Architectures des Ordinateurs

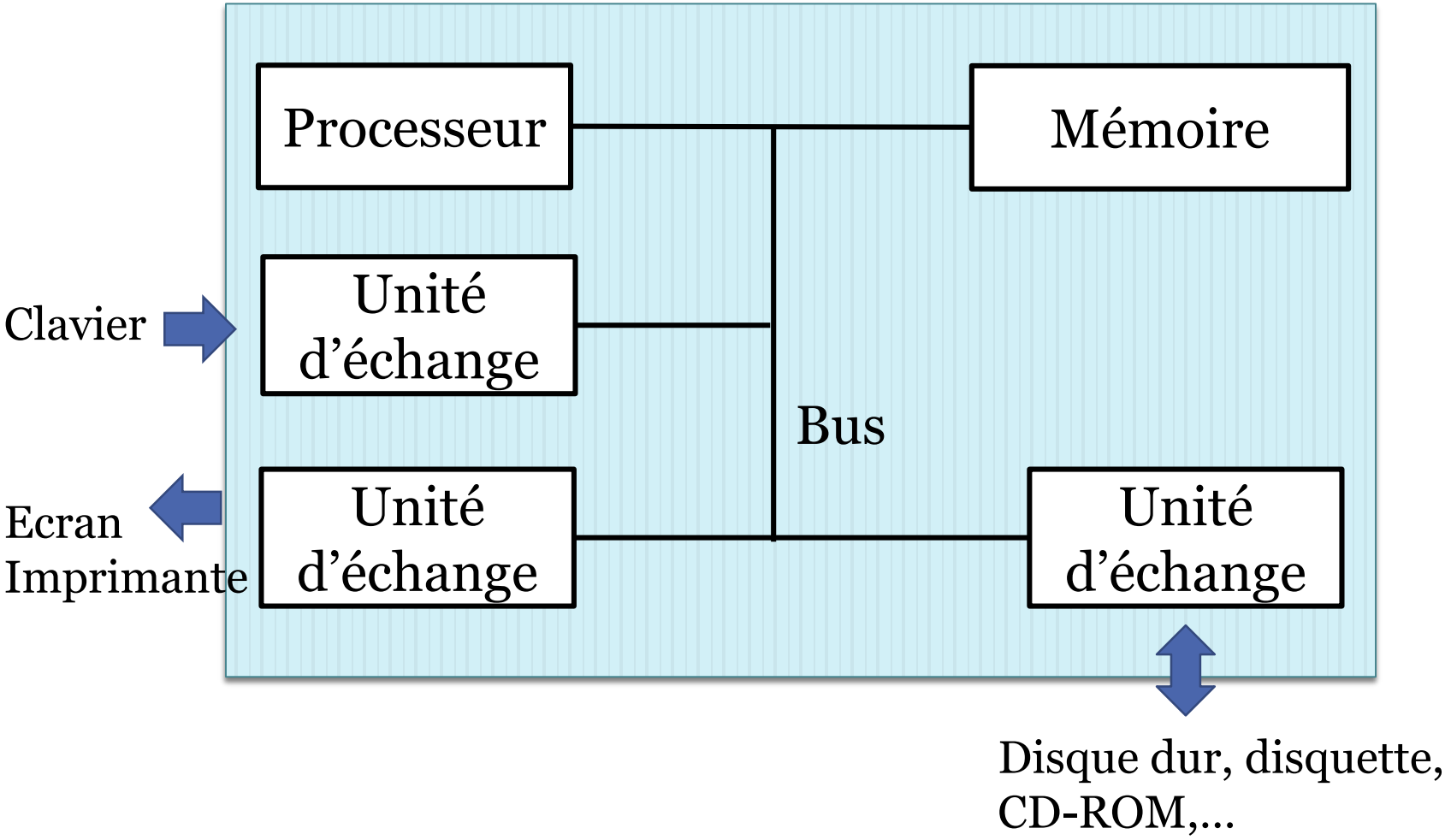
1ère année GI

COURS 4
MICROPROCESSEUR

MAROUA MASMOUDI KOTTI

2022-2023

Introduction



Processeur

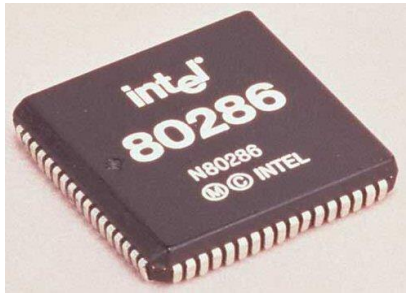
- Le **processeur** ou Unité centrale de traitement (**UCT**) (ou **CPU** de l'anglais *Central Processing Unit*) est le composant de l'ordinateur qui exécute les instructions machine des programmes informatiques.
- Avec la mémoire notamment, c'est l'un des composants qui existent depuis les premiers ordinateurs et qui sont présents dans tous les ordinateurs.
- Un processeur construit en un seul circuit intégré est un **microprocesseur**.

Processeur

- Le processeur est le cœur de l'ordinateur.
- C'est lui qui réalise les opérations. C'est est un circuit électronique cadencée au rythme d'une horloge interne, c'est-à-dire un élément qui envoie des impulsions (top).
- A chaque top d'horloge les éléments de l'ordinateur accomplissent une action.
- A chaque top d'horloge (instructions simples), le processeur :
  - Lit l'instruction à exécuter en mémoire ;
  - Effectue l'instruction ;
  - Passe à l'instruction suivante.

Processeur

- Chaque type de processeur possède son propre jeu d'instructions. On distingue ainsi les **familles** de processeurs, comme :
  - 80x86 : le « x » représente la famille. On parle ainsi de 386, 486, 586, 686, etc.
  - ARM
  - IA-64
  - MIPS
  - Motorola 6800
  - PowerPC
  - SPARC
  - …



1982

1995

2000

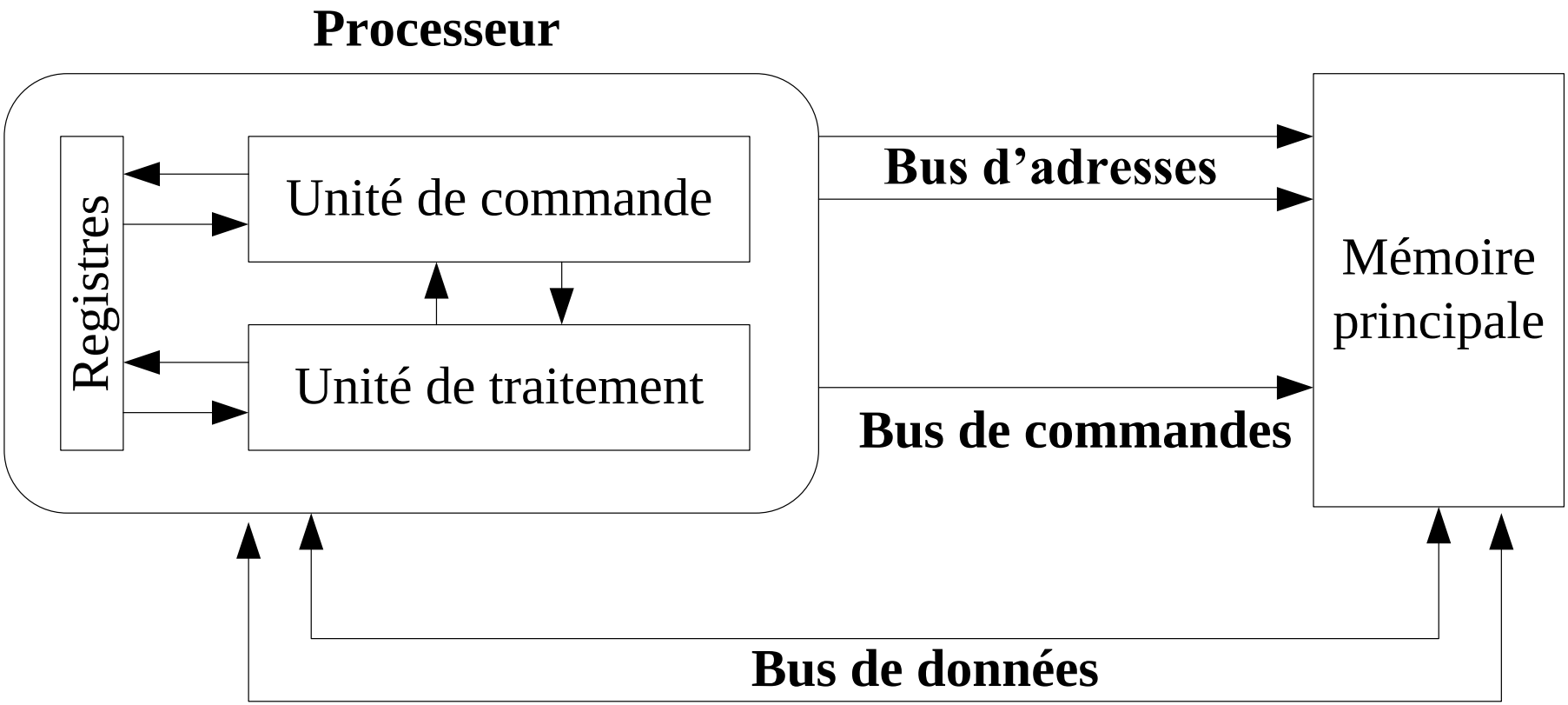
Performances d'un microprocesseur

- 3 critères de performance :
  1. La longueur du mot de données : ou encore largeur du bus de données.
  2. Le nombre d'octets que le microprocesseur peut adresser : ou encore largeur du bus d'adresses.
  3. Le nombre d'instructions qu'il est capable de traiter par seconde.

