Ministère de l'Enseignement Supérieur Et de la Recherche Scientifique
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF
Faculté des Mathématiques et Informatique

Polycopié Pédagogique
Architecture des Ordinateurs
AO

Cours
Ingénieur-1-

Pr. FIZAZI Hadria
Université des sciences et de la technologie d'Oran

2025/2026

AVANT-PROPOS

Matière *: Architecture des ordinateurs*

Volumed'horairehebdomadaire *: (1h :30 x 14 semaines)*

Evaluation *: Contrôle continu : 40 % Examen :60 %*

Ce polycopié de cours porte sur l'architecture des ordinateurs il est destiné aux étudiants de première année ingénieur. Il vise à fournir aux étudiants une compréhension approfondie de l'architecture des ordinateurs, en combinant des aspects théoriques et pratiques.

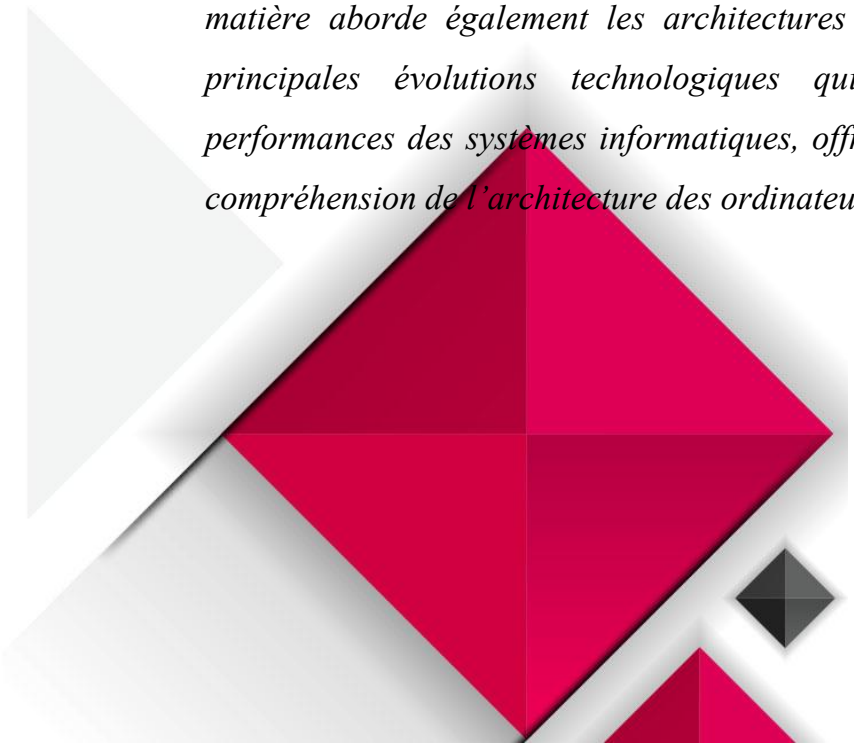
Les objectifs pédagogiques inclus la compréhension des fondamentaux de l'architecture des ordinateurs, la maîtrise de l'architecture interne des processeurs, l'analyse et étude de cas des processeurs spécifiques et l'exploration des architectures de processeurs modernes.

TABLE DES MATIÈRES

OBJECTIFS DE L'ENSEIGNEMENT

La matière a pour objectif de présenter de manière claire et structurée le principe de fonctionnement de l'ordinateur. Elle vise à permettre aux étudiants de comprendre comment un système informatique traite l'information en étudiant les différents composants matériels qui le constituent. Dans ce cadre, le cours propose une description détaillée de l'architecture de l'ordinateur, en expliquant le rôle et l'organisation des éléments essentiels tels que le microprocesseur, la mémoire centrale, les unités d'entrée et de sortie ainsi que les bus de communication qui assurent l'échange des données entre ces différents composants.

Pour faciliter la compréhension des concepts théoriques, l'étude s'appuie sur l'exemple d'un microprocesseur courant qui sert de référence pour illustrer l'organisation interne et le fonctionnement d'un processeur. Les étudiants découvrent ainsi les différentes unités fonctionnelles du processeur, le cycle d'exécution des instructions et les mécanismes de traitement de l'information. La matière aborde également les architectures de processeurs modernes et les principales évolutions technologiques qui ont permis d'améliorer les performances des systèmes informatiques, offrant ainsi une base solide pour la compréhension de l'architecture des ordinateurs actuels.



LISTES DES FIGURES

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 6

LISTES DES TABLEAUX

Chapitre 1

Chapitre 2

Chapitre 4

Chapitre 5

INTRODUCTION GÉNÉRALE

L'architecture des ordinateurs constitue l'un des domaines fondamentaux de l'informatique et de l'ingénierie informatique. Elle permet de comprendre comment les systèmes informatiques sont conçus, organisés et comment leurs différents composants matériels collaborent pour assurer le traitement automatique de l'information. La connaissance de l'architecture d'un ordinateur est essentielle pour les étudiants en ingénierie, car elle constitue la base de nombreuses disciplines. Dans ce contexte, ce polycopié de cours a pour objectif d'introduire les concepts fondamentaux liés à l'organisation et au fonctionnement interne des ordinateurs, tout en proposant une approche progressive qui combine des notions théoriques et des exemples pratiques afin de faciliter la compréhension des étudiants de première année ingénieur.

Le premier chapitre est consacré à l'organisation générale de l'unité centrale d'un ordinateur. Il débute par des généralités permettant de définir ce qu'est un ordinateur et d'expliquer son rôle dans le traitement et la gestion de l'information. Ce chapitre présente ensuite les principaux composants d'un ordinateur, notamment le processeur, la mémoire et les dispositifs d'entrée et de sortie. Enfin, il introduit deux modèles fondamentaux d'organisation des systèmes informatiques : l'architecture de Von Neumann et l'architecture de Harvard.

Le deuxième chapitre est consacré à l'étude de l'architecture interne des processeurs, considérés comme le cœur du fonctionnement de l'ordinateur. Il présente les principaux composants du processeur tels que l'unité de commande (Ucom) et l'unité arithmétique et logique (UAL), ainsi que l'unité mémoire et ses différentes caractéristiques, y compris le rôle de la mémoire cache. Le chapitre aborde également les registres du processeur, utilisés pour stocker temporairement les données et les instructions. Il traite aussi des bus de communication qui assurent l'échange des données et des signaux entre les différents composants du système. Enfin, il introduit la notion de jeu d'instructions, les modes d'adressage, les étapes d'exécution d'une instruction ainsi que les systèmes d'entrées/sorties permettant la communication avec les périphériques.

Le troisième chapitre introduit les notions relatives aux instructions d'un ordinateur et au langage machine. Il permet aux étudiants de comprendre comment les programmes sont représentés au niveau matériel et comment les instructions sont interprétées et exécutées par le processeur. Cette partie met en évidence le lien entre le matériel et le logiciel, en montrant comment les opérations programmées par l'utilisateur sont traduites en instructions compréhensibles par la machine. L'étude du langage machine permet ainsi d'avoir une vision plus concrète du fonctionnement interne de l'ordinateur et de la manière dont les programmes sont exécutés.

Le quatrième chapitre présente une étude de cas consacrée au processeur Intel 8086, qui constitue un exemple important dans l'histoire des microprocesseurs. L'analyse de son architecture permet de comprendre concrètement l'organisation interne d'un processeur, son mode de fonctionnement et la structure de son jeu d'instructions. Cette étude permet aux étudiants d'appliquer les notions théoriques vues dans les chapitres précédents à un exemple réel. Ainsi, ce polycopié vise à fournir une base solide pour la compréhension des principes fondamentaux de l'architecture des ordinateurs.

Le cinquième chapitre présente l'étude de cas d'un microprocesseur de 32 bits. Il est dédié à la présentation du microprocesseur MIPS R3000 en l'anglais Microprocessor without Interlockeed Pipeline Stage, conçu en 1985 par la société Computer System. Ce processeur gère des adresses mémoire de 32 bits, ce qui est idéal pour les anciens systèmes et applications. Il est utilisé surtout dans les systèmes embarqués comme les ordinateurs de poche, les consoles de jeux, les routeurs Cisco etc….

Le sixième chapitre est dédié à l'architecture des processeurs récents permettant de comprendre les nouvelles technologies. Contrairement aux premières générations de microprocesseurs comme Intel 8086, les processeurs modernes reposent sur des techniques avancées telles que le pipeline, l'exécution superscalaire, le multicœur et la gestion optimisée de la mémoire cache. Ces architectures modernes permettent d'exécuter plusieurs instructions simultanément et d'optimiser l'utilisation des ressources matérielles.

