

# **Architecture des Ordinateurs**

**MUSTAPHA JOHRI**

**EST - Béni-Mellal**

**Mémoires & microprocesseurs**

# Mémoires

# Introduction

- Pour mémoriser une information sur 1 seul bit, on peut utiliser une bascule.
- Pour mémoriser une information sur  $n$  bits, on peut utiliser un registre.

**Question** : Si on veut mémoriser une information de taille importante, alors comment peut-on le faire ?

# Mémoire

Une mémoire est un dispositif capable :

- d'enregistrer une information,
- de la conserver (mémoriser),
- de la restituer (possible de la lire ou la récupérer par la suite).

# Caractéristiques de la mémoire

La performance globale d'un ordinateur est fortement influencée par la quantité et le type de mémoire qu'il possède. En général, on caractérise la mémoire selon les critères suivants :

## 1. Volatilité :

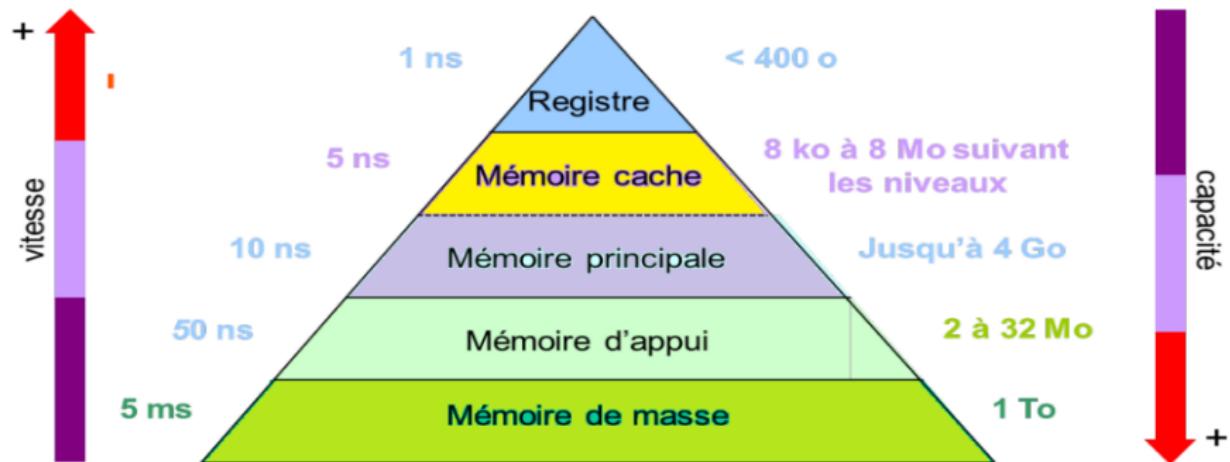
- La mémoire volatile (ou vive) perd son contenu lorsque l'ordinateur est éteint, comme la RAM.
- La mémoire non volatile conserve l'information en tout temps, comme la ROM.

## 2. Vitesse et temps d'accès.

## 3. Modes de stockage :

- Le stockage primaire est directement accessible par le microprocesseur, par exemple la mémoire vive (RAM).
- Le stockage secondaire, tel que le disque dur, n'est pas directement accessible par le microprocesseur.

# Hiérarchie des mémoires



## Hiérarchie des mémoires

- Les registres : Ce sont les éléments de mémoire les plus rapides, situés au niveau du processeur, et ils servent au stockage des opérandes et des résultats intermédiaires.
- La mémoire cache : C'est une mémoire rapide de faible capacité destinée à accélérer l'accès à la mémoire centrale en stockant les données les plus utilisées.
- La mémoire principale : C'est l'organe principal de stockage des informations. Elle contient les programmes (instructions et données) et est plus lente que les deux mémoires précédentes.
- La mémoire tampon : Elle sert de mémoire intermédiaire entre la mémoire centrale et les mémoires de masse. Elle remplit le même rôle que la mémoire cache.
- La mémoire de masse : C'est une mémoire périphérique de grande capacité utilisée pour le stockage permanent ou la sauvegarde des informations.

# Représentation de l'information et stockage en mémoire

Soit le programme C suivant :

```
1 #include <stdio.h>
2
3 int main() {
4     int a = 11, b = 315;
5     double c = 3.4;
6     char d;
7     return 0;
8 }
```

Le problème abordé ici concerne l'emplacement et la manière dont les informations associées aux variables a, b, c et d seront stockées.

# Stockage en mémoire : Bascule

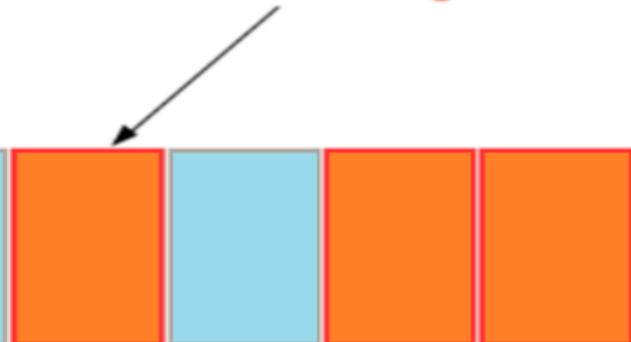
Une bascule peut être :