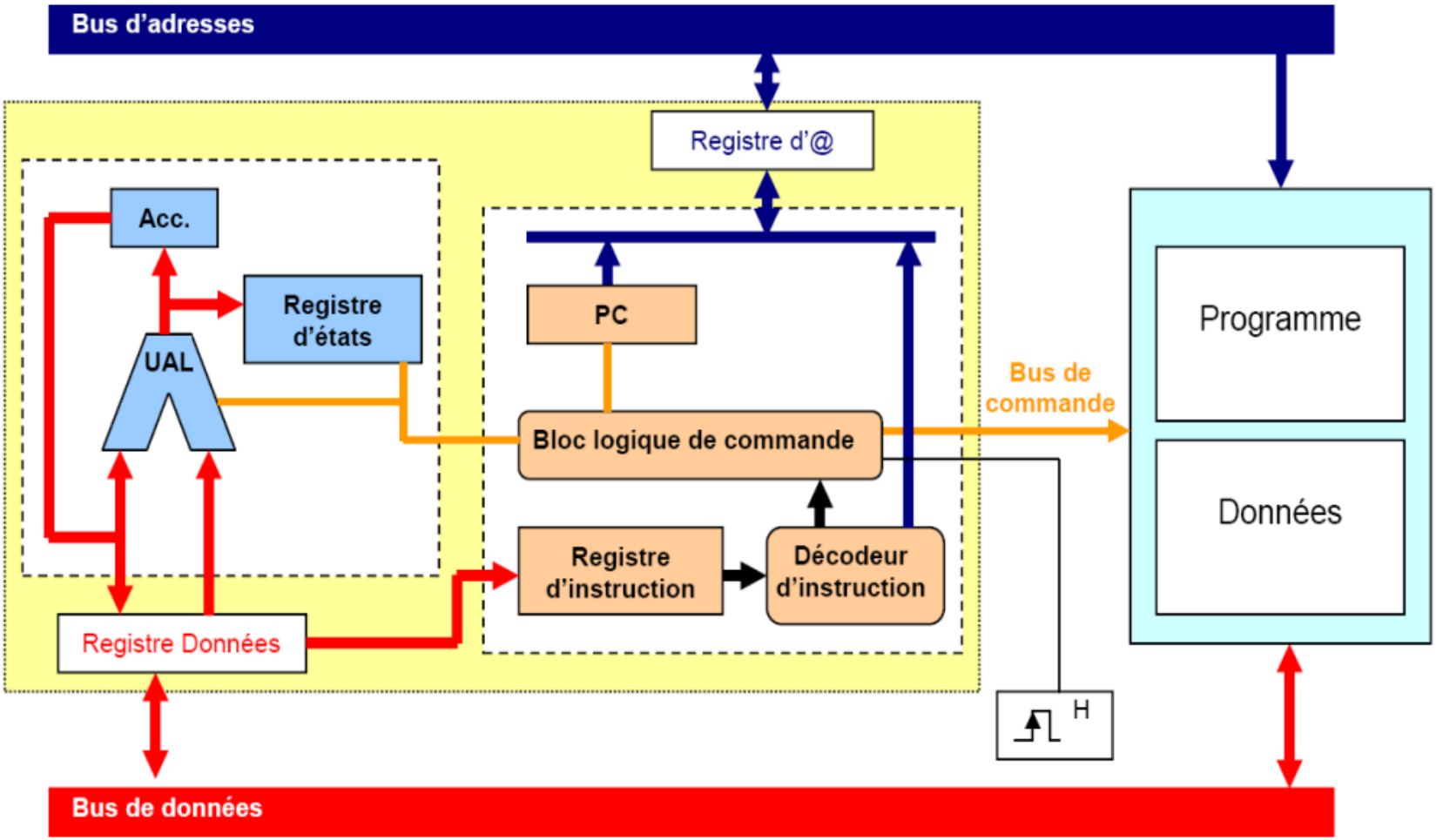
Architecture de base d'un microprocesseur

- Un microprocesseur est construit autour des principaux éléments suivants :
  - L'unité de commande.
  - L'unité de traitement.
  - Les bus.
  - Les registres.

Architecture de base d'un microprocesseur



Architecture de base d'un microprocesseur



Unité de commande

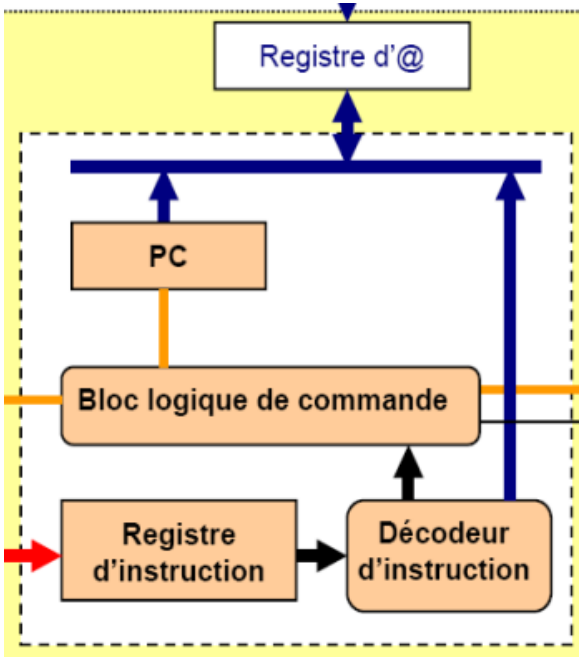
- Elle est responsable :
  - de la lecture des instructions arrivant depuis la mémoire,
  - de la recherche en mémoire de l'instruction
  - du décodage de l'instruction en binaire
  - et de l'envoi des ordres d'exécutions nécessaires à l'unité de traitement par le séquenceur (ou le contrôleur) synchronisé par une horloge.

➢ **Horloge:** (interne ou externe) émet des impulsions permettant la synchronisation de tous les éléments de l'UCT.

➢ Une horloge est un système logique, piloté par un oscillateur, qui émet périodiquement une série d'impulsions calibrées. Ces signaux périodiques constituent le **cycle de base** ou **cycle machine**.

Unité de commande

- Elle contient:
  - Le registre d'instruction (IR): contient l'instruction en cours de traitement (lu en mémoire via le bus de données);
  - Le compteur de programme (PC : Program Counter ou CO : Compteur Ordinal ou IP : Instruction Pointer) contient l'adresse de l'instruction suivante;
  - Le décodeur d'instruction;
  - Le bloc logique de commande (ou séquenceur) : Il organise l'exécution des instructions au rythme d'une horloge.

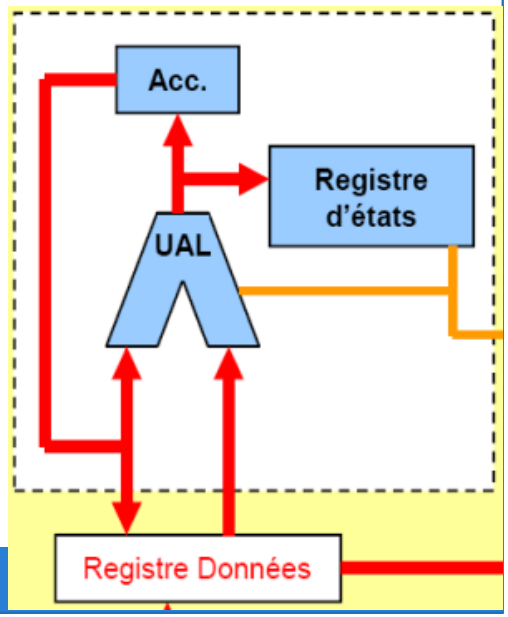


Unité de traitement

- Elle est appelée aussi unité arithmétique et logique (UAL) ou unité de calcul.
- Elle fonctionne sous une impulsion (ordre) de l'unité de commande.
- Elle assure les traitements nécessaires à l'exécution des instructions.

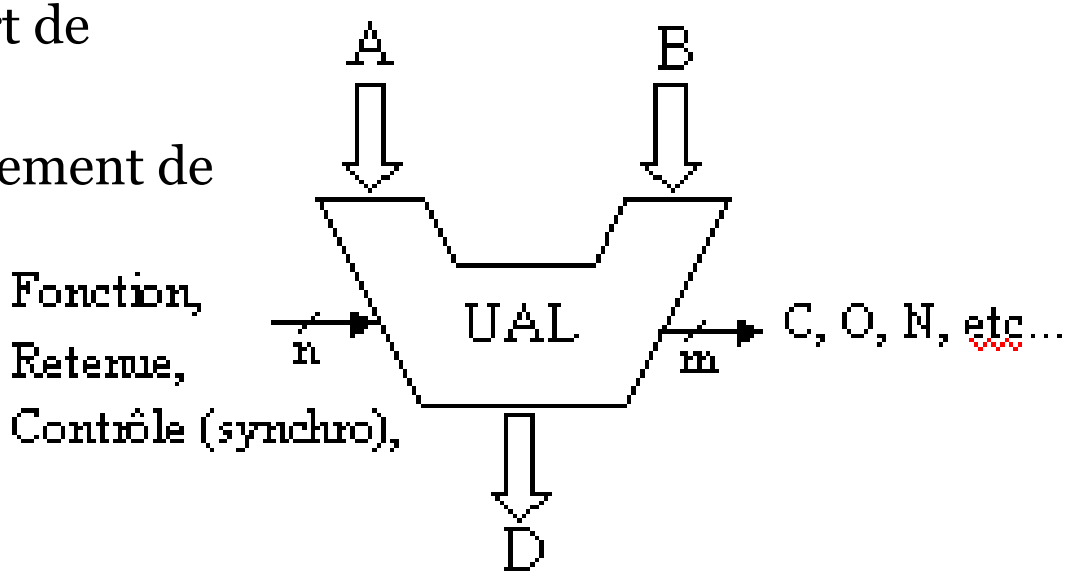
Unité de traitement

- Elle regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions:
  - Les accumulateurs sont des registres de travail qui servent à stocker une opérande au début d'une opération arithmétique et le résultat à la fin de l'opération.
  - L'Unité Arithmétique et Logique (UAL) est un circuit complexe qui assure les fonctions logiques (ET, OU, Comparaison, Décalage, etc…) ou arithmétique (Addition, soustraction…).
  - Registre d'états: stocke les indicateurs d'état appelé drapeaux ou flags indiquant certains états particuliers :
    - signe du résultat (-/+, Zéro) éventuelle retenue (Carry), dépassement de capacité (Overflow).
    - Ces bits indicateurs sont testés pour déterminer la suite du déroulement du programme lors des branchements conditionnels.