CHAPITRE-1-

Organisation générale de l'unité centrale d'un ordinateur

L'ordinateur est une machine complexe composée de plusieurs éléments matériels qui travaillent ensemble afin de traiter, stocker et transmettre l'information. Chaque composant possède un rôle précis et indispensable au bon fonctionnement du système informatique. Parmi ces composants, **l'unité centrale (UC)** occupe une place essentielle car elle constitue le cœur de l'ordinateur. Elle se présente sous la forme d'un boîtier qui regroupe les principaux éléments nécessaires au fonctionnement de la machine. On y trouve notamment la **carte mère,** qui représente le circuit principal reliant tous les composants, ainsi que le **processeur (CPU)** chargé d'exécuter les instructions des programmes. L'unité centrale contient également **la mémoire vive (RAM),** utilisée pour stocker temporairement les données en cours de traitement. Elle comprend aussi les dispositifs de stockage comme **le disque dur** ou le SSD pour conserver les informations. L'ensemble de ces composants communique grâce aux **bus de données** et permet d'assurer l'interaction avec les périphériques externes tels que l'écran et le clavier afin de traiter efficacement les informations.

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

1.1 Généralités sur l'Ordinateur

1.1.1 Qu'est-ce qu'un ordinateur ?

L'ordinateur est une machine électronique programmable servant au traitement del'information codée sous forme numérique. Il est capable de :

- Acquérir des informations (programmes et données)➔Fonction d'entrée
- Stocker et traiter ces informations ➔Fonction de Traitement
- Donner des résultats ➔ Fonction de sortie

1.1.2 Historique

- **En 1643** Blaise Pascal invente une machine mécanique à base roues dentées. Elle automatisait les opérations arithmétiques.

- **En 1883** le Britannique Charles Babbage réalise une machine analytique à base de roues dentelées permettant de lire les programmes sur un ruban perforé.

- **En 1940-1950** apparition des calculateurs électroniques.

- **1ère génération (1945–1955) :** Cette période correspond aux premiers ordinateurs électroniques utilisant des technologies à lampes, des relais, des tubes à vide et des résistances. Ces machines représentaient les premiers calculateurs électroniques et constituaient le début du développement des ordinateurs

- **En 1951** Von Neumann a mis en service la machine EDVAC (ElectronicDiscrete Variable AutomaticCompute) où les programmes et les données sont enregistrés en mémoire.

- **2ème génération (1955–1965) :** Cette génération se caractérise par l'introduction des ordinateurs utilisant des transistors, qui remplacent progressivement les tubes à vide. Cette évolution technologique a permis de réduire la taille des machines et d'améliorer leur fiabilité.

- **1955 IBM 650**premier ordinateur fabriqué en série

- **3ème génération (1965–1971) :** Les ordinateurs de cette période utilisent la technologie des circuits intégrés (puces) SSI et MSI, permettant d'intégrer un grand nombre de transistors sur une seule puce en silicium. Cette avancée a favorisé la miniaturisation des machines et l'amélioration de leurs performances. Elle a également marqué l'apparition des systèmes d'exploitation plus complexes.

- Le premier micro-ordinateur mis en marche c'est le Kenback 1 **en 1971**
- **4ème génération (1971–1977) :** Cette période est marquée par l'avènement de la microinformatique grâce à la technologie LSI (Large Scale Integration). Elle voit également le développement des premiers réseaux de machines, facilitant la communication entre ordinateurs. En 1976, la commercialisation de l'Apple I illustre l'émergence des ordinateurs personnels.
- **5ème génération (1977 et après) :** Cette période se caractérise par l'utilisation des technologies VLSI et ULSI (Very Large / Ultra Large Scale Integration), permettant d'intégrer des milliers à des milliards de transistors sur une seule puce. En 1981, IBM lance le PC (Personal Computer), popularisant l'informatique individuelle. À partir des années 1990, le développement des nanotechnologies et la miniaturisation des composants matériels ont permis d'améliorer considérablement les performances et la compacité des ordinateurs. **2007** iPhone mis en marche par Appel

1.1.3 Principaux Composants d'un ordinateur

Un ordinateur est composé d'une unité centrale (UC), des unités d'entrées/sorties (E/S) et une mémoire centrale (MC ou Main Memory).

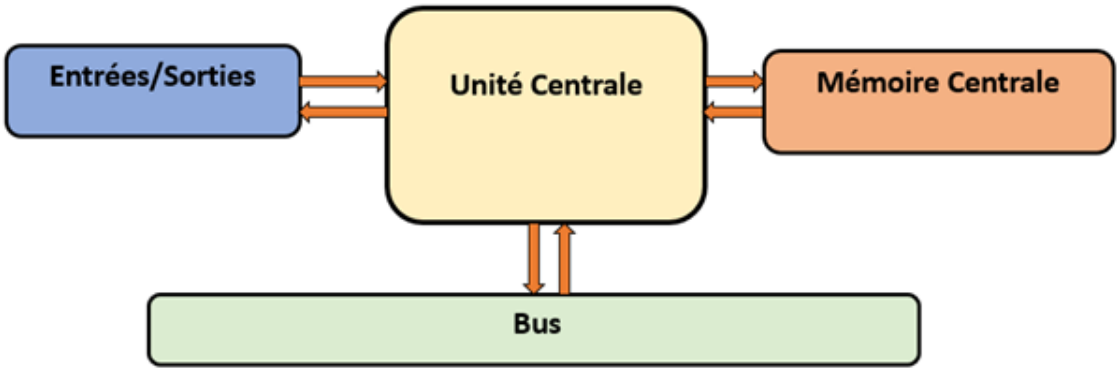


Figure 1.1 : Principaux Composants d'un Ordinateur

- **Une Centrale (UC)** est composée d'une unité de commande (UCom) et d'une unité detraitement ou unité arithmétique et logique (UAL). Donc UC = Ucom + UAL
- **Une mémoire centrale (MC ou Main Memory)** représente l'espace de travail de l'ordinateur. C'est l'organe principal de rangement des informations utilisées par le processeur. Elle contient les programmes (instructions et données).

- **Unités d'entrées/sorties (E/S)** désignent l'ensemble des transferts de données qui permettent au microprocesseur et à la mémoire de communiquer avec le reste du monde. Les Entrées/Sorties proviennent des périphériques.

1.2 Architecture de Base Machine de Von Newman et Machine d'Harvard

Il existe deux architectures informatiques, qui diffèrent dans la manière d'accéder aux mémoires :

- **L'Architecture de Von Neumann** : Les programmes et les données sont stockés dans la même mémoire et gérés par le même sous-système de traitement de l'information.

- **L'Architecture de Harvard** : Les programmes et les données sont stockés dans deux mémoires distinctes et gérés par différents sous-systèmes.

1.2.1 Architecture de Von Neumann :

L'architecture Von Neumann est un type d'architecture informatique dans laquelle l'unité centrale de traitement (CPU), la mémoire et les périphériques d'entrée/sortie (E/S) utilisent tous un seul bus partagé pour la communication. Cette architecture doit son nom à John von Neumann, pionnier du développement d'ordinateurs électroniques

John Von Neumann » un physicien et mathématicien d'origine Hongroise, propose 1946 un modèle d'ordinateur le EDVAC (ElectronicDiscrete Variable Automatic Computer). Il a introduit deux concepts dans le traitement de l'information :

- **Programme enregistré :**

Von Neumann a utilisé la mémoire pour stocker les programmes, d'où le nom de machine à programme enregistré. Rupture de séquence

- **Rupture de séquence :**

Von Neumann a automatisé les opérations de décision logique en munissant la machine d'une instruction appelée rupture de séquence conditionnelle ou branchement conditionnelle.