- chargée : bit à 1
- non chargée : bit à 0

**Bascule non chargée**

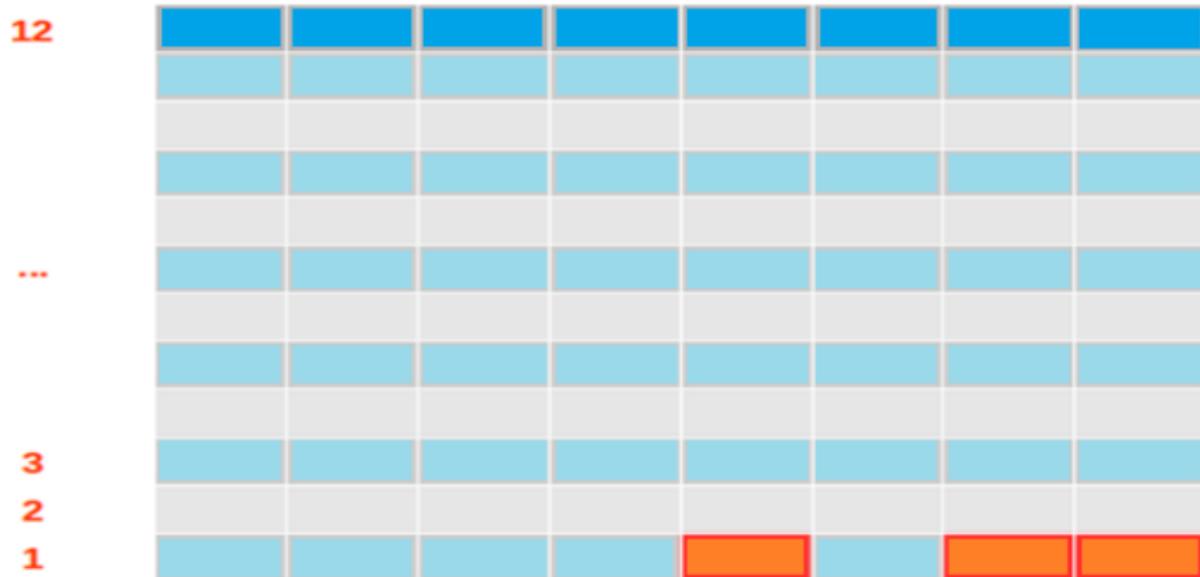


**Bascule chargée**



## Stockage en mémoire : sur 1 emplacement

Retournons à notre code. En utilisant une représentation sur 8 bits, nous pouvons exprimer "a" comme suit :  $a = (11)_{10} = (1011)_2 = (00001011)_2$ .

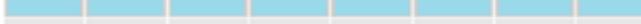


la valeur de a est stockée à l'adresse 1

## Stockage en mémoire : adresse

Pour le programme C précédent, comment peut-on chercher la valeur de la variable *a*? Comment l'ordinateur parvient-il à identifier quelles bascules contiennent quelles informations ?

C'est grâce à l'adresse que l'ordinateur arrive à se repérer dans la mémoire. Lorsqu'il stocke une donnée, il la met à une adresse précise.

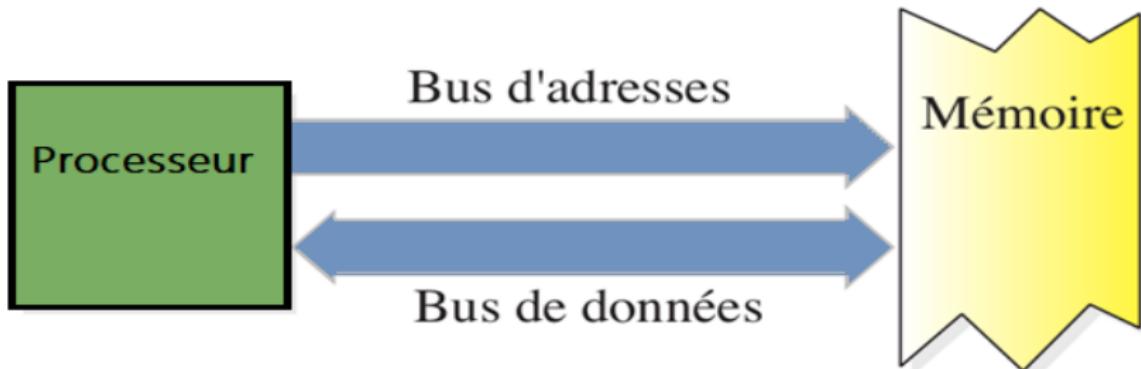
Adresse	Emplacement
7 = 111	
6 = 110	
5 = 101	
4 = 100	
3 = 011	
2 = 010	
1 = 001	
0 = 000	