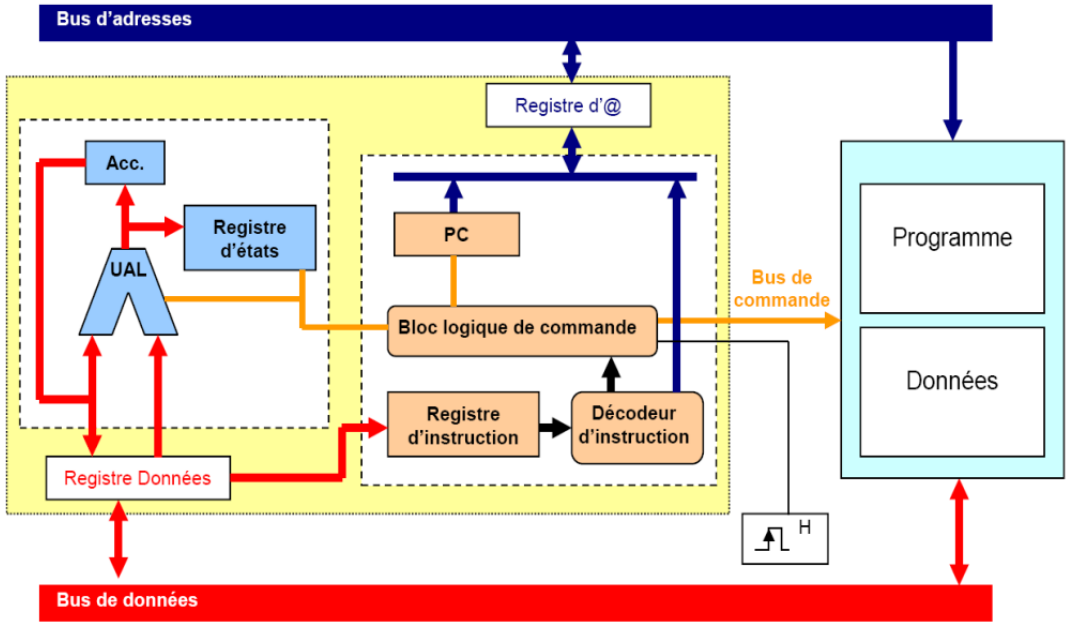
UAL

- Deux opérandes en entrée A et B (pour une opération binaire) et une destination D=F(A, B)
- N lignes en entrée permettant:
  - de sélectionner la fonction à exécuter,
  - d'apporter un éventuel report de retenue,
  - de synchroniser le fonctionnement de l'unité.
- M sorties indiquant une éventuelle
  - retenue,
  - dépassement de capacité,
  - résultat nul, positif ou négatif, etc.

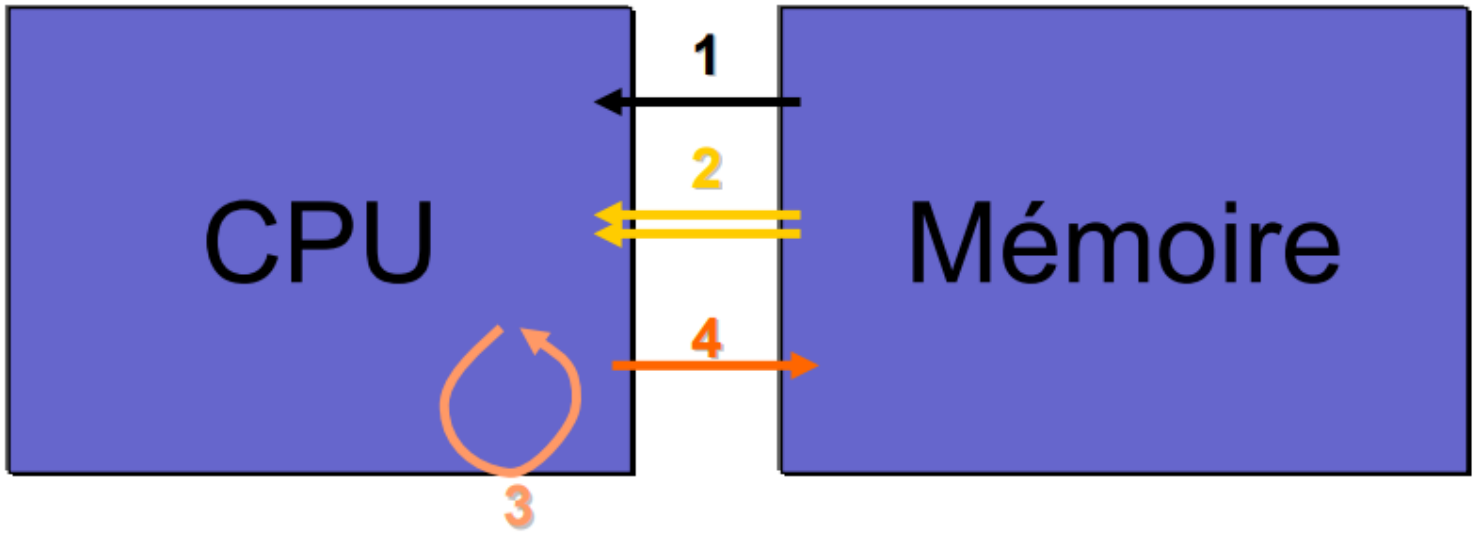


Microprocesseur : Architecture interne complète

- **Registre d'adresses (RA) :** il contient toujours l'adresse de la prochaine information à lire par l'UAL : soit la suite de l'instruction en cours, soit la prochaine instruction ;
- **Registre de données ou registre de Mot (RM)** (mémoire rapide) a pour fonction de contenir les données transitant entre l'unité de traitement et l'extérieur: il contient l'instruction (ou l'opérande).
- **L'horloge** qui synchronise toutes les actions de l'unité centrale ;

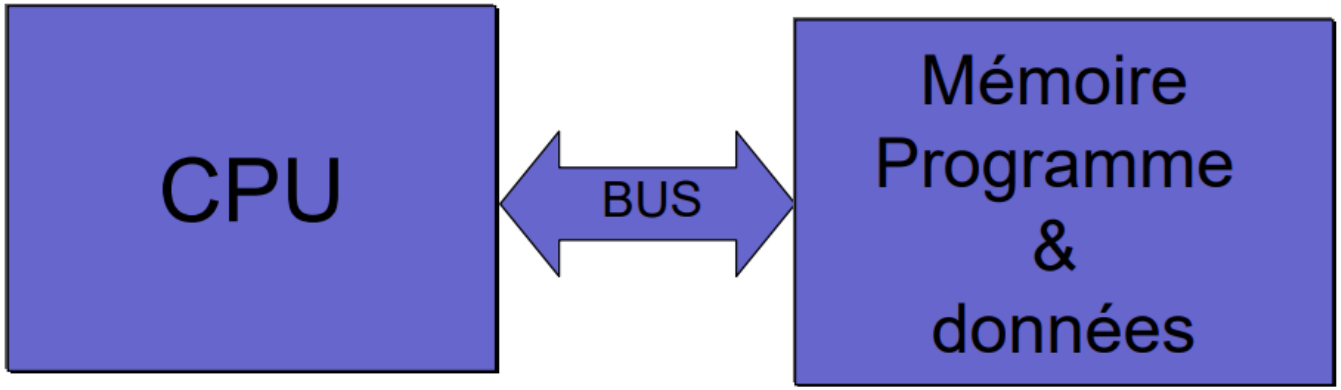


Le fonctionnement basique d'une operation de calcul



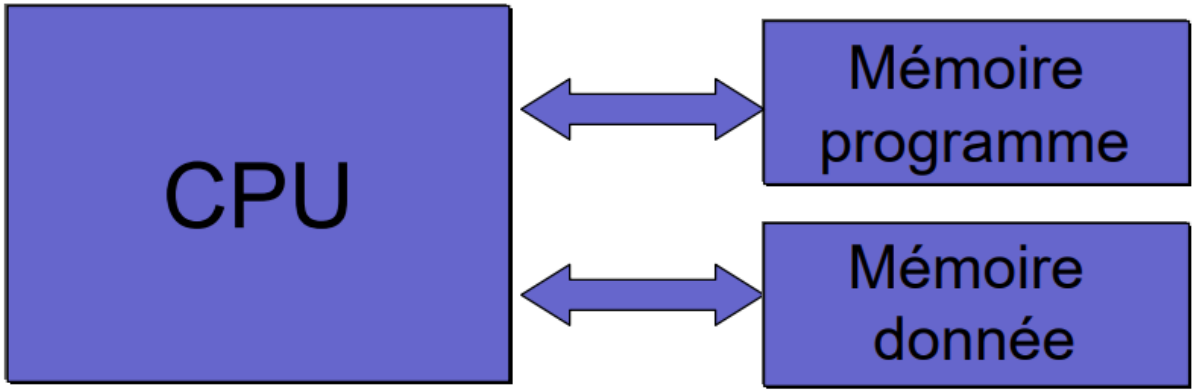
- (1) Charger une instruction depuis la mémoire
- (2) Charger les opérandes depuis la mémoire
- (3) Effectuer les calculs
- (4) Stocker le résultat en mémoire

Architecture Von Neuman



- L'architecture de **Von Neuman** (1945) est un modèle qui utilise une structure unique pour les instructions et les données
- Un seul chemin d'accès à la mémoire
  - Un bus de données (programme et données),
  - Un bus d'adresse (programme et données)
- **Inconvénient**
  - Goulot d'étranglement pour l'accès à la mémoire

Architecture de Havard

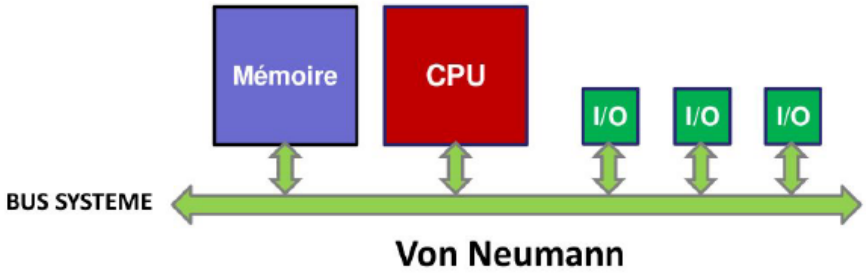


- L'architecture de Harvard sépare physiquement la mémoire de données de la mémoire de programme.
  - Un bus de données programme,
  - Un bus de données pour les données,
  - Un bus d'adresse programme,
  - Un bus d'adresse pour les données.
- Meilleure utilisation du CPU :
  - Chargement du programme et des données en parallèle

Von Neuman VS Havard

1) Modèle de Von Neumann

- 1 seule mémoire (instructions et données)
- 1 bus d'adresse, 1 bus de donnée



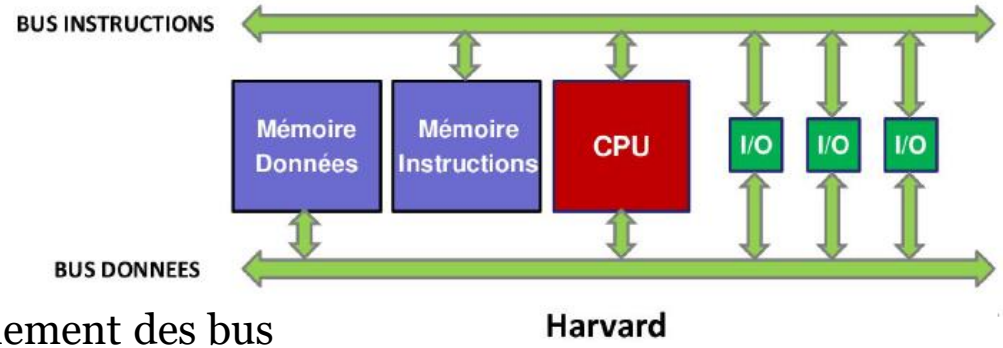
Avantage: gestion unifiée de la mémoire

Inconvénient: goulot d'étranglement pour l'accès à la mémoire

2) Modèle Harvard

- 2 mémoires : 1 pour les instructions, 1 pour les données
- 2 bus d'instructions (adresses / instructions), 2 bus de données (adresses / données)

Avantage: plus grande rapidité d'accès car données et programme peuvent transiter simultanément.

