par des 0 et 1.
  - Tout est représenté par du binaire, les entiers sont en binaire :

    0000 1010 représente le nombre décimale 10.

- Habituellement, nous regroupons les bits par 8 pour former un octet.
- Cela permet de stocker un caractère car 8 bits permettent 256 combinaisons différentes ce qui est assez pour les jeux de caractères.

Révision numérique

- Les ordinateurs utilisent des états électriques que nous représentons par des 0 et 1.
  - Tout est représenté par du binaire, les entiers sont en binaire :

    0000 1010 représente le nombre décimale 10.

- Habituellement, nous regroupons les bits par 8 pour former un octet.
- Cela permet de stocker un caractère car 8 bits permettent 256 combinaisons différentes ce qui est assez pour les jeux de caractères.

Révision numérique

- En principe, vous êtes supposés savoir faire les conversions des valeurs en puissance de 2 **de tête** :

$2^0 = 1$	$2^8 = 256$	2^{10} +/-= 1 000
$2^1 = 2$	$2^9 = 512$	2^{20} +/-= 1 000 000
$2^2 = 4$	$2^{10} = 1024$	2^{30} +/-= 1 000 000 000
$2^3 = 8$	$2^{11} = 2048$	2^{40} +/-= 1 000 000 000 000
$2^4 = 16$	$2^{12} = 4096$	
$2^5 = 32$	$2^{13} = 8192$	
$2^6 = 64$	$2^{14} = 16384$	
$2^7 = 128$	$2^{15} = 32768$	

❑ Révision numérique

- Utilisation de la base hexadécimale :

➢ Utilisation de 16 symboles :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A ou a, B ou b, C ou c, D ou d, E ou e, F ou f

Les valeurs sont les suivantes :

H	D	H	D	H	D
0	0	7	7	E	14
1	1	8	8	F	15
2	2	9	9		
3	3	A	10		
4	4	B	11		
5	5	C	12		
6	6	D	13		

Révision numérique

- La notation hexadécimal est intéressante car un seul caractère représente un quartet de bit ::

B	H	D	B	H	D
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	A	10
0011	3	3	1011	B	11
0100	4	4	1100	C	12
0101	5	5	1101	D	13
0110	6	6	1110	E	14
0111	7	7	1111	F	15

Révision numérique

- Le passage d'un nombre binaire en hexadécimal doit être inné pour les informaticiens.
- Vous devez être capable de changer de base numéraire facilement du :
  - décimal vers le binaire
  - décimal vers hexadécimal
  - binaire vers décimal
  - binaire vers hexadécimal
  - hexadécimal vers binaire
  - hexadécimal vers décimal

⇒ Le tout de TETE !!!

❑ Révision numérique

❖ Addition binaire :

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 0 \text{ retenue de } 1$$

Application des règles habituelles de l'addition :

$$\begin{array}{rr} & 1\,1\,1\,1\,1\,1\,1\,1\,0 \\ & 1\,0\,0\,1\,1\,0\,1\,1\,0 \\ + & 1\,1\,1\,0\,1\,1\,1\,1\,1 \\ \hline & 1\,1\,0\,0\,0\,1\,0\,1\,0\,1 \end{array}$$

Révision numérique

- La représentation d'un entier négatif s'effectue en utilisant la technique du complément à 2 :

	1100110000110011
Inversion des bits :	0011001111001100
Addition de 1 :	0011001111001101

Le complément à 2 de 1100110000110011 est 0011001111001101

Révision numérique

- Grâce à cette technique, on peut dans un nombre de bits donnés représenter des nombres positifs et négatifs. Le bit de gauche indique alors le signe du nombre, 1 le nombre est négatif, 0 le nombre est positif.
- Par exemple sur 8 bits :
  - Nombre positif de 0 à 127 (0000 0000 à 0111 1111).
  - Nombre Négatif de -128 à -1 (1000 0000 à 1111 1111).

- Addition de deux nombres de signe opposés :

	- 1	1111 1111
+	1	0000 0001
=	0	0000 0000, retenue de dépassement du format 1 (non utilisée).

Révision numérique

- On procède comme en décimal pour effectuer une multiplication binaire :

```
          10101
      *     101
=         10101
      +   00000.
      +   10101..

=       1101001
```

Révision numérique

- On procède comme en décimal pour effectuer une multiplication binaire :

$$
\begin{array}{rcr}
 & & 10101 \\
 & * & 101 \\
= & & 10101 \\
 & + & 00000. \\
 & + & 10101.. \\
 & & \\
= & & 1101001
\end{array}
$$