➢ Description de l'architecture de Von Neumann

L'architecture de Von Neumann se compose de quatre composants :

- **CPU :** L'unité centrale de traitement (processeur) est le cerveau de l'ordinateurcomposée d'une unité de contrôle et d'une unité arithmétique et logique. Il effectue tous les calculs et opérations nécessaires à l'exécution des instructions.
- **Mémoire :** La mémoire contient à la fois les instructions et les données que le processeur doit exécuter. La mémoire est organisée comme une séquence linéaire de cellules adressables, chacune contenant un nombre fixe de bits.
- **Bus système :** le bus système est un ensemble de fils qui connectent le processeur, la mémoire et les périphériques d'E/S. Il est utilisé pour transmettre des données, des instructions et des signaux de contrôle entre ces composants. On a **un bus de Contrôle** permet la communication avec les différents composants de l'ordinateur. **Un bus de données** (programme et données). **Un bus d'adresse** (adresse des programmes et des données)
- **Périphériques d'entrée/sortie (E/S) :** ces appareils sont utilisés pour communiquer avec le monde extérieur. Les exemples incluent les claviers, les écrans et les imprimantes

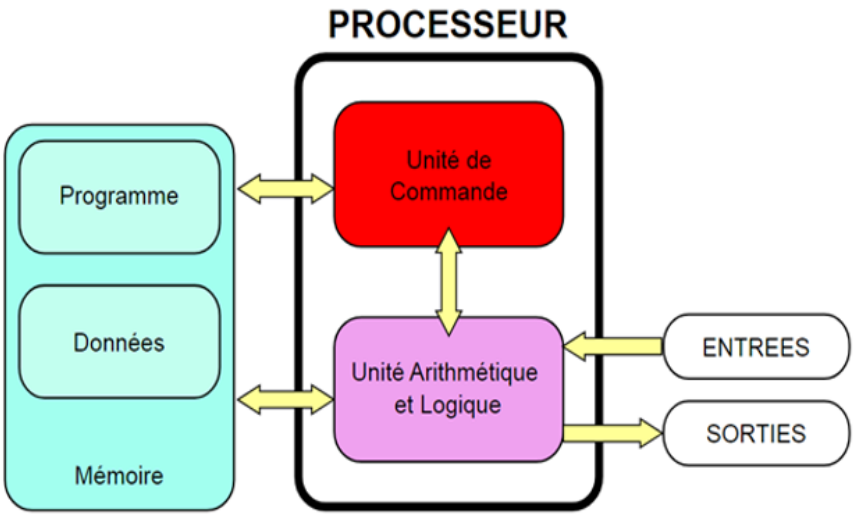


Figure 1.2 : Architecture de Von Neumann

1.2.2 Machine d'Harvard

L'architecture Harvard est une architecture informatique qui sépare physiquement la mémoire des instructions (programme) de la mémoire des données, utilisant des bus distincts pour chacune, ce qui permet au processeur de les lire simultanément pour une exécution plus rapide*ce qui peut améliorer les performances*.

Le nom de cette structure vient du nom de l'université Harvard où une telle architecture a été mise en pratique pour la première fois en 1930.

- **Description de l'architecture de Harvard**

Dans cette architecture, les mémoires de programmes et des données sont séparées : l'adressage de ces mémoires est indépendant.
Une architecture est constituée d'un bus de données, d'un bus de programme et de deux bus d'adresses.
mémoires, ce qui permet une grande souplesse pour l'enregistrement et l'utilisation des données.
Le nom de cette structure vient du nom de l'université de Harvard où une telle architecture a été mise en pratique pour la première fois et fut construit par IBM en 1944.
Il fut le premier ordinateur à utiliser des systèmes de mémoire séparés (des données et des instructions)

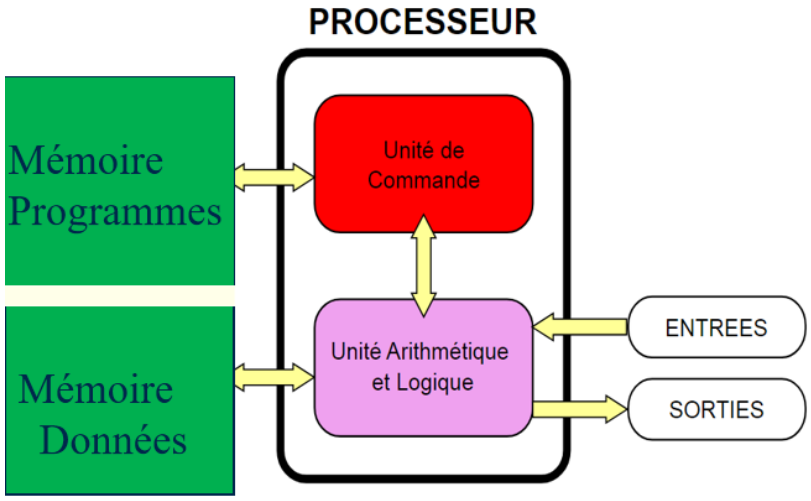


Figure1.3 : Architecture de Harvard

Remarques :

- Dans l'architecture de Von Neumann le processeur a besoin de deux cycles d'horloge pour exécuter une instruction, il lit d'abord l'instruction (mémoire programme) après il accède à la donnée (mémoire donnée) car il n'y a qu'une seule mémoire.
- Tandis que dans l'architecture de Harvard le processeur prend un cycle d'horloge pour compléter une instruction, il peut lire une instruction et accéder à la donnée en même temps car les deux mémoires sont séparées.
- Les PCs Modernes prennent les avantages des deux architectures
  Une mémoire principale pour les programmes et les données.
  Une mémoire cache pour les programmes et Une mémoire cache pour les données. On parle d'architecture Harvard Modifiée.

1.2.3 Avantages et Inconvénients des deux machines Von Newman et Harvard

Avantages	
Machine Von Newman	**Machine Harvard**
• Jeu d'instruction triche • Accès mémoire facile • Un seul bus de données *donc* facilite l'écriture de programmes et réduit la complexité du système dans son ensemble. • **L**e code peut être stocké dans la mémoir e des données pour un accès rapide	• Jeu d'instruction facile à mémoriser. • **C**odage d'instruction facile. Chaque instruction est codée sur un mot et dure un cycle. • **O**n peut charger les instructions et man ipuler les données en même temps. • bien adapté aux systèmes embarqués
Inconvenient	
Machine Von Newman	**Machine Harvard**
• Temps d'exécution d'une instruction variable • De plus, le processeur ne peut exécuter qu'une seule instruction à la fois, ce qui peut limiter la vitesse globale du système. • le bus partagé peut devenir un ralentissement des performances et une évolutivité réduite.	• Jeu d'instruction très pauvre. • Comme les mémoires d'instructions et de données sont séparées, il peut être plus difficile de partager des données entre différentes parties d'un programme.

Tableau 1.1 : Les avantages et inconvénients des deux machines

1.2.4 Différence entre l'architecture de Von Neumann et de Harvard

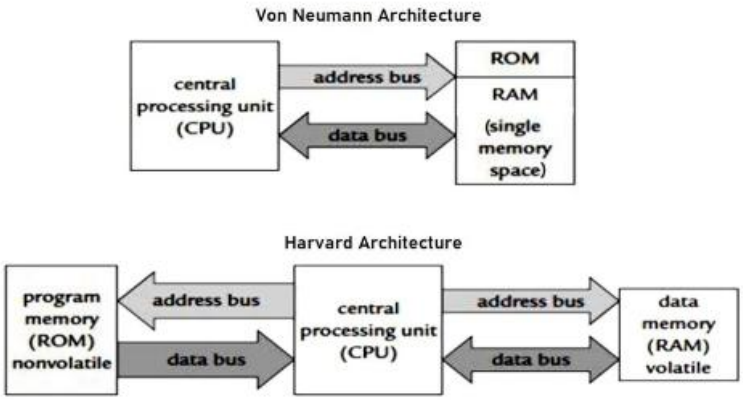


Figure 1.4 : Architecture de Von Neumann et Harvard

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

1.2.5 Comparaison entre l'architecture de Von Newman et l'architecture Harvard

Aspect	Architecture Von Neumann	Architecture de Harvard
Mémoire	Mémoire unique pour les instructions et les données	Mémoire séparée pour les instructions et les données
Accéder	Le processeur accède aux instructions et aux données via un bus partagé	Le processeur accède aux espaces de mémoire d'instructions et de données séparément
Performance	Peut devenir un goulot d'étranglement si trop d'appareils sont connectés au bus partagé	Performances améliorées car le processeur peut accéder simultanément à la mémoire d'instructions et de données
Modification	Il est plus facile de modifier les programmes car les instructions et les données sont stockées dans la même mémoire	Plus difficile de modifier les programmes qui nécessitent que le CPU modifie son propre code
Partage de données	Facile à partager des données entre différentes parties d'un programme	Plus difficile de partager des données entre différentes parties d'un programme
Applications	Convient à l'informatique à usage général où la flexibilité est requise	Convient aux systèmes embarqués où les performances sont critiques et où le code n'est pas fréquemment modifié

Tableau1.2 : Comparaison entre les deux architectures

CHAPITRE-2-

Architecture interne des processeurs

Le processeur, ou unité centrale de traitement (CPU), est le cœur de tout système informatique. C'est lui qui exécute les instructions des programmes et coordonne l'ensemble des opérations effectuées par l'ordinateur. Pour accomplir ces tâches, le processeur est constitué de plusieurs éléments internes, chacun jouant un rôle spécifique et essentiel dans le traitement des données.