↑                      ↓

Bit de poids fort      Bit de poids faible

## Stockage en mémoire : bus d'adresses / bus de données

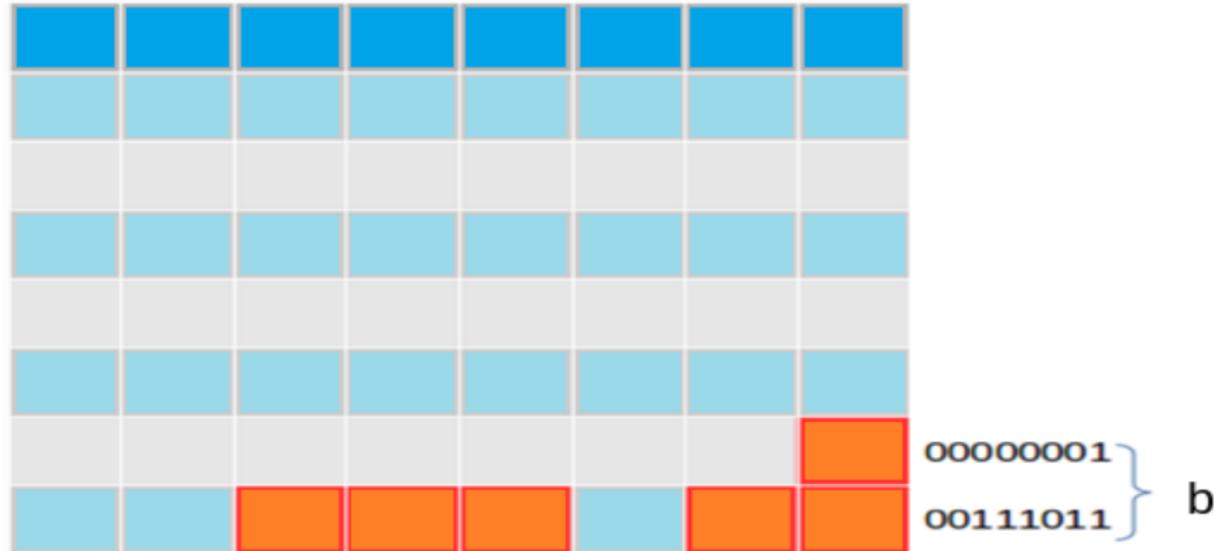
- Le processeur récupère de la mémoire centrale les instructions et les données nécessaires pour accomplir une certaine tâche.
- Chaque élément d'information est identifié par une adresse spécifique, transmise par le processeur via le bus d'adresses.
- Le transfert des données entre la mémoire et le processeur se fait ensuite à travers le bus de données.



## Stockage en mémoire : sur plusieurs emplacements

- Nous avons vu comment stocker  $a = 1110$  en mémoire.
- Maintenant, comment stocker  $b = 315$  ?
- Remarque :  $b$  dépasse 255 : la taille de  $b$  dépasse 1 octet.
- Idée : Stocker  $b$  sur plusieurs emplacements mémoire consécutifs.
- En effet,  $b = (315)_{10} = (100111011)_2 = 00000001\ 00111011$ .
- Il faudra donc deux emplacements pour  $b$ .

## Stockage en mémoire : sur plusieurs emplacements



# Types de données fondamentaux

- Les ordinateurs modernes couramment utilisent des données de 8, 16, 32 ou 64 bits, bien que d'autres tailles soient également envisageables.
- La nomenclature standardisée par les éditeurs de langages de développement logiciel est la suivante :

bit (*bit*)  1

octet (*byte*)  8

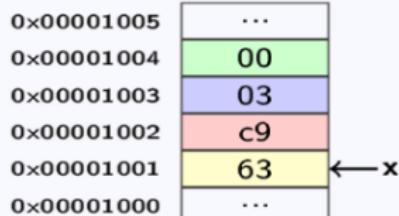
mot (*word*)  16

double mot (*d-word*)  .....  32

quad-mot (*q-word*)  .....  64

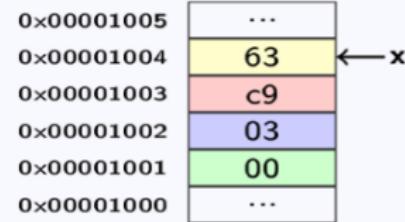
# Ordre d'empilement lors du stockage sur plusieurs octets : Boutisme