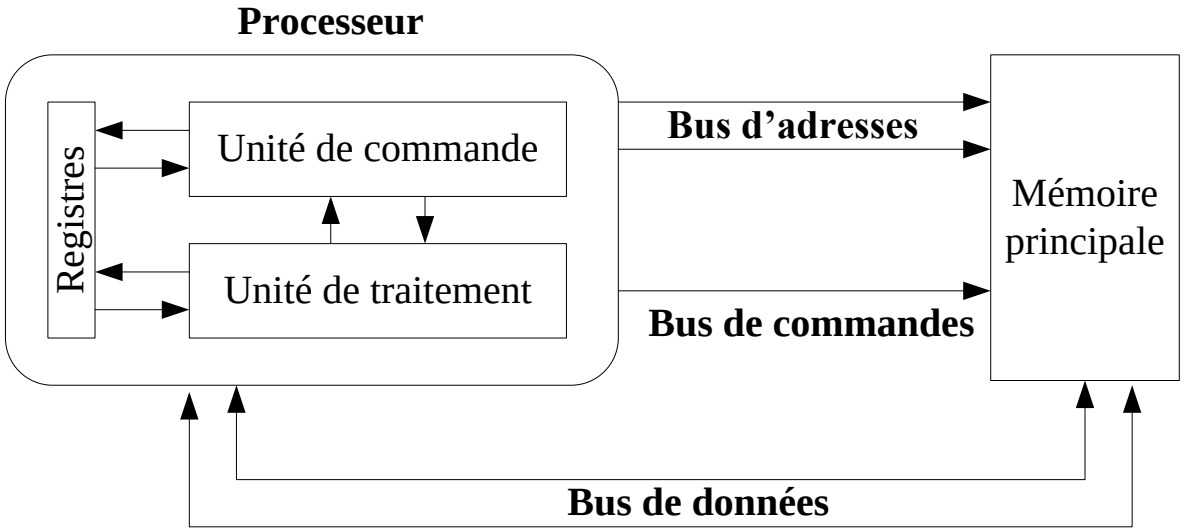


Inconvénients: le prix, à cause du doublement des bus

Et gestion plus complexe car le contrôle des accès données et programme différencié.

Les bus

- On appelle bus, en informatique, un ensemble de liaisons physiques pouvant être exploitées en commun par plusieurs éléments matériels afin de communiquer.
- Ils assurent la communication et l'échange entre les différentes unités du processeur et la mémoire principale.

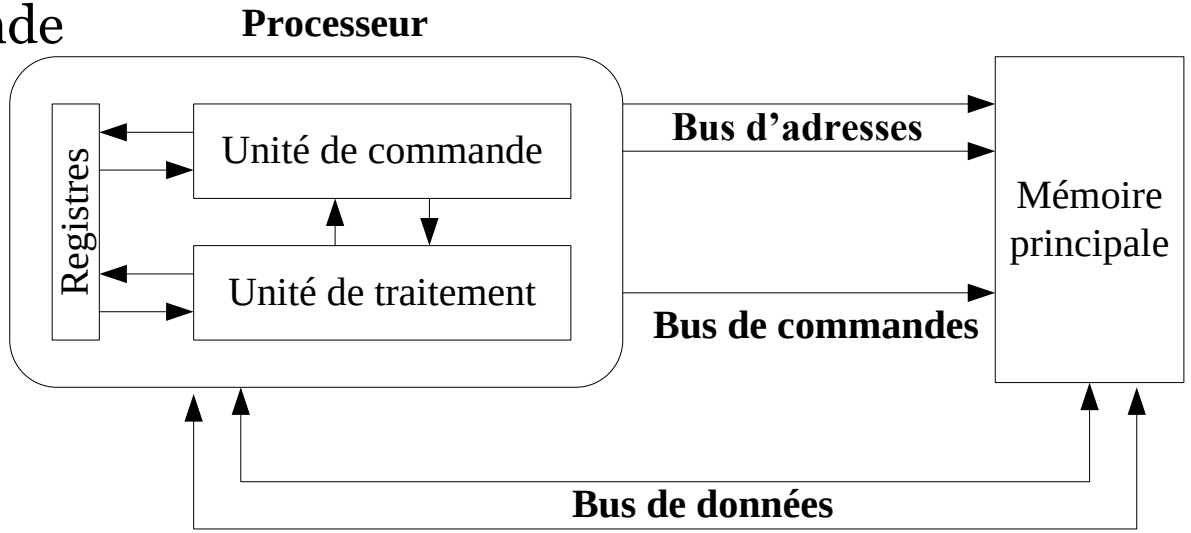


Les bus

- Les bus ont pour but de réduire le nombre de voies nécessaires a la communication des différents composants, en mutualisant les communications sur une seule voie de données.
- Un bus est caractérisé par le volume d'informations transmises simultanément. Ce volume, exprimé en bits, correspond au nombre de lignes physiques sur lesquelles les données sont envoyées de manière simultanée.
- Si un bus est compose de 16 lignes, il pourra envoyer 16 bits en même temps.
- On parle ainsi de largeur pour designer le nombre de bits qu'un bus peut transmettre simultanément.

Les bus

- Les bus interconnectent tous les circuits internes. C'est-a-dire que les différents sous-systèmes de l'ordinateur échangent des données grâce à eux.
- Les types des bus :
  - Le bus d’adresse
  - Le bus de données
  - Le bus de commande



Les bus

- **Le bus d'adresse :**
  - c'est un bus unidirectionnel.
  - Il transporte les adresses mémoire auxquelles le processeur souhaite accéder pour lire ou écrire une donnée: Il permet au processeur de désigner à chaque instant la case mémoire auquel il veut faire appel.
  - Il est composé de ***n*** fils, on utilise donc un bus de ***n*** bits.
  - La mémoire peut posséder au maximum 2^n cases mémoires ou adresses (adressables de 0 à $2^n - 1$).

Les bus

- **Le bus de données :**
  - c'est un bus bidirectionnel.
  - Il permet la circulation des données.
  - Lors d'une lecture, c'est la mémoire qui envoie un mot sur le bus (le contenu de l'emplacement demandé).
  - Lors d'une écriture, c'est le processeur qui envoie le mot de données.

Les bus

- **Le bus de commande :**
  - appelé aussi bus de contrôle.
  - Il transporte les orders et les signaux de synchronisation en provenance de l'unité de commande et à destination de l'ensemble des composants matériels.
  - Il synchronise les transferts de données entre le processeur et les périphériques (la mémoire, les entrées/sorties).
  - On trouve en particulier le signal R/W (Read/Write), qui est utilisé pour indiquer à la mémoire principale si l'on effectue un accès en lecture ou en écriture.

Traitement d'une instruction

- Une instruction est l'unité de travail que peut réaliser un processeur.
- Une instruction est composée de deux éléments :
  - Le code operation : C'est un code binaire qui correspond a l'action à effectuer par le processeur.
  - Le champ operande : Donnée ou bien adresse de la donnée.
- La taille d'une instruction peut varier, elle est généralement de quelques octets (1 a 8), elle dépend également de l'architecture du processeur.

Jeu d'instructions

- Le jeu d'instructions est l'ensemble des instructions-machine qu'un processeur peut executer.
- Ces instructions-machines permettent d'effectuer des opérations élémentaires (addition, ET logique, etc.) ou plus complexes (division, passage en mode basse consommation. . .).
- Le jeu d'instructions définit quelles sont les instructions supportées par le processeur.
- Le jeu d'instructions précise aussi quels sont les registres du processeur manipulables par le programmeur (les registres architecturaux).

Jeu d’instructions

- Le jeu d'instruction des PC actuels est le x86, un jeu d'instructions particulièrement ancien, apparu en 1978.
- Les anciens macintoshs (la génération de macintosh produits entre 1994 et 2006) utilisaient un jeu d'instruction différent : le PowerPC (depuis 2006, les macintoshs utilisent un processeur X86).
- Mais les architectures x86 et Power PC ne sont pas les seules au monde : il existe d'autres types d'architectures qui sont très utilisées dans le monde de l’informatique embarquée : architectures ARM, MIPS et SPARC. Pour résumer, il existe différents jeux d'instructions, que l'on peut classer suivant divers critères.

Jeu d'instructions
Architecture RISC et CISC

RISC (Reduced Instruction Set Computer)	CISC (Complex Instruction Set Computer)
Instructions simples d'un seul cycle	Instructions complexes de plusieurs cycles
Instructions au format fixe	Instructions au format variable
Décodeur simple (câble)	Décodeur complexe (microcode)
Beaucoup de registres	Peu de registres
Peu de modes d'adressage	Beaucoup de modes d'adressage
Compilateur complexe	Compilateur simple

- La plupart des architectures actuelles sont de type RISC, mais l'architecture x86 d'Intel est de type CISC.