Dans ce chapitre, nous allons présenter les composants principaux d'un ordinateur, tels que le processeur (CPU), la mémoire vive (RAM), les périphériques de stockage, et les différents périphériques d'entrée et de sortie.

L'objectif est de comprendre la fonction de chacun de ces éléments ainsi que la manière dont ils interagissent pour exécuter les tâches demandées par l'utilisateur.

2.1 PRINCIPAUX COMPOSANTS d'un ORDINATEUR (Ucom& UAL)

Nous allons définir et présenter les différents composants d'un ordinateur, en expliquant leur rôle et leur fonctionnement au sein du système.

Mémoire centrale ou mémoire principale

C'est une suite de "mots mémoire" ayant chacun une adresse. Elle contient des programmes et des données et une partie du système d'exploitation de l'ordinateur.

Unités d'entrées/sorties

Elles permettent de transférer des informations entre l'unité centrale et les unités périphériques. Comme unités périphériques on a les mémoires de masse (disques, bandes, ..) et autres unités (écran, clavier, imprimante, modem, etc...).

Bus d'adresse et bus de données :

Les différents organes du système sont reliés par des voies de communication appelés bus

Unité centrale (UC)

Elle contient une unité de traitement (UT ou UAL): et une unité decommande (UCom) .

Unité de traitement ou unité arithmétique et logique

Elle effectue les opérations des programmes et donc aussi des entrées-sorties de données avec la mémoire.

Unité de commande

Elle contrôle le fonctionnement de l'UAL, de la mémoire et des unités d'entrées/sorties (E/S), assurant la coordination de toutes les opérations internes du processeur. Elle est constituée de plusieurs composants essentiels, tels qu'un décodeur, un séquenceur, un compteur ordinal (CO), un registre d'instruction (RI) et plusieurs registres généraux. Ces éléments travaillent ensemble pour interpréter les instructions, gérer leur exécution et synchroniser le flux de données entre les différentes unités du processeur.

2.2 Rôle des Principaux Composants

2.2.1 Unité de Commande (Ucom)

Elle va chercher en mémoire, une par une, les instructions et les données. Elle décode chaque instruction et elle envoie un signal à l'UAL pour déclencher l'exécution de l'instruction.

❖ **Décodeur d'instructions**

- Il a en entrée le registre d'instruction. (RI)
- Le décodeur décode le champ du code opération du registre d'instruction
- Il indique au séquenceur quelle opération doit être effectuée

❖ **Séquenceur ou bloc logique de commande :**

- Il génère les signaux de commande pour déclencher et synchroniser l'exécution des différentes unités participant à l'exécution de l'instruction en cours.
- Il met à jour le Compteur ordinal (CO) en mettant l'adresse de la prochaine instruction.
- Lorsque les instructions sont exécutées séquentiellement, le compteur ordinal est augmenté de 1 sauf dans le cas où il y a un branchement/saut. Il met l'adresse de branchement dans le compteur ordinal.
  Ce dernier est alors utilisé pour chercher l'instruction en mémoire et la mettre dans le registre d'instruction **(RI).**

❖ **Compteur ordinal :**

- Il stocke l'adresse de l'instruction à exécuter. Sa taille est identique à celle d'uneadresse mémoire.

❖ **Registre d'instructions: (RI)**

- Il stocke l'instruction en cours d'exécution. Sa taille est égale à un mot mémoire.

Chaque instruction contient :

✓ **Un champ code-opération (Cop)**

Exemple : ADD = code de l'instruction d'addition

✓ **0 à 4 champs opérande dans une instruction.**

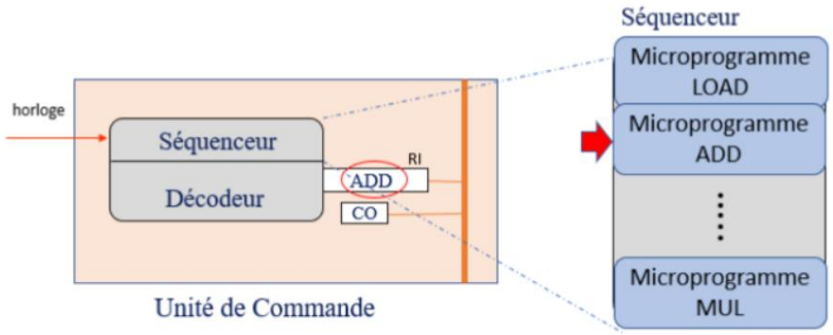


Figure 2.1 : Unité de commande

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

2.2.2 Unité Arithmétique et Logique ou unité de traitement

- Elle est composée de Trois registres appelés Accumulateurs :

  ✓ Deux servants pour stocker la donnée 1 et la donnée 2
  ✓ Le troisième pour stocker le résultat de l'opération à exécuter.

- Les opérations de bases effectuées par L'UAL sont :

  Les opérations arithmétiques : Additions, Soustractions, Division et Multiplication

  Les opérations logiques : OR, NOR, AND, NAND et XOR.

  Après chaque opération effectuée et selon le résultat obtenu, l'UAL met à jour le registre d'état. Ce dernier est composé de plusieurs bits où chaque bit est un indicateur d'états ou drapeau ou flag.

Comme indicateurs d'état on a :

S (Signe) : le bit de signe du résultat de la dernière opération.

Z (Zéro) : indicateur mis à 1 si le résultat obtenu est nul.

N (Négatif) : indicateur mis à 1 pour un résultat négatif.

C (Carry) : mis à 1 en cas de retenue ou débordement.

V (Overflow) : mis à 1 en cas de débordement.

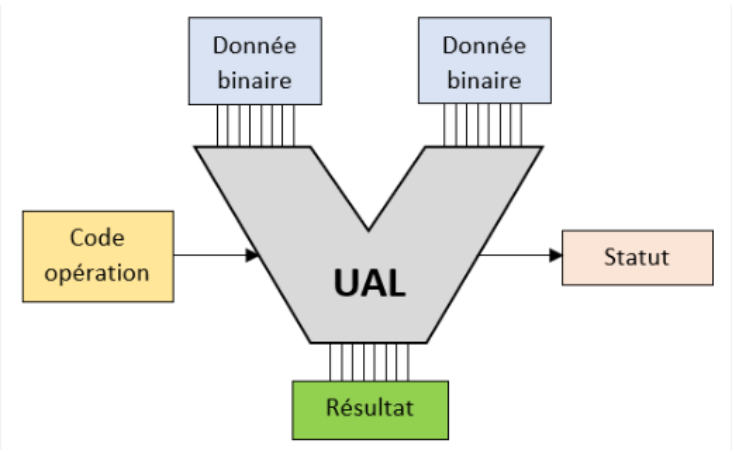


Figure 2.2 : Unité Arithmétique et Logique

Exemple : Les étapes d'addition de deux nombres

1) Chercher le premier nombre dans la mémoire et le placer dans l'accumulateur
2) Chercher le deuxième nombre dans la mémoire et le placer dans l'accumulateur 2
3) Une commande d'addition est délivrée au circuit d'addition via le bus interne de commandes.
4) Activation de l'additionneur avec les contenus des deux accumulateurs précédents comme sources
5) Stocker le résultat dans l'accumulateur 3
6) Sauver le résultat dans la mémoire
7) Fin de l'opération.

Exemple :

Supposant que les opérations arithmétiques de bases sont codées comme suite : 0000 pour l'Addition, 0001 pour la soustraction, 0010 pour la division et 0011 pour la multiplication.

0011	0111	0011	1001

Cette instruction veut dire : Faire la multiplication (0011) du contenu de l'adresse 0111 = 7 avec le contenu de l'adresse 0011 =3 et mettre le résultat dans l'adresse 1001=9.

Présentation des composants de l'UAL :

Parmi les composants de l'unité arithmétique et logique on peut citer :

- LesCircuits combinatoires tels que les Additionneurs, les Soustracteurs et les comparateurs,etc…

Les Circuits Combinatoires : Un circuit combinatoire est un circuit numérique dont les sorties **S** dépendent uniquement des entrées $\mathbf{E_i}$.

- $S_i=F(E_i)$
- $S_i=F(E_1, E_2,\ldots,E_n)$

Il est possible d'utiliser des circuits combinatoires pour réaliser d'autres circuits pluscomplexes.