$$248163_{10} = 111100100101100011_2$$



**little-endian**

lsb (*Less Significant Bit*) à  
l'adresse la plus petite



**big-endian**

msb (*Most Significant Bit*)  
à l'adresse la plus petite

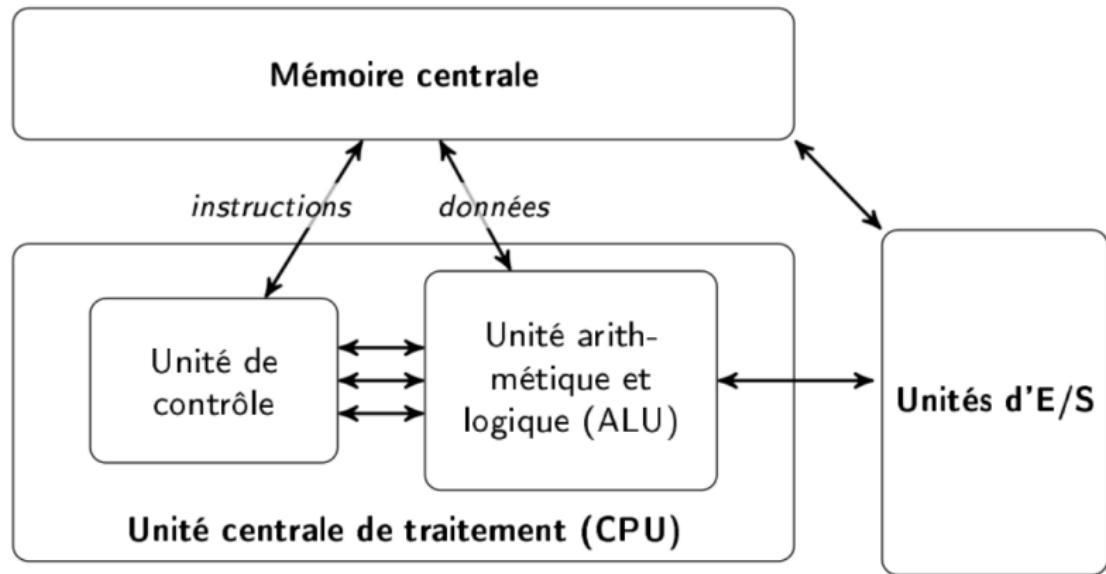
# Processeurs

# Processeur

Le processuer (micro-processeur), ou CPU (Central Processing Unit), est le cerveau de l'ordinateur, responsable des actions suivantes :

- Lecture d'instruction : Le processeur lit une instruction de la mémoire.
- Interprétation d'instruction : L'instruction est décodée pour déterminer l'action à effectuer.
- Lecture des données : Le CPU récupère des données depuis la mémoire ou un périphérique d'entrée/sortie.
- Traitement des données : Le CPU effectue des opérations arithmétiques ou logiques sur les données.
- Écriture des données : Les données sont enregistrées dans la mémoire ou dans un périphérique.

# Processeur

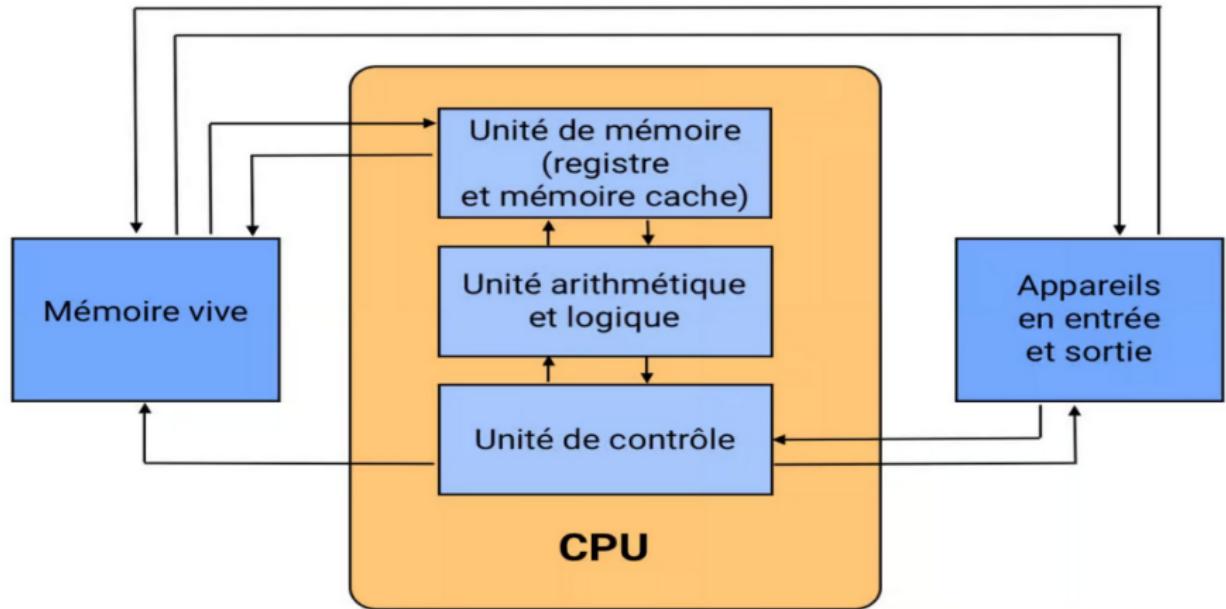


# Processeur

Un processeur est composé essentiellement de quatre éléments :

- Unité de contrôle ou séquenceur : Responsable de la gestion globale du processeur, cette unité peut décoder les instructions, sélectionner les registres à utiliser, gérer les interruptions, ou initialiser les registres au démarrage.
- Unité arithmétique et logique (UAL) : Effectue les calculs et les opérations logiques nécessaires à l'exécution des différentes instructions.
- Registres : Ce sont des mémoires internes destinées à stocker temporairement les données et les instructions en cours de traitement.
- Bus interne : Il assure l'interconnexion entre ces différents composants du processeur.