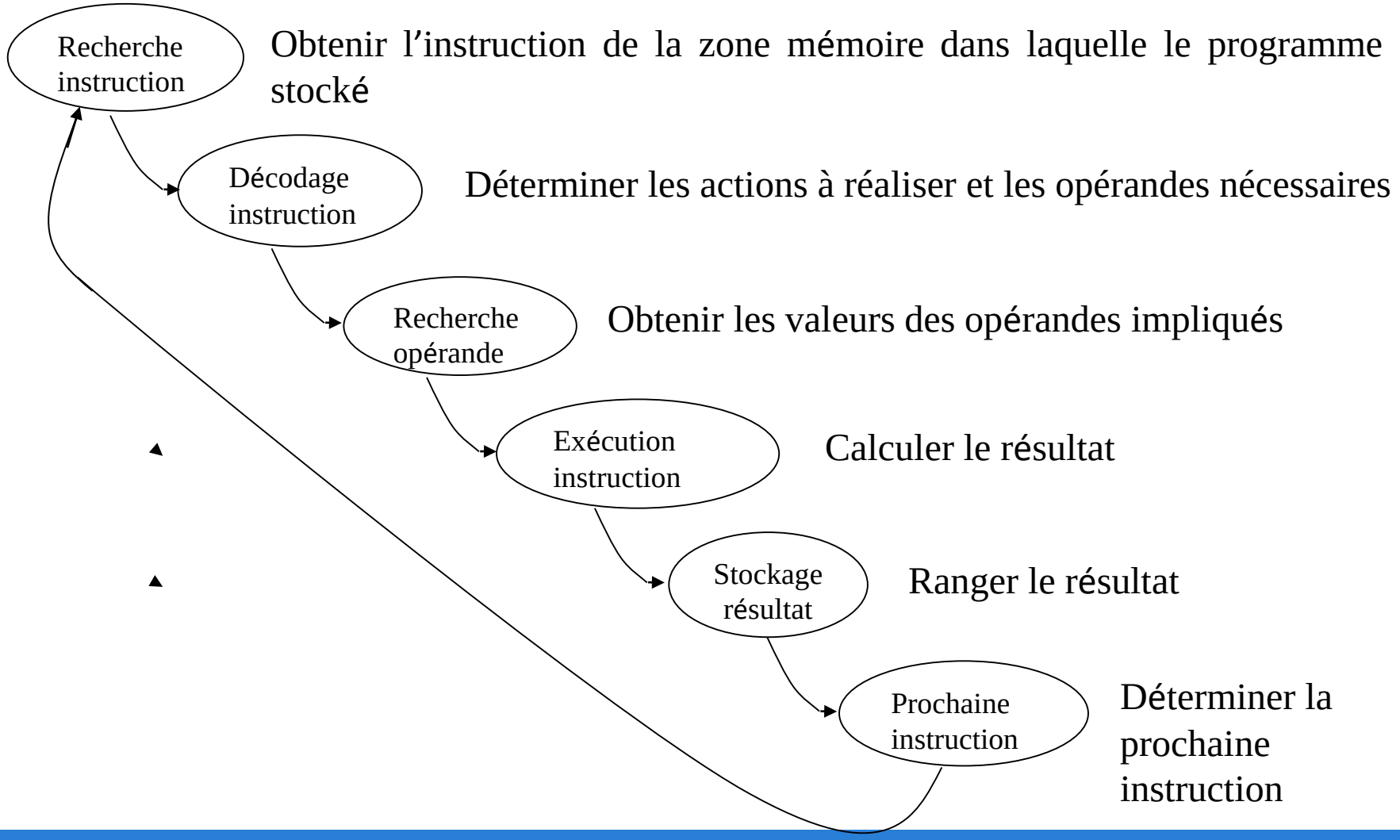
Processeur CISC

- Avant les années 80, la tendance était d'augmenter le nombre d'instructions du processeur de manière à faciliter son utilisation.
- Les processeurs CISC possèdent un jeu étendu d'instructions complexes. Chacune de ces instructions peut effectuer plusieurs opérations élémentaires comme charger une valeur en mémoire, faire une opération arithmétique et ranger le résultat en mémoire.
  - En une seule instruction, traitement de séquences complexes d'opérations
  - Permettait d'éviter l'appel à plusieurs instructions et donc plusieurs accès (lents) à la mémoire
- Inconvénient: seule une petite proportion du jeu d'instruction est utilisée en pratique.
- Exemples: Motorola 68000, Intel Pentium

Processeur RISC

- Reduced Instruction Set Computer
- 1980: conception du premier processeur RISC (Berkeley)
  - Architecture beaucoup plus simple à concevoir
  - Niveau de performance similaire
- Principe:
  - Format fixe d'instruction (32-bit), 1 instruction = 1 cycle
  - Architecture load-store: les instructions de traitement opèrent sur des registres et sont séparées des instructions d'accès à la mémoire
  - Une file de registres importante (32) pour permettre à l'architecture load-store d'opérer efficacement
- Ex: ARM (Advanced RISC Machine)

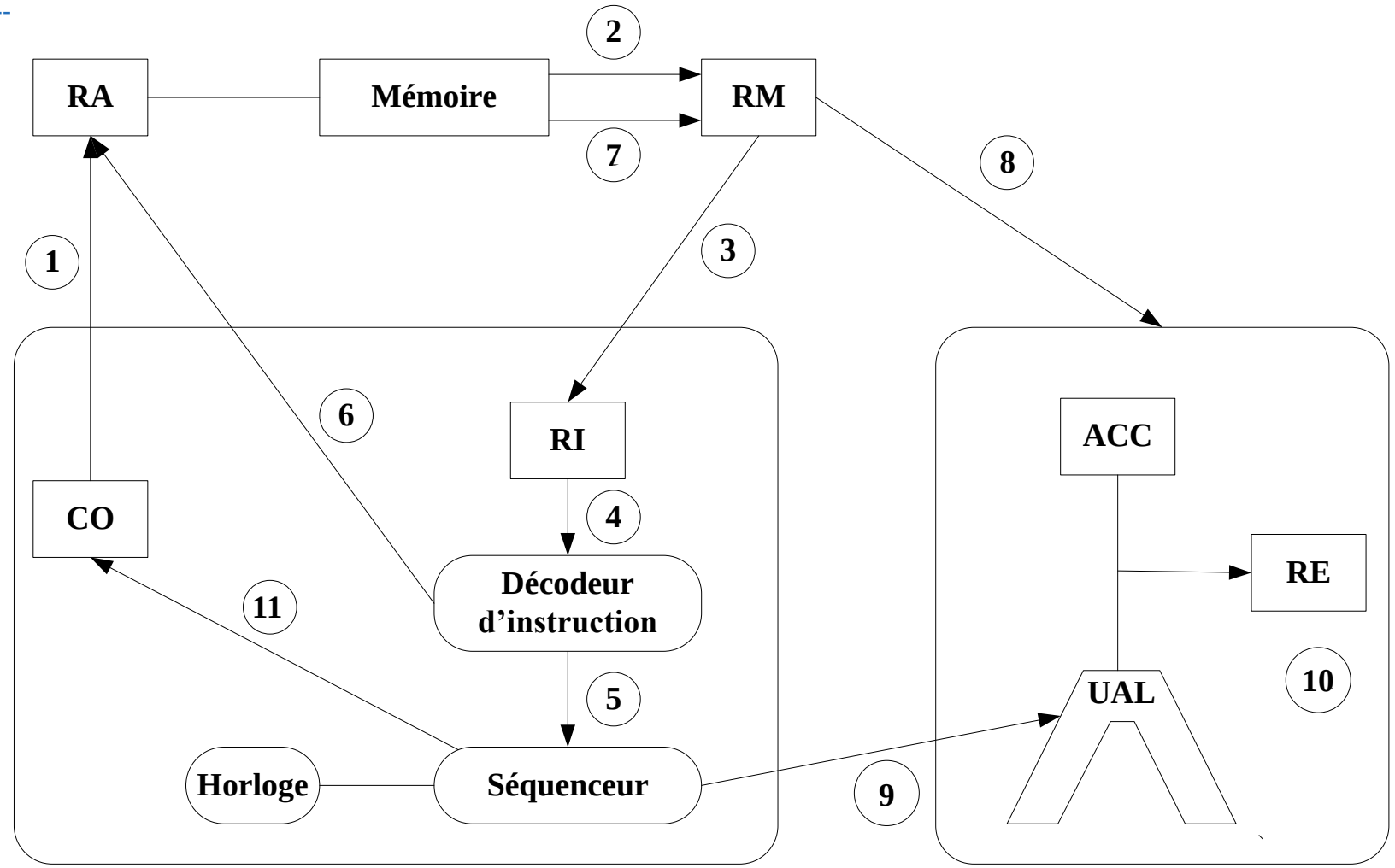
Cycle d'exécution d'une instruction



Cycle d'exécution d'une instruction

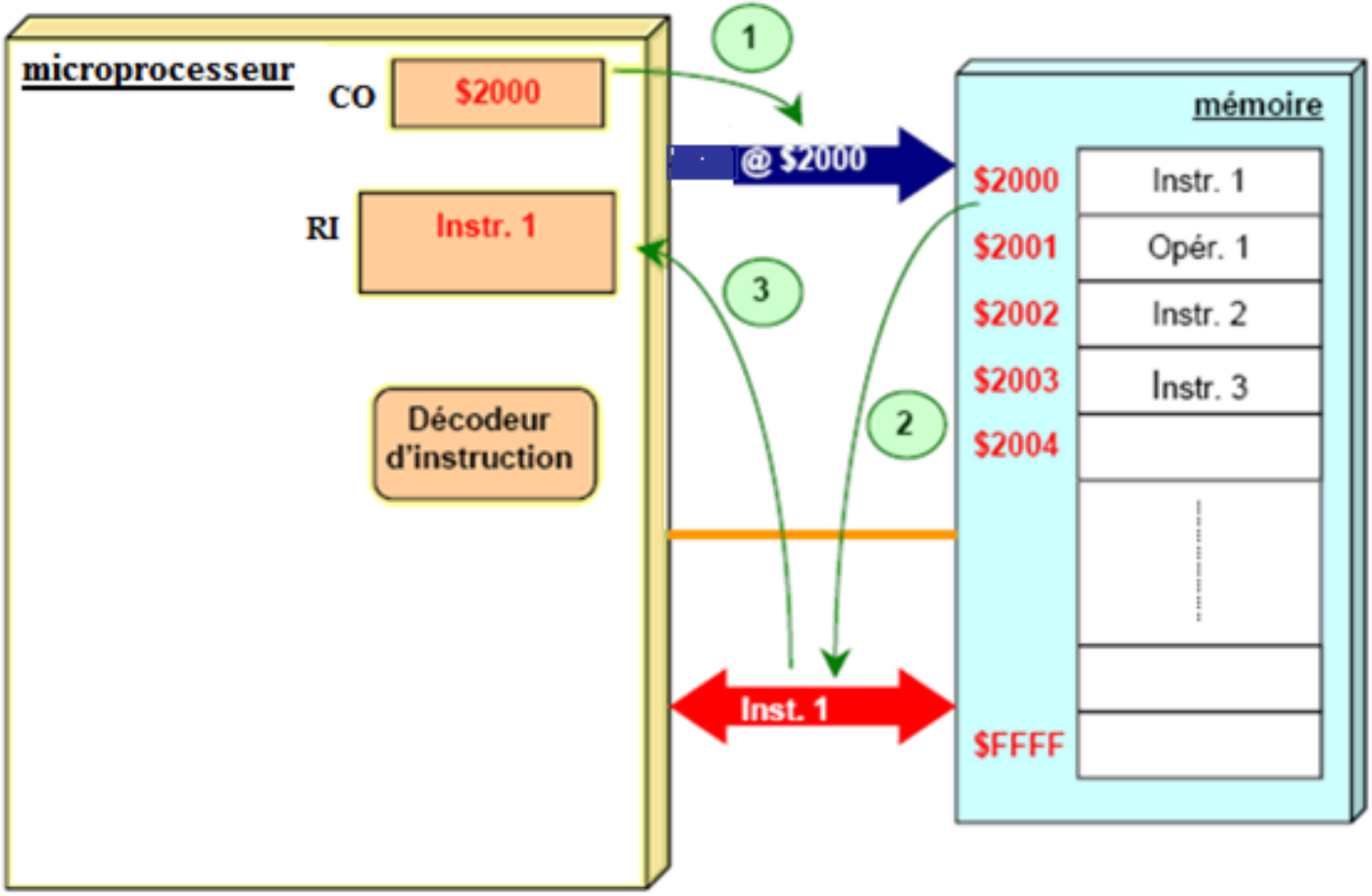
1. Transfert de l'adresse de la nouvelle instruction du CO vers le RA **(Recherche instruction)**.
2. Transfert de l'instruction cherchée dans le RM.
3. Transfert de l'instruction du RM vers le RI.
4. **Décodage** de l'instruction par le décodeur d'instructions ce qui permet de déterminer l'opération à exécuter et l'adresse des opérandes.
5. L'unité de commande donne **l'ordre d'exécution** à l'UAL.
6. Chargement de l'adresse de l'opérande sur le RA (**recherche opérandes)**.
7. Le contenu de l'emplacement mémoire correspondant à l'adresse de l'opérande est placé dans le RM.
8. Transfert de contenu du RM dans l'accumulateur ou tout autre registre concerné par l'opération en cours.
9. **Exécution** de l'opération par l'UAL sous le contrôle du Séquenceur.
10. Les drapeaux du RE sont positionnés.
11. Incrémentation du CO pour pointer sur la prochaine opération.