1. Demi-Additionneur

Le demi additionneur est un circuit combinatoire qui permet de réaliser la sommearithmétique de deux nombres **A** et **B** sur un bit, à la sortie on va avoir la somme **S** et la retenue **R** (Carry).

- **Table de vérité du Demi-Additionneur :**

Entrées			Sorties		
A Cumulande	B Cumulateur		S somme	R retenus	Mintermes *Les « 1 »*
0	0		0	0	
0	1		1	0	$\bar{A}B$
1	0		1	0	$A\bar{B}$
1	1		0	1	AB

Tableau 2.1: Table de vérité(demi-additionneur)

Donc on aura :

$$R= AB$$
$$S = \bar{A}B + A\bar{B} = A \oplus B$$

- **Logigramme Demi Additionneur:**

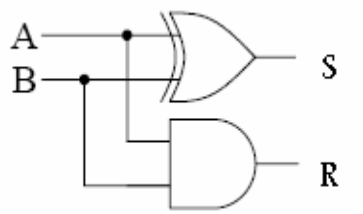


Figure 2.3 : Logigramme Demi-additionneur

2. L'additionneur complet.

La représentation de l'additionneur complet est donnée par le schéma suivant où

- R_i indique la retenue du rang **i** et
- R_{i-1} la retenue précédente du rang **(i-1)**

Remarque : En binaire lorsque on fait une addition il faut tenir en compte de la retenue entrante (R_i).
L'additionneur complet un bit possède 3 entrées :

- a_i **:** le premier nombre sur un bit.
- b_i **:** le deuxième nombre sur un bit.
- R_{i-1} **:** la retenue entrante sur un bit.

Il possède deux sorties :

- S_i **:** la somme
- R_i **:** la retenue sortante

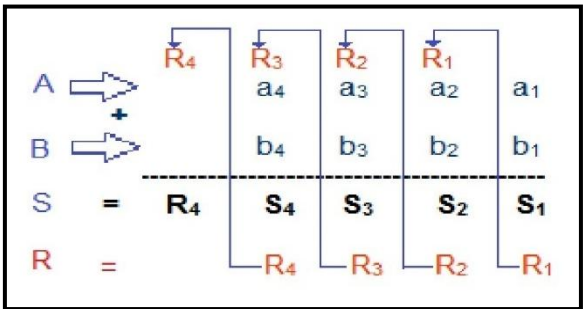


Figure 2.4 : Exemple d'un additionneur complet à 4 bits

- **Table de vérité d'un Additionneur complet :**

L'analyse du fonctionnement d'additionneur complet est illustrée par la table de vérité suivante :

A_i	B_i	R_{i-1}		S_i	R_i
0	0	0		0	0
0	0	1		1	0
0	1	0		1	0
0	1	1		0	1
1	0	0		1	0
1	0	1		0	1
1	1	0		0	1
1	1	1		1	1

Tableau 2.2 : Table de vérité(Additionneur Complet)

Les équations logiques des sorties S et Ri basées sur les mintermes:

$$S_i = \bar{A}_i\bar{B}_i\, R_{i-1} + \bar{A}_i\, B_i\, \bar{R}_{i-1} + A_i\bar{B}_i\, \bar{R}_{i-1} + A_i B_i\, R_{i-1}$$
$$= \bar{A}_i(\bar{B}_i\, R_{i-1} + B_i\, \bar{R}_{i-1}) + A_i(\bar{B}_i\, \bar{R}_{i-1} + B_i\, R_{i-1})$$
$$= R_{i-1} \oplus Ai \oplus Bi$$

$$R_i = \bar{A}_i B_i R_{i-1} + A_i\bar{B}_i\, R_{i-1} + A_i B_i\, \bar{R}_{i-1} + A_i B_i R_{i-1}$$
$$= \bar{A}_i B_i(R_{i-1} + R_{i-1}) + A_i B_i(\bar{R}_{i-1} + R_{i-1})$$
$$= A_i\, B_i + R_{i-1}\,(Ai \oplus Bi)$$

- **Logigramme d'un additionneur complet**

Logigramme de l'additionneur complet est illustré par la figure suivante :

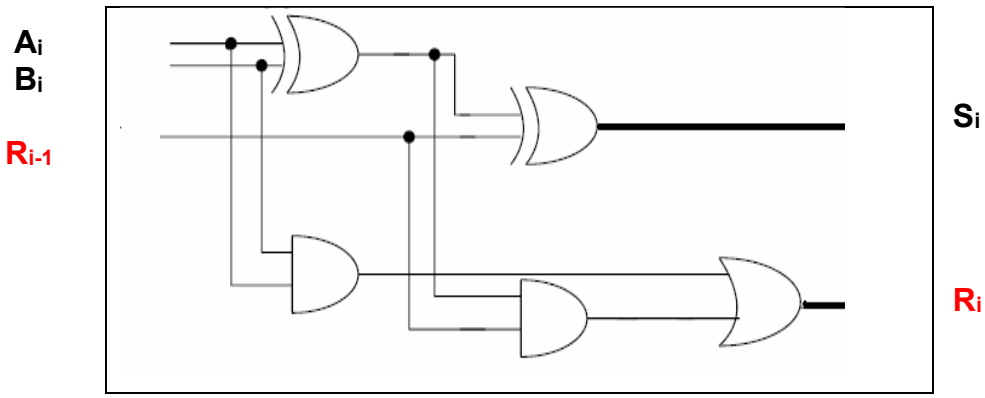


Figure 2.5 : Logigramme Additionneur Complet

3. Demi-soustracteur

Le soustracteur binaire portant sur un bit unique mène aux 4 cas présentés par la table de vérité suivante

- **Table de vérité du demi-soustracteur** :

A	B		D Différence	R retenue	Mintermes
0	0		0	0	
0	1		1	1	$\bar{A}B$
1	0		1	0	$A\bar{B}$
1	1		0	0	

Tableau 2.3: Table de vérité(demi-soustracteur)

Les équations logiques sont :

$$D = \bar{A}B + A\bar{B} = A \oplus B$$
$$R = \bar{A}B$$

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

- **Logigramme du demi-soustracteur (D.S) :**

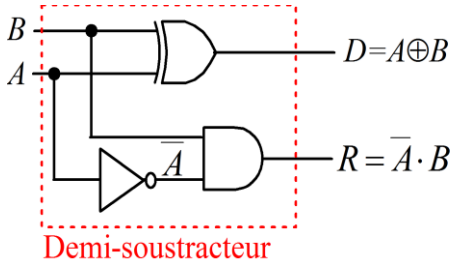


Figure 2.6 : Logigramme Demi-Soustracteur

4. Soustracteur complet

L'analyse de fonctionnement du soustracteur complet est illustrée par la table de vérité suivante :

- **Table du demi-soustracteur de vérité**

A	B	R_{i-1}		D	R_i
0	0	0		0	0
0	0	1		1	1
0	1	0		1	1
0	1	1		0	1
1	0	0		1	0
1	0	1		0	0
1	1	0		0	0
1	1	1		1	1

Tableau2.4 : Table de de vérité(demi-soustracteur)

Les équations logiques des sorties **D** et **Ri** en utilisant les mintermes :

$$D = \overline{A}\overline{B}R_{i-1} + \overline{A}B\overline{R_{i-1}} + A\overline{B}\overline{R_{i-1}} + ABR_{i-1}$$
$$= R_{i-1} \oplus A \oplus B$$

$$R_i = \overline{A}\overline{B}R_{i-1} + \overline{A}B\overline{R_{i-1}} + \overline{A}BR_{i-1} + ABR_{i-1}$$

Après simplification des équations précédentes on obtient :

$$D = R_{i-1} \oplus A \oplus B$$
$$R_i = \left(\overline{A \oplus B}\right).R_{i-1} + \overline{A}B$$

- **Logigramme du soustracteur complet :**

Logigramme du soustracteur complet est donnée par la figure suivante :

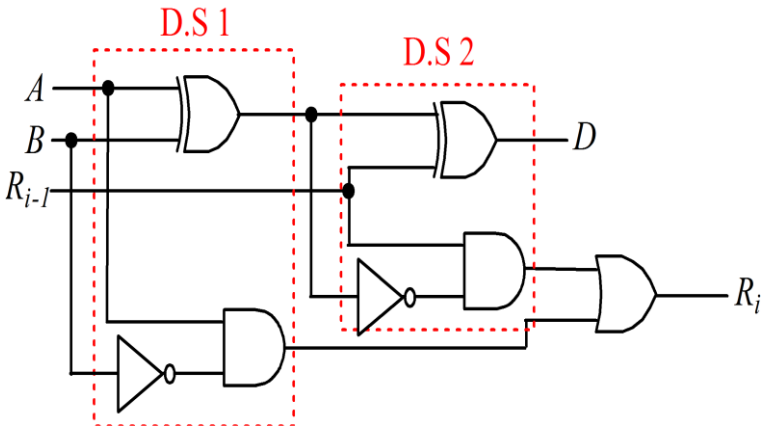


Figure 2.7 : Logigramme du soustracteur complet

5. Comparateur

C'est un circuit combinatoire qui permet de comparer entre deux nombres binaire **A** et **B** de **n** bits avec **n** égal à**1** bit ou plus. La représentation du comparateur entre 2 nombres A et B est donnée par lafigure 1.