# Processeur



# Familles de microprocesseurs

Les familles de microprocesseurs sont définies par le jeu d'instructions qu'ils peuvent exécuter, c'est-à-dire l'ensemble des opérations qu'ils peuvent effectuer. Ainsi, une famille de microprocesseurs regroupe tous les processeurs capables de traiter le même ensemble d'instructions. Parmi les familles les plus courantes, on trouve :

## Intel :

- Intel Core (i3, i5, i7, i9)
- Intel Xeon
- Intel Atom
- Intel Pentium

## AMD :

- AMD Ryzen (3, 5, 7, 9)
- AMD EPYC
- AMD Athlon
- AMD FX

## ARM :

- ARM Cortex-A (A53, A55, A72,...)
- ARM Cortex-R
- ARM Cortex-M

**IBM POWER**

|

**NVIDIA Grace**

# Architecture des microprocesseurs

L'architecture des microprocesseurs se divise en deux grandes familles :

- L'architecture CISC (Complex Instruction Set Computer) : Intel x86, AMD x86, ...
- L'architecture RISC (Reduced Instruction Set Computer) : ARM, PowerPC, MIPS, RISC-V, ...

# L'architecture CISC

- Les microprocesseurs CISC se caractérisent par un jeu d'instructions plus complexe, comprenant souvent plus de 200 instructions.
- L'objectif de cette diversité d'instructions complexes est de simplifier la programmation en langage d'assemblage.
- Cependant, cette approche de programmation nécessite un câblage interne complexe, ce qui peut ralentir la vitesse globale du microprocesseur.
- Avantage : Développement plus simple.
- Inconvénient : Temps d'exécution plus long des instructions.

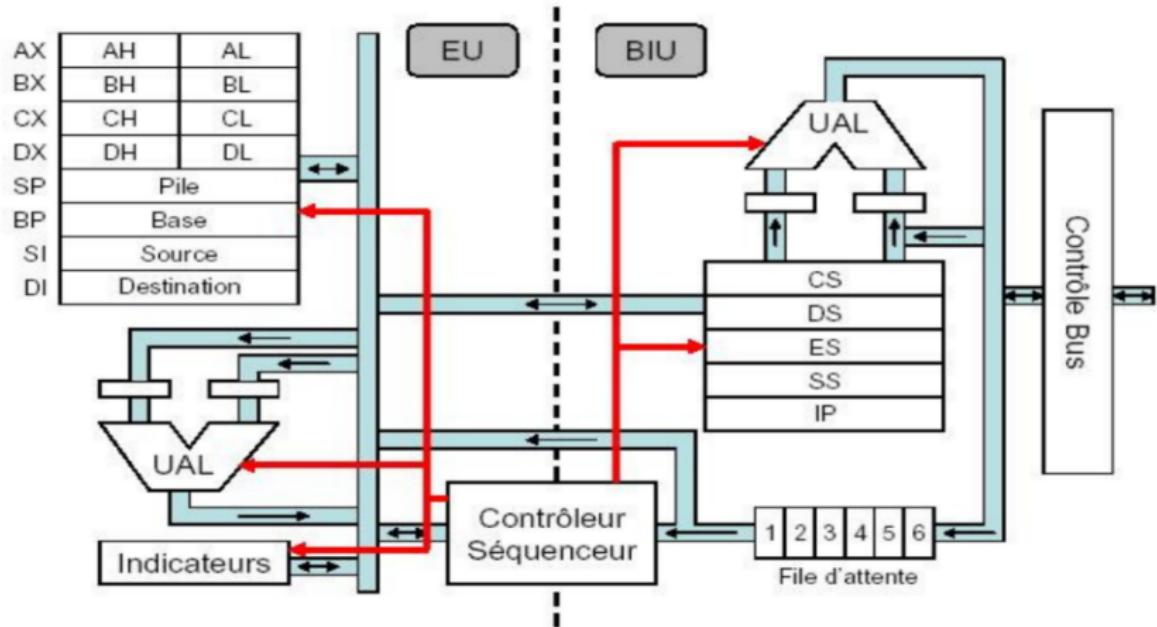
# L'architecture RISC

- Idée de base : En général, 80% des traitements des langages de haut niveau font appel à seulement 20% des instructions du microprocesseur.
- Ainsi, les processeurs RISC ont un jeu d'instructions plus simple et réduit (moins de 100 instructions).
- Les caractéristiques de l'architecture RISC peuvent être résumées comme suit :
  - le nombre d'instructions en langage assembleur est réduit, ce qui permet un câblage interne plus rapide.
  - Réduction du jeu d'instructions aux instructions les plus couramment utilisées.
  - Amélioration de la vitesse de traitement.
- Avec cette architecture, l'exécution devient plus rapide, mais cela nécessite un compilateur plus complexe.

# Microprocesseur 8086

- Le microprocesseur 8086 se présente sous la forme d'un boîtier contenant des circuits intégrés qui dispose de sorties pour se connecter à un environnement externe.
- Comme tout CPU, le 8086 dispose d'un certain nombre de registres.

# Microprocesseur 8086



# Microprocesseur 8086

- L'unité d'interface de bus (BIU) recherche les instructions en mémoire et les range dans une file d'attente.
- L'unité d'exécution (EU) exécute les instructions contenues dans la file d'attente.
- Les deux unités fonctionnent simultanément.

# Registres du CPU 8086

- Un registre est un composant de mémoire interne au microprocesseur dont sa capacité est calibrée sur une taille des instructions.
- Le rôle du registre est de stocker des adresses ou des résultats intermédiaires des calculs en cours.
- Pour manipuler les registres, on utilise "les commandes" de l'assembleur.

# Registres du CPU 8086

Le CPU 8086 comporte principalement :

- 3 groupes de 4 registres de 16 bits
- 1 registre d'état (FLAGS)
- 1 compteur programme de 16 bits : Pointeur d'Instruction (IP).