Cycle d'exécution d'une instruction



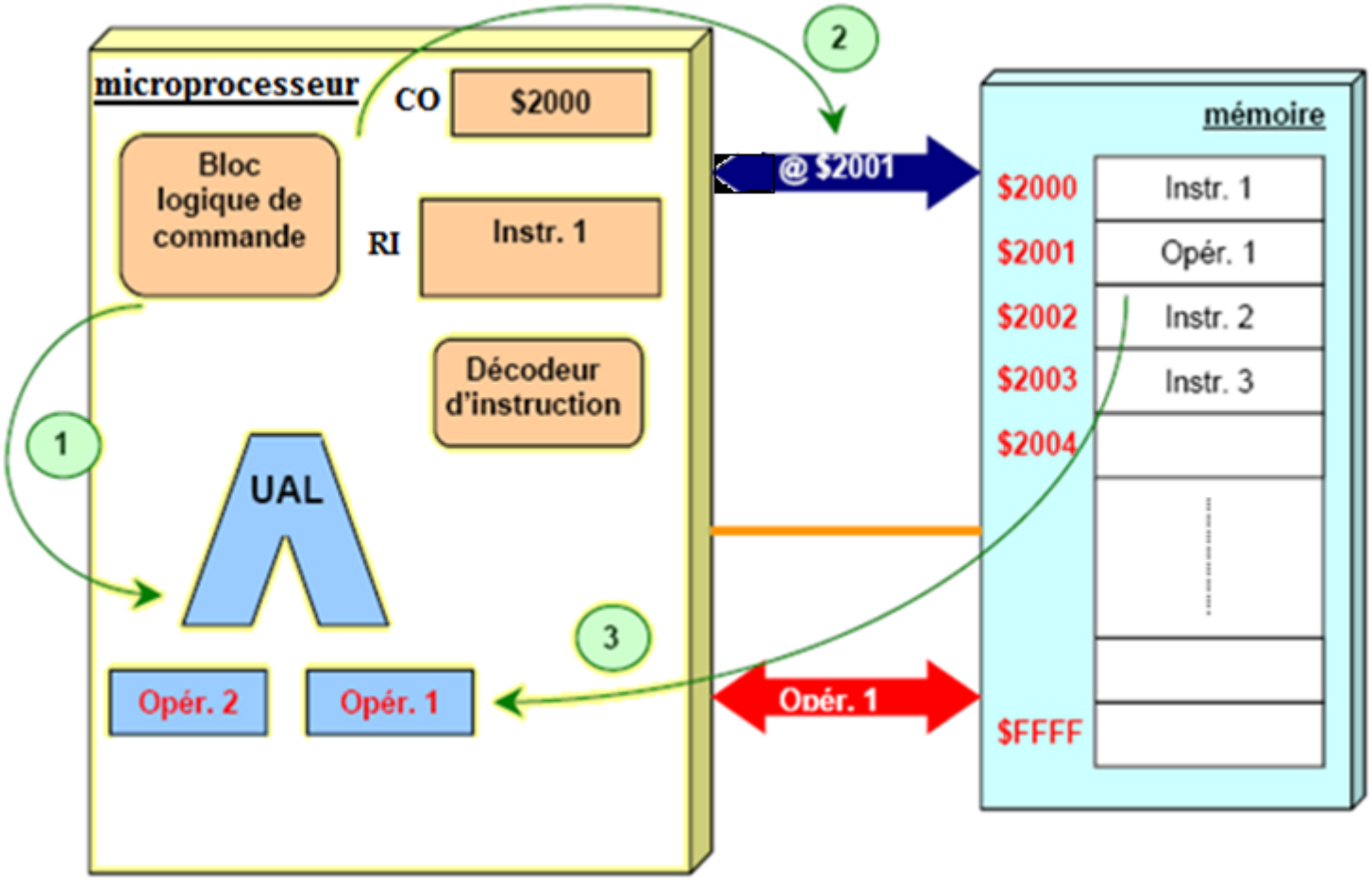
Cycle d'exécution d'une instruction

- **Phase 1 : Recherche de l'instruction à traiter**



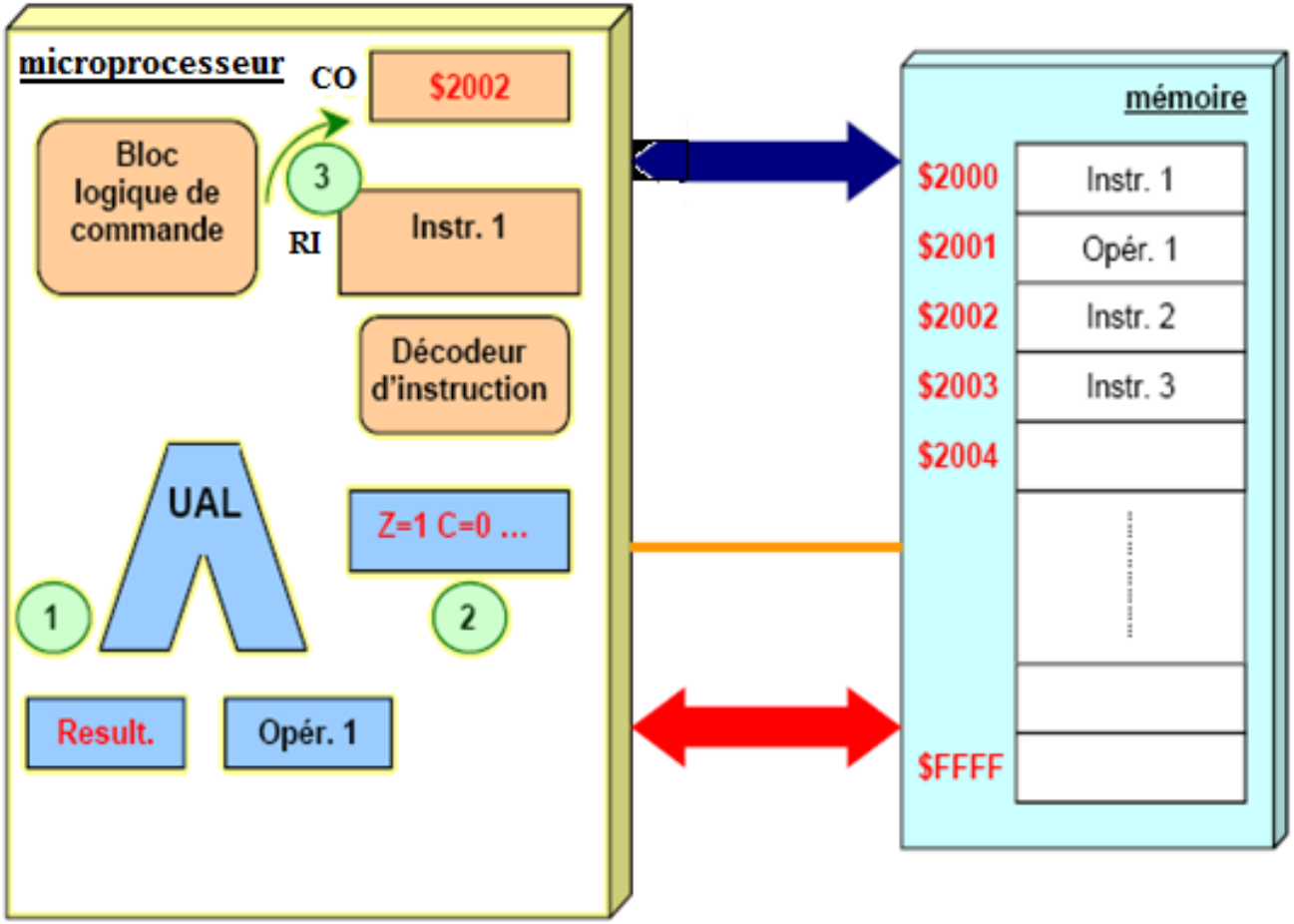
Cycle d'exécution d'une instruction

- **Phase 2 : Décodage de l'instruction et recherche de l'opérande**



Cycle d'exécution d'une instruction

- **Phase 3 : Exécution de l'instruction**



Structure d'une instruction

- Les ordinateurs sont capables de faire un certain nombre d'opérations simples, par exemple :
  - additionner deux nombres,
  - tester le signe d'une valeur numérique,
  - copier le contenu d'un registre dans un autre registre,
  - stocker en mémoire le résultat d'une opération,
  - etc.