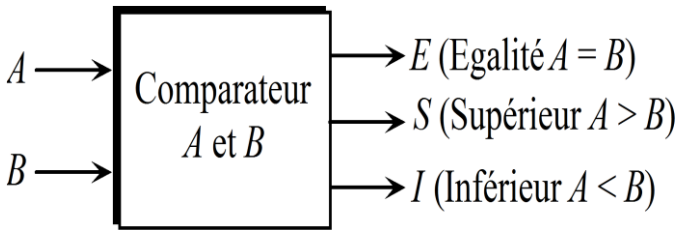


Figure 2.8 : comparateur de 2 nombres A et B

Donc un comparateur possède **2** entrées :

- **A**
- **B**

Et il possède **3** sorties

- **E** : **E**galité (**A=B**)
- **I** : **I**nférieur (**A< B**)
- **S** : **S**upérieur (**A > B**)

6. Comparateur sur un bit

Le comparateur à 1bit est un circuit combinatoire dont les entrées A et B sont sur 1bit. Le fonctionnement de ce comparateur est donné parla table de vérité suivante :

- **Table de vérité du comparateur à 1bit**

A	B		E	I	S
0	0		1	0	0
0	1		0	1	0
1	0		0	0	1
1	1		1	0	0

Tableau2.5 : Table de de vérité(Comparateur à 1 bit)

Les équations caractéristiques des sorties **E, I** et **S** en considérant les mintermes sont comme suite :

$$E = \bar{A}\bar{B} + AB = \overline{A \oplus B}$$

$$I = \bar{A}B$$

$$s = A\bar{B}$$

Le logigramme correspondant au comparateur à 1 bit est donné par la figure 2 :

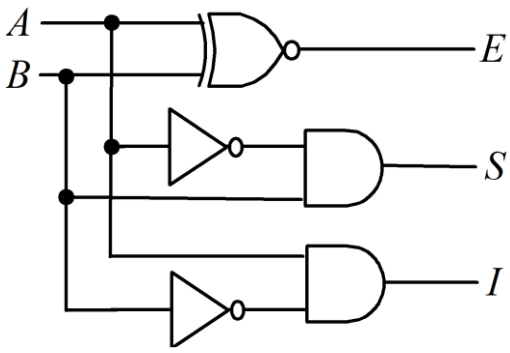


Figure 2.9 : logigramme du comparateur sur 1 bit

7. Comparateur sur 2 bits

Le comparateur sur 2 bitspermet de faire la comparaison entre deux nombres **A (a_2a_1)** et **B(b_2b_1)** chacun sur deux bits.

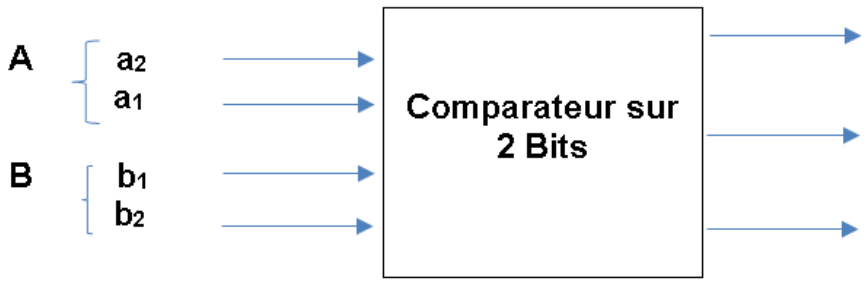


Figure 2.10 : comparateur des nombres A et B chacun sur 2 bits

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

2.3 Unité Mémoire

La mémoire centrale (MC) représente l'espace de travail de l'ordinateur. C'est l'organe principal de rangement des informations utilisées par le processeur. Dans un ordinateur, pour exécuter un programme il faut le charger (copier) dans la mémoire centrale.Le temps d'accès à la mémoire centrale et sa capacité sont deux éléments qui influent sur le temps d'exécution d'un programme

La mémoire centrale peut être vu comme un large vecteur (tableau) de mots ou octets. Un mot mémoire stocke une information sur n bits. Chaque mot possède sa propre adresse. La mémoire peut contenir des programmes et des données.

Adresse	programmés
0000	0001100
0001	0011100
0010	0001110
.....	

Partie instructions

Partie données

Figure 2.11 : Exemple de mémoire pour les adresse et programmes

2.3.1 Caractéristiques d'une mémoire

- **Capacité :**
  La Capacitéd'une mémoire représente la quantité d'informations qu'elle peut stocker.

$$\text{Capacité} = 2^{\text{taille de l'adresse}} \times \text{la taille du mot}$$

Elle est exprimée en :

Kilo-octet ➔ Ko = 1024 octets $=2^{10}$ octets
Méga-octet ➔ Mo = 1024 Ko $=2^{20}$ octets
Giga-octet➔Go = 1024 Mo $=2^{30}$ octets
Téra-octet➔ To = 1024 Go $=2^{40}$ octets

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

❖ **Volatilité** (inverse : permanente)

La Volatilité est le laps de temps durant lequel elle maintient les informations. Elle perd son contenu dès que l'on coupe le courant. Nécessite une alimentation continue

❖ **Temps d'accès**

Letemps d'accèsest letemps pour accéder à l'information pour la lecture/écriture d'un mot mémoire. Il correspond à l'intervalle de temps entre la demande de lecture/ écriture et la disponibilité de la donnée. De l'ordre de la nano-seconde(10 10^{-9}) pour les mémoires actuelles. De l'ordre de la milli-seconde pour les supports magnétiques.

❖ **Cycle mémoire :**

Le cycle mémoireest le temps minimal entre 2 accès mémoire ; temps d'accès + temps des opérations de maintien, stabilisation, synchronisation, etc…

❖ **Type d'accès**

- Accès direct à l'information, accès aux mots-mémoire par leur adresse.
- Accès séquentiel : c'est l'accès le plus long car pour accéder à une information il faut parcourir toutes celles qui la précèdent.

❖ **Débit :**

Le débitest le nombre d'informations lues ou écrites par unité de temps (seconde). Il est exprimé en bits par seconde : Débit = n / Tc (où n est le nombre de bits transférés par cycle et Tc= temps de cycle).

❖ **Le temps de cycle (Tc) :**

Le temps de cyclereprésente l'intervalle de temps minimum entre le lancement d'une opération de lecture ou d'écriture et sa fin (accomplissement).

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

Ce tableau ci-dessous présente une comparaison entre deux types fondamentaux de mémoire informatique : la mémoire vive (RAM) et la mémoire morte (ROM).

Mémoire vive: RAM	**Mémoire morte :** ROM
Mémoire volatile Cycle de lecture/écriture **RAM** Mémoire vive **SRAM** Mémoire vive statique **DRAM** Mémoire vive dynamique **SDRAM** Mémoire vive dynamique Synchrone	A lecture seule, Conserve l'information **ROM** : information stockée de manière définitive, **PROM** : Programmable par l'utilisateur **EPROM EEPROM** : Programmable + effacement par UV ou électriquement

Tableau 2.6: Comparaison entre la RAM et la ROM

La figure ci-dessous illustre un schéma simplifié du fonctionnement de la mémoire centrale dans un ordinateur.

- **Registre d'adresse** : Il contient l'adresse mémoire de la cellule à accéder (lecture ou écriture).
- **Mémoire centrale** : C'est l'ensemble des cellules mémoire où sont stockées les données et les instructions.
- **Registre de mot** : Il stocke temporairement les données lues depuis la mémoire ou celles à écrire dans la mémoire.

Les flèches montrent le flux d'informations : le registre d'adresse envoie l'adresse à la mémoire centrale, qui en retour transmet ou reçoit les données via le registre de mot. Ce mécanisme est essentiel pour accéder rapidement aux données en mémoire.