# Registres du CPU 8086

## REGISTRES GENERAUX

AX	
AH	AL
BX	
BH	BL
CX	
CH	DL
DX	
DH	DL
DI	
SI	
SP	
BP	

Accumulateur

Base

Compteur

Données

Destination index

Source index

Pointeur de pile

Pointeur de base

## REGISTRE DE SEGMENT

DS
ES
CS
SS

Segment de données

Extra segment

Segment de code

Segment de pile

## COMTEUR DE PROGRAMME

IP
----

Pointeur d'instruction

## REGISTRE DE FLAG

0	D	I	T	S	Z	A	P	C
15	11	10	9	8	7	6	4	2

# Quelques Flags fréquemment utilisés

- **CF (Carry Flag)** : Vaut 1 si l'instruction exécutée provoque une retenue et 0 sinon.
- **PF (Parity Flag)** : Indique si le nombre de bits à 1 dans le résultat est pair.
- **ZF (Zero Flag)** : Vaut 1 si le résultat de l'opération est zéro.
- **SF (Sign Flag)** : Indique le signe du résultat de l'opération (1 pour négatif, 0 pour positif).
  - **OF (Overflow Flag)** : Indique le dépassement lors d'opérations arithmétiques signées.

# Fonctionnement d'un CPU : Cycle d'exécution d'une instruction

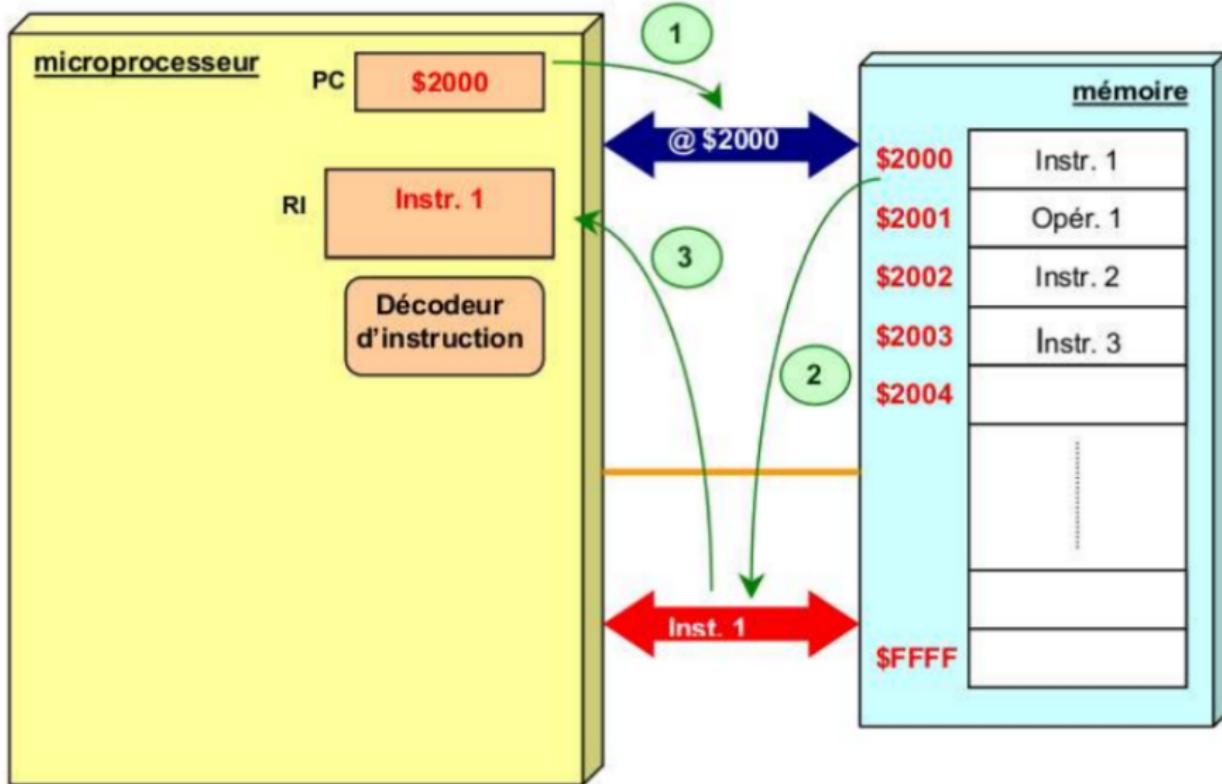
Le cycle d'exécution d'une instruction comprend principalement trois phases :

- Recherche de l'instruction
- Décodage de l'instruction
- Exécution de l'instruction

## Phase 1 : Recherche de l'instruction

- Le compteur ordinal (Program Counter PC) appelé aussi Pointeur d'Instruction IP contient l'adresse de l'instruction suivante du programme.
- L'unité de commande place cette adresse sur le bus d'adresses et émet un ordre de lecture.
- Après un certain temps correspondant au temps d'accès à la mémoire, le contenu de la case mémoire sélectionnée est disponible sur le bus de données.
- L'instruction est ensuite stockée dans le Registre d'Instructions RI du processeur.