Structure d'une instruction

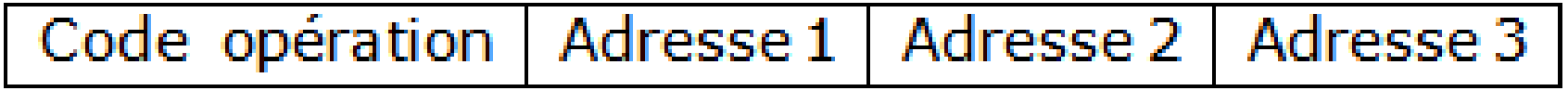
- Une instruction machine doit fournir au CPU toutes les informations pour déclencher une opération élémentaire :
  - un code opération qui est essentiel pour spécifier le type d'action demandé (1er champ).
  - une ou plusieurs adresses, par exemple, l'adresse de l'opérande (ou des opérandes), l'adresse de sauvegarde du résultat, l'adresse de l'instruction suivante (dans les champs suivants).
  - Par conséquent le format d'une instruction machine comportera un champ code opération et jusqu'à quatre champs adresse. On parle d'instructions à ***n*** adresses, ***n*** = 0, 1, 2, 3, 4.

Structure d'une instruction

Instruction à zéro adresse

Instruction à une adresse

Instruction à deux adresses



Instruction à trois adresses

Structure d'une instruction

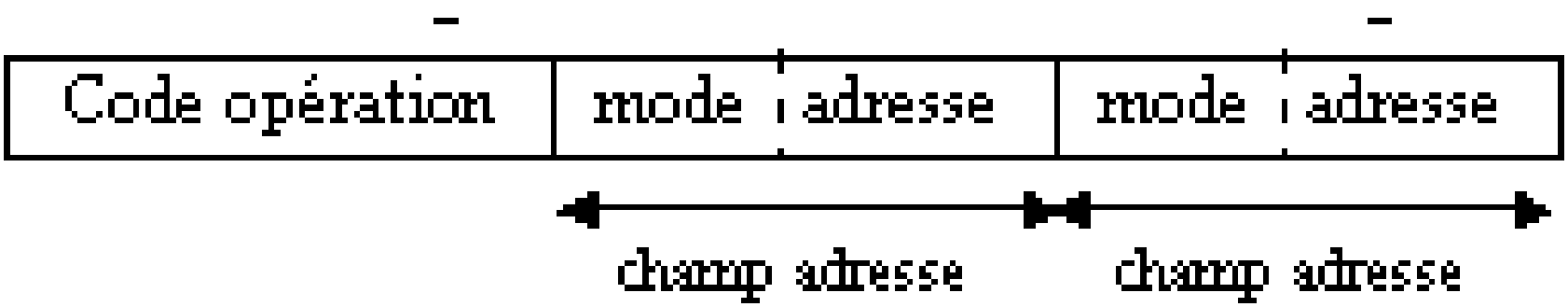
- On distingue six groupes d'instructions :
  - **Instructions de transfert de données** : le transfert peut être de la mémoire au registre, de registre au registre, de registre à mémoire (exemple : LOAD, MOVE, STORE).
  - **Instructions arithmétiques** : addition, soustraction, multiplication, …
  - **Instructions logiques** : ET, OU, NON, XOR, …
  - **Instructions de contrôle de séquence** : branchements conditionnels ou non, appel de procédure, etc.
  - **Instructions d'entrée/sortie** (READ, WRITE,…).
  - **Instructions de manipulations diverses** : décalage, conversion de format, permutation circulaire de bits, échange d'octets, incrémentation, etc.

Modes d'adressage

- Les adresses spécifiées par les opérandes informent le processeur sur les emplacements des données.
- Un champ adresse peut permettre de référencer un registre ou un mot en mémoire.
- Il peut contenir le numéro du registre ou l'adresse effective du mot.
- Mais ce ne sont pas les seules manières d'identifier un opérande.
- Il existe plusieurs manières pour identifier une opérande → **modes d'adressage**.

Modes d'adressage

- Il existe plusieurs modes d'adressage qui spécifient la manière dont chaque instruction détermine l'emplacement des valeurs des opérandes.
- Le mode est défini soit par le code opération lorsque celui-ci impose un type déterminé, soit par un code faisant partie du champ adresse.



Modes d'adressage

- Les différents modes d'adressage:
  - Adressage implicite
  - Adressage immédiat
  - Adressage d'un registre
  - Adressage direct
  - Adressage indirect
  - Adressage indirect par registre avec déplacement.

Adressage implicite

- Le mode d'adressage implicite correspond à une instruction contenant uniquement le code opérande (zéro @).
- Il porte sur des registres particuliers (adressage registre) → le code opération identifie automatiquement l'opérande (un registre).
- Exemples :
  - L'instruction NOT AX
  - incrémentation de l'accumulateur : l'opérande=ACC.
  - test signe du résultat: l'opérande=RE

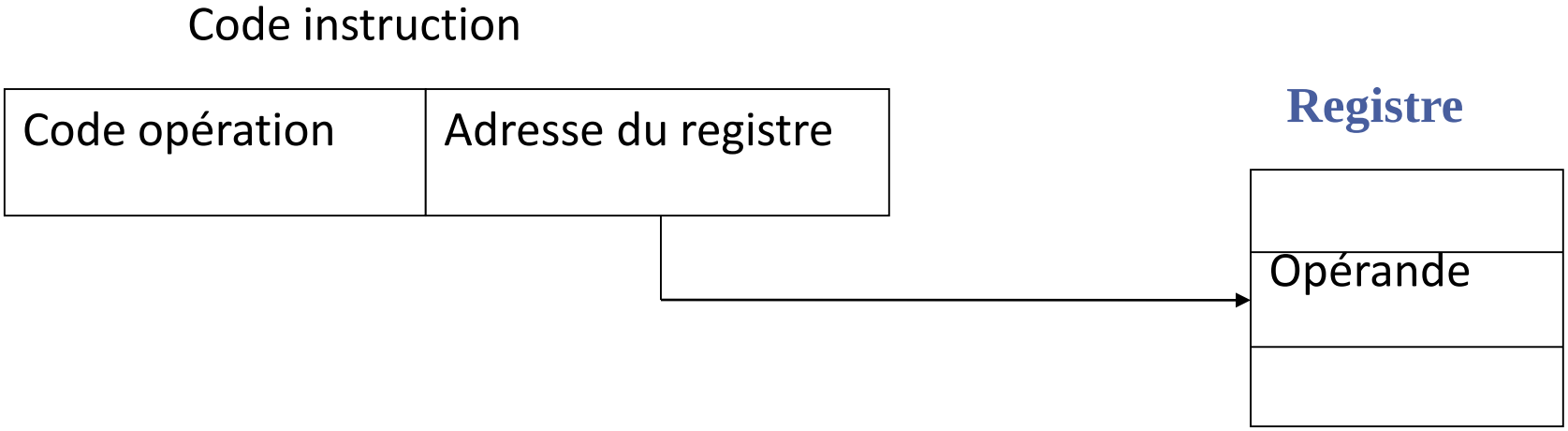
Adressage immédiat

- L'instruction supporte la donnée à utiliser : la valeur de l'opérande représentée en hexadécimal est contenue dans le champs adresse.
- Exemple : MOV AX, 20.

Code opération	Valeur de l'opérande

Adressage registre

- Le champ adresse contient l'adresse du registre contenant l'opérande.



- Exemple : MOV AX, BX

Adressage direct

- L'instruction contient l'adresse d'un emplacement mémoire stockant la donnée. L'adresse de l'emplacement est représentée en hexadécimal et placée entre crochets.
- Exemple :

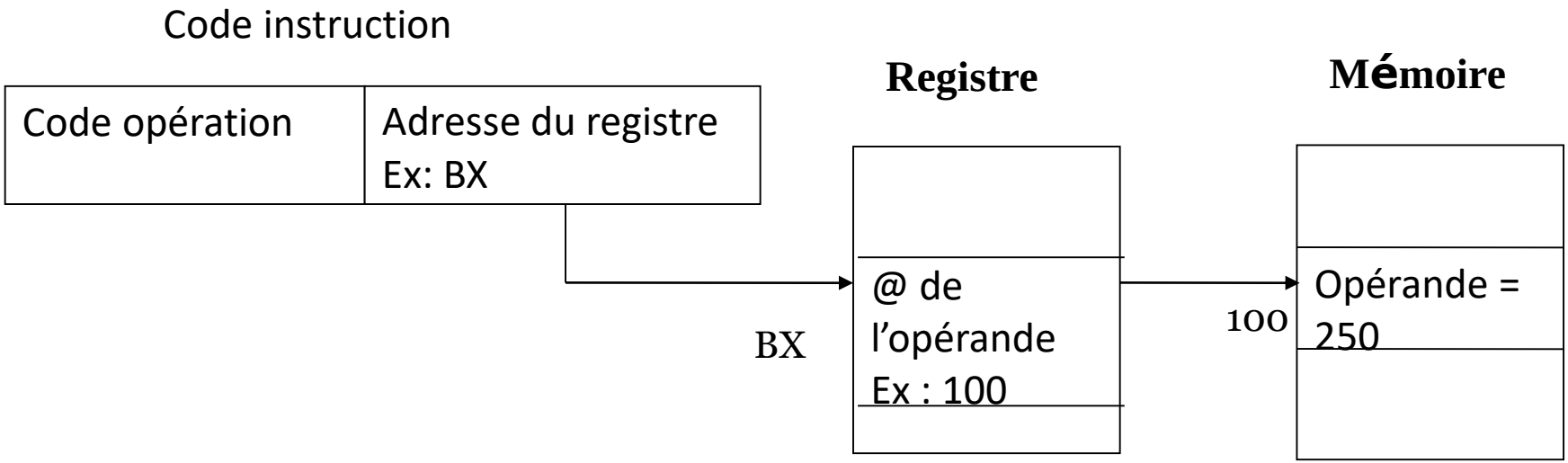
  MOV AX, [20].

  → Sachant que la case mémoire à l'adresse 20 hex contient 250;

  → AX prend la valeur contenue à l'adresse 20 de la mémoire, c à d, 250.