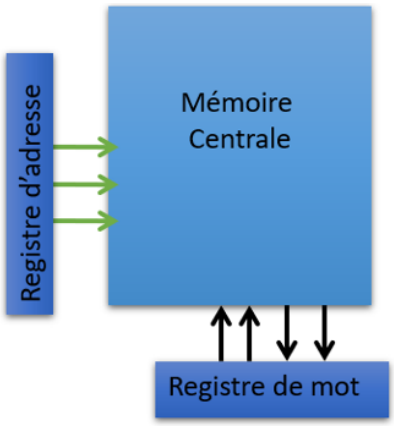


Figure 2.12 : Unité mémoire

2.3.2 Principe de la mémoire cache

- Le microprocesseur envoie au cache toutes ses requêtes comme s'il agissait de la mémoire principale :

- Soit la donnée ou l'instruction requise est présente dans le cache et elle est alors envoyée directement au microprocesseur.
  On parle de succès de cache (en anglais Hit).

- Soit la donnée ou l'instruction n'est pas dans le cache et le contrôleur de cache envoie alors une requête à la mémoire principale.

- Une fois l'information récupérée, il la renvoie au microprocesseur tout en la stockant dans le cache.
  On parle de défaut de cache (en anglais Miss)

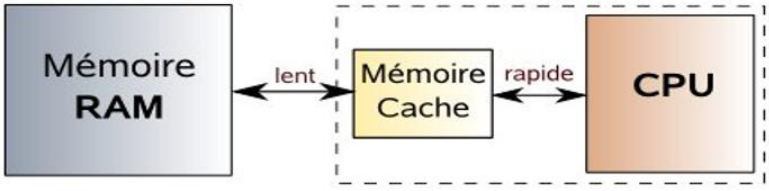


Figure 2.13.a : Exemple avec une mémoire Cache

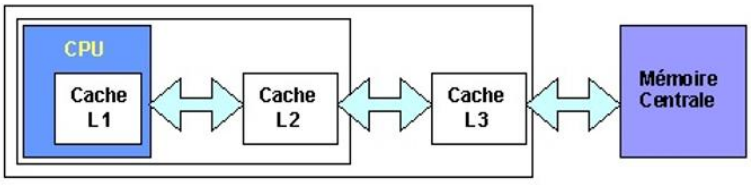


Figure 2.13.b : Exemple avec plusieurs mémoires Cache

Ce tableau ci-dessous présente les avantages et inconvénients de la mémoire cache en se basant sur les caractéristiques mentionnées.

Avantages	Inconvénients
• Elle est très rapide d'accès plus que la m émoire principale. • Elle consomme moins de temps d'accès par rapport à .la mémoire. • Elle stocke le programme qui peut être e xécuté dans un temps court. • Elle stocke les données pour une utilisati on temporaire.	• Elle a une capacité limitée • Elle est très coûteuse.

Tableau2.7 : Les avantages et inconvénients de la mémoire cache

Les ordinateurs utilisent différents types de mémoire pour stocker et gérer les données nécessaires à leur fonctionnement. Chaque type de mémoire a des caractéristiques spécifiques en termes de rapidité, capacité, volatilité et coût, qui influencent son rôle dans le système. La figure ci-dessous présentte les différentes types de mémoires.

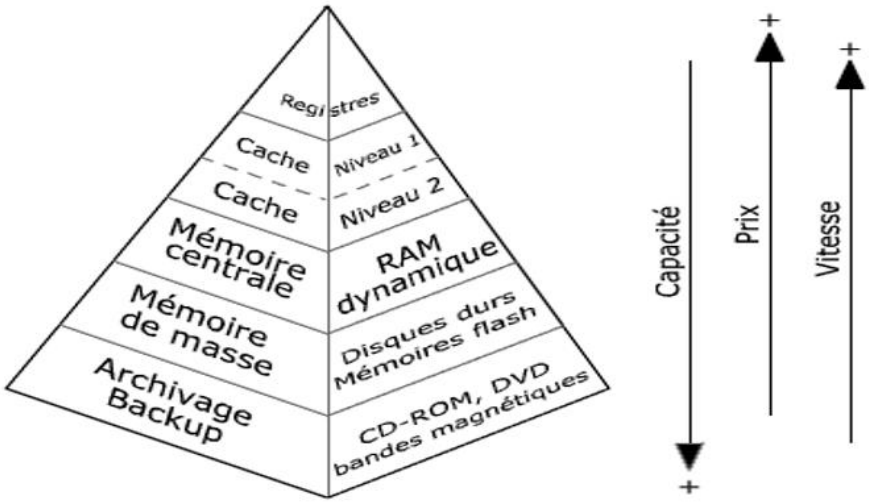


Figure 2.14 : Classification des mémoires

2.3.3 Catégories de mémoires

- **Les registres**sont les éléments de mémoire les plus rapides. Ils sont situés au niveau du processeur et servent au stockage des opérandes et des résultats intermédiaires.

- **La mémoire cache**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**est l'organe principal de rangement des informations. Elle contient les programmes (instructions et données) et est plus lente que les deux mémoires précédentes.
- **La mémoire d'appui**sert de mémoire intermédiaire entre la mémoire centrale et les mémoires de masse. Elle joue le même rôle que la mémoire cache.

- **La mémoire de masse**est une mémoire périphérique de grande capacité utilisée pour le stockage permanent ou la sauvegarde des informations.
  Elle utilise pour cela des supports magnétiques (disque dur, ZIP) ou optiques (CDROM,DVDROM)

2.4 Les registres

- Un registre permet la mémorisation de n bits. Il est donc constitué de n bascules, mémorisant chacune un bit.
- L'information est emmagasinée sur un signal de commande et elle est ensuite conservée et disponible en lecture.
- Le microprocesseur peut contenir d'autres registres autres que CO, RI et ACC. Ces registres sont considérés comme une mémoire interne (registre de travail) du microprocesseur.
- Ces registres sont plus rapides que la mémoire centrale, mais le nombre de ces registres est limité.
- Généralement ces registres sont utilisés pour sauvegarder les données avant l'exécution d'une opération.
- La taille d'un registre de travail est égale à la taille d'un mot mémoire.

2.4.1 Exemple de Registres relatives à l'UAL

Registres arithmétiques : pour les opérations arithmétiques ou logiques les plus utilisés :

- **Registres de base et d'index :** pour stocker la base ou bien l'index d'un tableau de données et ainsi calculer des adresses dans ce tableau.
- **Registre d'état ou registre condition** [Program Status Word (PSW)] chacun de ses bits est un "drapeau" [flag] qui indique un état.,
- **Registre pointeur** du sommet d'une pile [Stack Pointer (SP)]
- **Registres généraux :** pour diverses opérations, tel que : stocker des résultats intermédiaires
- **Registres spécialisés :** pour certaines opérations tel que les décalages [shift register], les opérations arithmétiques à virgule flottante [floating point register],

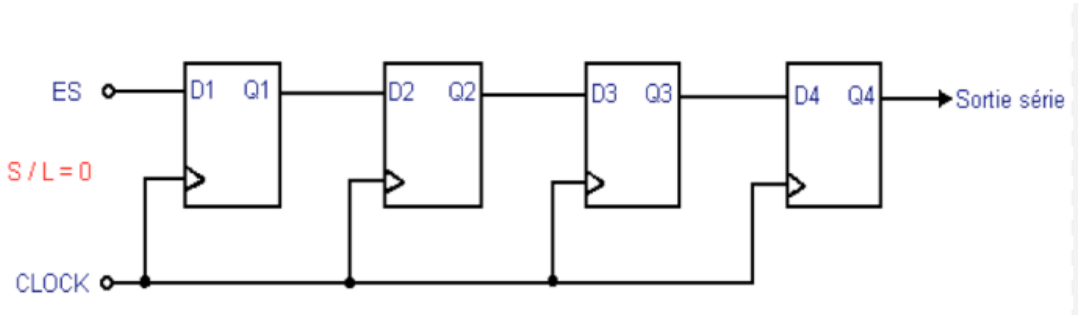


Figure 2.15 : Exemple d'un registre à 4bits à base de bascule D

- En synchronisme avec le signal d'écriture W (write) le registre mémorise les données présentent sur les entrées D1, D2, D3 et D4.
- Elles sont conservées jusqu'au prochain signal de commande W.
- Dans cet exemple les états mémorisés sont lus sur les sorties Q0, Q1, Q2, Q3 en coïncidence avec un signal de validation R (read)

2.5 Les bus

En informatique, on appelle BUS l'ensemble des liaisons physiques pouvant être exploités en commun par plusieurs éléments matériels afin de les faire communiquer. Donc un bus est un mécanisme qui permet le transfert de données entre éléments dans l'ordinateur.