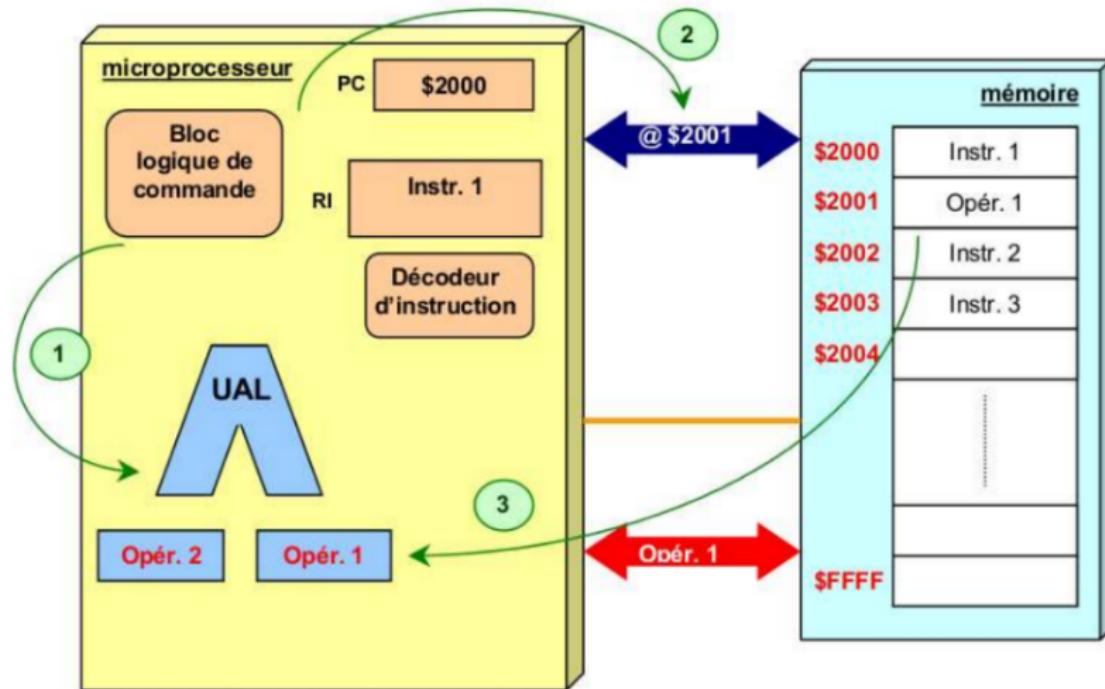
# Phase 1 : Recherche de l'instruction



## Phase 2 : Décoder l'instruction

- L'unité de contrôle identifie le type d'opération à effectuer.
- Les champs de l'instruction sont analysés pour déterminer les sources de données et où stocker le résultat.
- Des signaux de contrôle sont générés pour activer les composants nécessaires du processeur, afin d'exécuter l'opération spécifiée par l'instruction dans la phase suivante.

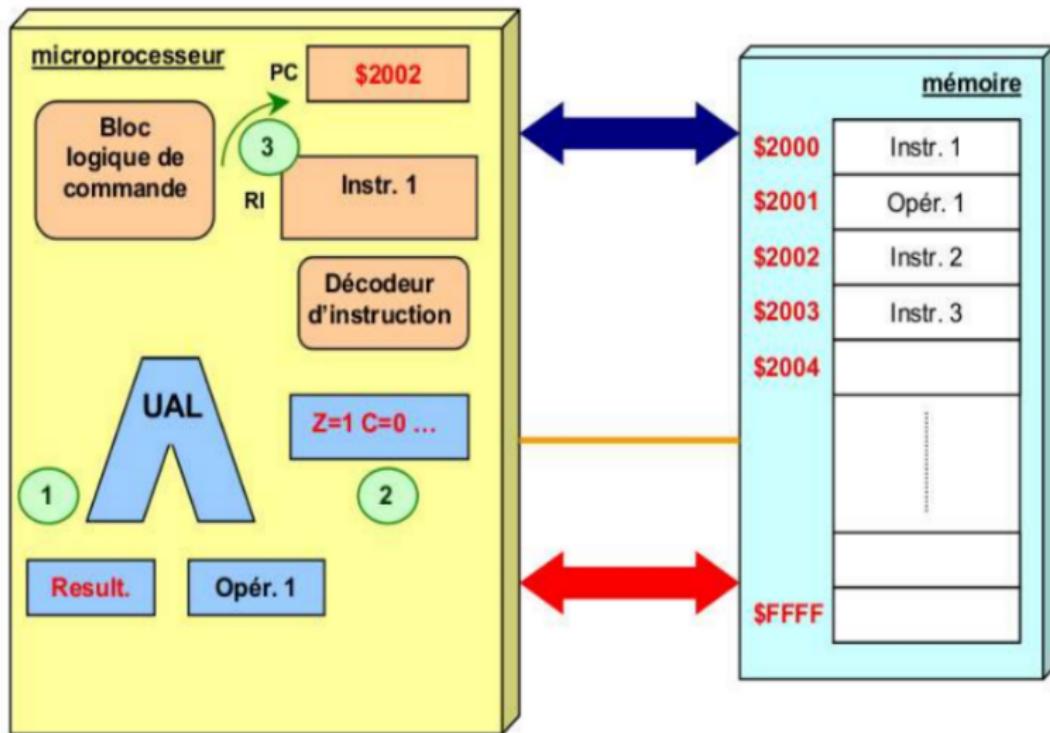
## Phase 2 : Décoder l'instruction



## Phase 3 : Exécuter l'instruction

- Pendant cette phase, l'instruction est exécutée.
- Ensuite, les drapeaux sont positionnés selon les résultats de l'exécution (voir registre d'état).
- Après cela, l'unité de commande met à jour le compteur ordinal (PC) pour passer à l'instruction suivante.