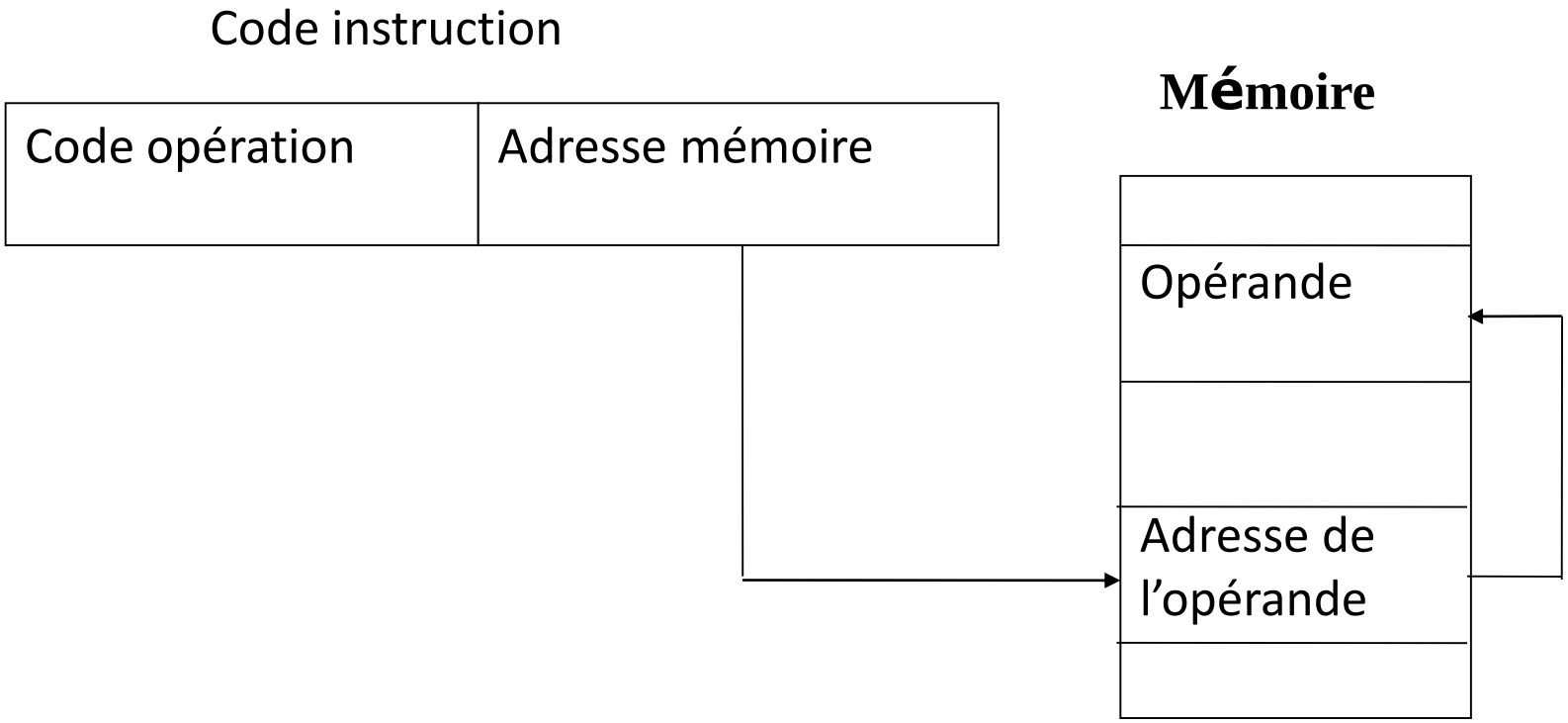
Adressage indirect

- Le champ adresse contient l'adresse d'un pointeur sur la donnée (l'adresse de l'adresse peut être contenue dans un registre ou dans une case mémoire)
- Exemple : Adressage indirect par registre (adressage basé)
  - MOV AL, [BX]



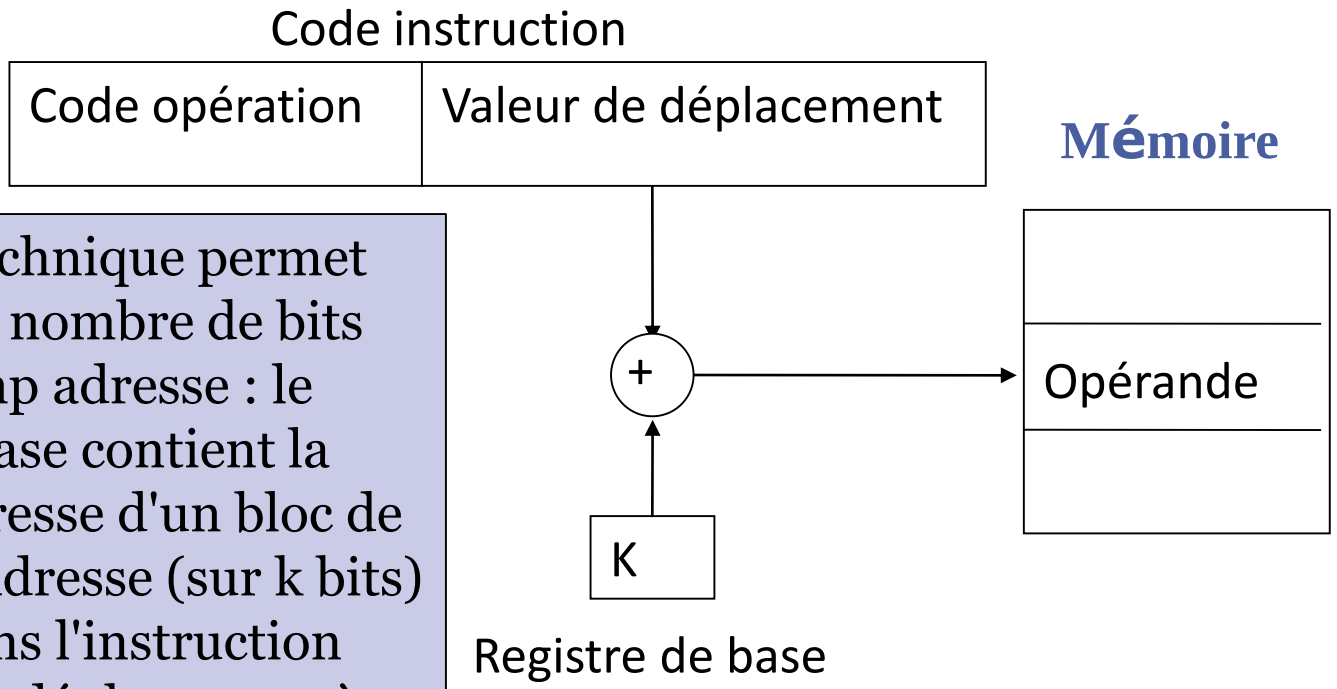
Adressage indirect

- Exemple : Adressage indirect par mémoire



Adressage indirect par register avec déplacement

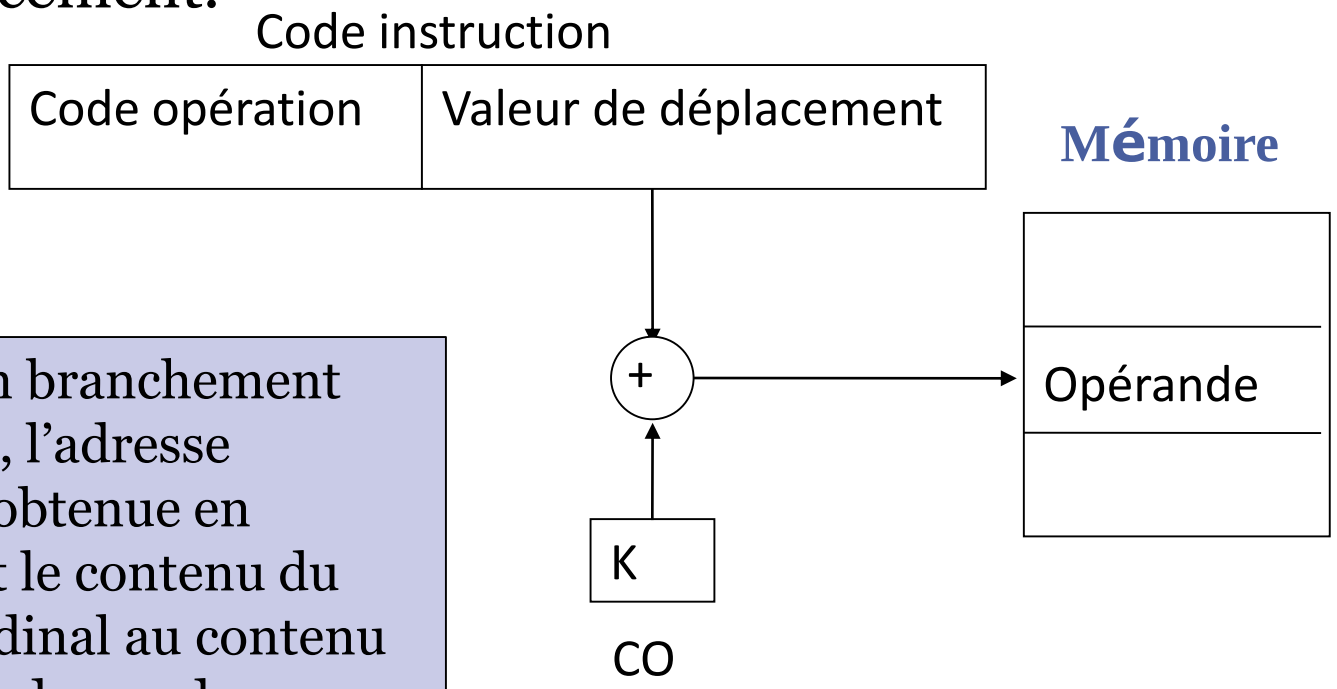
- Dans le mode d'adressage indirect par registre avec déplacement, le spécificateur d'opérande comprend l'adresse d'un registre de base ainsi que la valeur de déplacement.



Ex : Cette technique permet de réduire le nombre de bits dans le champ adresse : le registre de base contient la première adresse d'un bloc de 2^k mots et l'adresse (sur k bits) contenue dans l'instruction représente le déplacement à l'intérieur du bloc.

Adressage relatif

- Dans le mode d'adressage relatif, le spécificateur d'opérande comprend l'adresse du compteur ordinal ainsi que la valeur de déplacement.



Ex : Dans un branchement conditionnel, l'adresse effective est obtenue en additionnant le contenu du compteur ordinal au contenu du champ d'adresse de l'instruction.

Adressage indexé

- Ce mode d'adressage est très bien utile lorsqu'on travaille, par exemple, sur des tableaux.
- Considérons un bloc de n mots consécutifs débutant à l'adresse A, le $j^{ième}$ mot se trouve à l'adresse A+ (j-1).
- Pour référencer ce mot, il est possible d'utiliser un registre **d'index**.
- L'adresse effective est calculée en additionnant le contenu de ce registre d'index à l'adresse qui se trouve dans le champ adresse de l'instruction.
- Sur certaines machines, tous les registres généraux peuvent être utilisés comme registres d'index. La présence d'un registre d'index s'accompagne généralement de la possibilité d'incrémentation et décrémentation automatiques.

Adressage indexé