## Phase 3 : Exécuter l'instruction



## Puissance d'un microprocesseur

- Le microprocesseur est équipé d'une horloge interne qui génère des impulsions, appelées "top".
- La fréquence de ces impulsions, mesurée en Hertz (Hz), représente le nombre de tops par seconde.
- Il est courant que la fréquence de l'horloge interne du processeur soit un multiple de celle de la carte mère.
- À chaque top de l'horloge, le microprocesseur effectue une action, qu'il s'agisse de l'exécution d'une instruction complète ou d'une partie d'instruction.

# Puissance d'un microprocesseur

- Pour évaluer la puissance, on utilise le CPI (Cycle Par Instruction), indiquant le nombre moyen de cycles d'horloge nécessaires à l'exécution d'une instruction.
- L'unité couramment utilisée est le MIPS (Millions d'Instructions Par Seconde).
- La plupart des processeurs modernes ont une fréquence de fonctionnement supérieure à 1 GHz.
- Un processeur cadencé à 2 GHz peut exécuter jusqu'à 2 milliards d'opérations par seconde.

# Puissance d'un microprocesseur

- La puissance d'un microprocesseur se mesure par le nombre d'instructions qu'il peut traiter par seconde.
- Pour améliorer les performances d'un microprocesseur, on peut :
  - Augmenter la fréquence d'horloge (limitation matérielle).
  - Diminuer le CPI en choisissant un jeu d'instructions adapté.

# Microprocesseur et Mémoire cache

La mémoire cache stocke une copie des données d'origine pour éviter des accès coûteux en temps. La mémoire cache fonctionne selon les étapes suivantes :

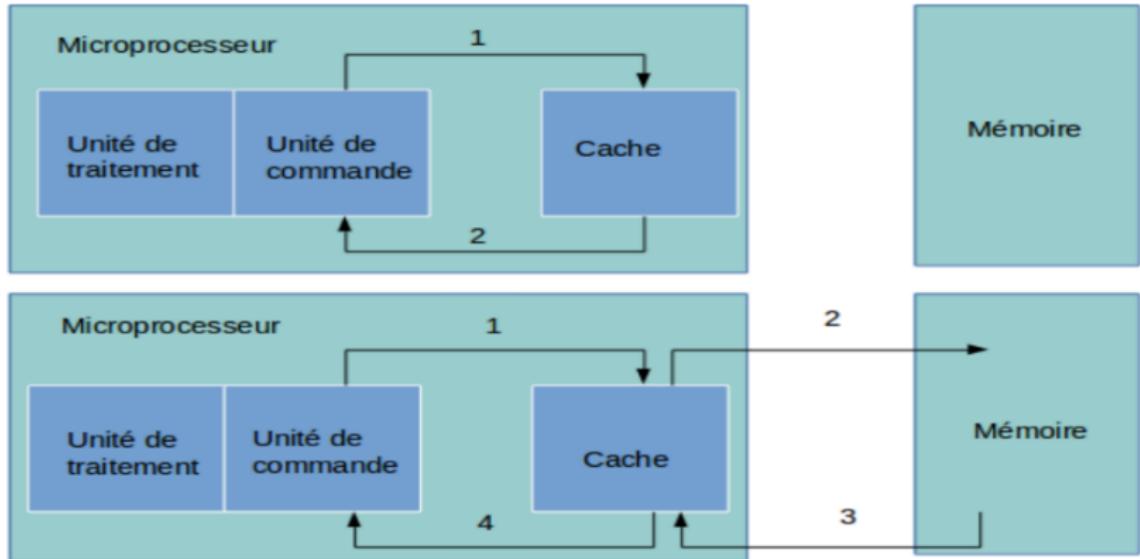
- Le microprocesseur initie une demande d'information.
- Le cache vérifie si cette information est présente :

Si elle est présente, le cache la transmet au microprocesseur, ce qui constitue un succès de cache.

Si elle n'est pas présente, le cache demande l'information à la mémoire principale, entraînant un défaut de cache.

- Le microprocesseur traite la demande et envoie la réponse au cache.
- Le cache stocke cette information pour une utilisation ultérieure et la transmet au microprocesseur selon les besoins.

# Microprocesseur et Mémoire cache



# Microprocesseur et Principe de localité

Pour améliorer les performances d'un microprocesseur, la mémoire cache exploite deux principes fondamentaux :

- Le principe de localité spatiale : Il suggère que l'accès à une donnée située à une adresse particulière  $x$  sera probablement suivi d'un accès à une zone mémoire très proche de  $x$ .
- Le principe de localité temporelle : Il indique que l'accès à une zone mémoire à un moment donné a de grandes chances de se répéter dans le programme dans un futur proche.