2.5.1 Classification des bus de communications

On peut classer les différents bus en deux catégories : **les BUS internes** à l'unité centrale et **les BUS externes.**

A. BUS externes

Le BUS externe, aussi appelé BUS d'extension ou d'entrée/sortie, relie le microprocesseur aux périphériques d'entrée et de sortie, tels que l'écran, le clavier, la souris, etc…

Exemple de Bus externes

- Serial Advanced Technology Attachment (SATA) :série
- Mall Computer System Interface (SCSI) :parallèle
- Universal Serial Bus (USB) :série

B. BUS internes

Ils servent à connecter entre-eux les différents composants internes de l'unité centrale.

Exemple :de Bus internes

1. Peripheral Component Interconnect (PCI) : parallèle
- PCI express : série

2.5.2 Décomposition d'un bus de communication

On distingue trois catégories de Bus

- **Bus d'adresses (unidirectionnel)**

Il permet à l'unité de commande de transmettre les adresses à rechercher et à stocker.

- **Bus de données (bi-directionnel)**

Sur lequel circulent les instructions ou les données à traiter ou déjà traitées en vue de leur rangement.

- **Bus de contrôle (bi-directionnel)**

Transporte les ordres et les signaux de synchronisation provenant de l'unité de commande vers les divers organes de la machine. Il véhicule aussi les divers signaux de réponse des composants.

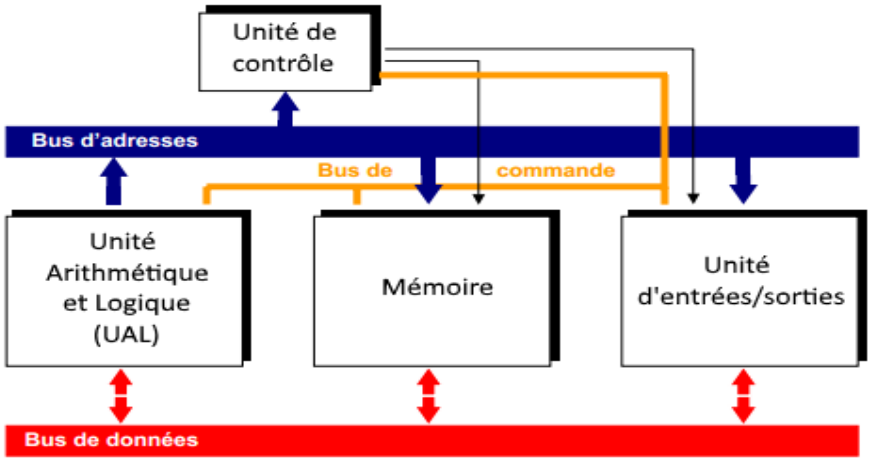


Figure 2.16 : Schéma des trois bus

2.5.3 Terminologie des bus d'un PC

Il existe des bus internes pour connecter des cartes d'extension. Ces bus sont appelés aussi **connecteurs**. On distingue :

1. Des connecteurs internes ou local
2. Des connecteurs externes appelés Input/Output Panel.

Exemple de Bus local : ISA et/ou PCI

- Industry Standard Architecture, Adressage 16 bits (64 ko), 8 MHz
- Peripheral Component Interconnect les plus récents,
- Ils ont une fréquence de 33 MHz
- Les différentes Vitesses de la carte mère sont : 66, 75, 83, 100, 133, 200 MHz

Exemple de Bus externes

IDE : Integrated Drive Electronics : connexion carte mère ⇔ contrôleur disque

SCSI : Small Computer System Interface : 7à14 périphériques, 8à16 bits, 10 Mb/s.

2.5.4 Structure d'un bus

- Voies de communication entre 2 ou plusieurs équipements
- Médium de communication partagé
- Signal émis par un équipement disponible en réception par équipements connectés
- Une seule émission à la fois
- Transmissions en parallèle

2.5.5 Caractéristiques d'un bus

1. **Types :**
   -Bus série : les données sont transmises les une à la suite des autres
   -Bus parallèle : plusieurs données sont transmises en parallèle
2. **Méthode d'arbitrage**
   -Solution centralisée par arbitre de bus
   -Chaque module contient sa logique d'accès : Solution distribuée
3. **Caractéristiques temporelles**
   -Synchrone
   -Asynchrone
4. **Nombre de transferts par seconde (T/s) :**

Il dépend de la fréquence (en Hertz) du signal d'horloge réelle du bus, c'est-à- dire le nombre de paquets de données envoyés ou reçus par seconde.

5. Calcul du débit (ou taux de transfert d'un bus)

Le débit maximal du bus (ou taux de transfert maximal), est la quantité de données qu'il peut transporter par unité de temps, en multipliant sa largeur par sa fréquence.

Débit = (nbre de transferts par seconde × largeur) / 8 octets/s

Exemple :

Un bus d'une largeur de 16 bits, cadencé à une fréquence de 133 MHz possède donc un débit égal à :

$$16 * 133.10^6 = 2128*10^6 \text{ bit/s, soit } 2128*10^6/8$$
$$= 266*10^6 \text{ octets/s soit 266 Mo/s}$$

6. Largeur de bus :

La largeur désigne le nombre de bits qu'un bus peut transmettre simultanément.

-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. On a :

Largeur du bus de données➔impact sur sa performance

Largeur du bus d'adresse➔ influe sur la capacité d'adressage

7. Taux de transfert de données ou Débit

Débit (octets/s) = [nombre de transferts par seconde * largeur] / 8.

A partir du débit on peut calculer le Taux de transfert ou la bande passante d'un bus:

Bande passante (en Mo/s) = largeur bus (en octets) * fréquence (en Hz)

8. Vitesse de transfert

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

Elle indique le nombre de paquets de données envoyés ou reçus par seconde. Elle est définie par sa fréquence exprimée en Hertz.

Exemple : Bus du processeur Pentium III

- **Bus de données**
  - 64 bits (8 chemins de 8 bits)
  - Taux de transfert de 1Go/s à 133 Mhz
  - Espace mémoire accessible de 64 Go
- **Bus d'Adresse**
  - Adresse 32 bits
  - Broche d'adresse haute A31 à A3
  - Broches de sélection d'octets BE7-BE0

Remarque :Vérification de la parité

Valable sur les bus d'adresse et de donnée. Broche DP7 à DP0

- **Bus de contrôle**
  - Signaux qui déterminent et imposent le cycle du bus du microprocesseur
  - HIT, HITM, HLDA .

2.5.6 Hiérarchie de bus

Problème :

- Nombre d'équipements augmente => performance diminuent
- Le bus est un goulet d'étranglement

Solution :

- Hiérarchie de bus

Architecture traditionnelle

Bus local de la mémoire cache vers le processeur

Contrôleur de cache connecte bus local/bus système

L'extension de bus relié au bus système par une interface d'extension.

Architecture haute performance

Bus haute performance intégré dans le système

2.6 Les jeux d'insructions

Chaque microprocesseur possède un certain nombre limité d'instructions qu'il peut exécuter. Ces instructions s'appellent jeu d'instructions.

2.7 Définition d'un jeu d'instructions

Le jeu d'instructions décrit l'ensemble des opérations élémentaires que le microprocesseur peut exécuter

1. **Les instructions d'affectation :** elle permet de faire le transfert des données entre les registres et la mémoire
2. **Les instructions arithmétiques et logiques**(ET , OU ,ADD,…)
3. **Les instructions de branchement** (conditionnel et inconditionnel)
4. **Les instructions d'entrées sorties.**

Les instructions et leurs opérandes (données) sont stockés dans la mémoire. La taille d'une instruction est égale au nombre de bits nécessaires pour la représenter en mémoire. Elle dépend du type de l'instruction et du type de l'opérande. L'instruction est découpée en deux parties :

- **Code opération :** indique quelle instruction à effectuer. C'est un code de taille N bits
- **Le champ opérande** :Il contient la donnée ou la référence (adresse) à la donnée.

Remarque :

Le champ opérande peut être découpé à sontour en plusieurs champs.

1°/ Machine à 3 adresses

Dans ce type de machine pour chaque instruction il faut préciser:

- L'adresse du premier opérande
- L'adresse du deuxième opérande
- L'adresse l'emplacement du résultat

Code opération	AdresseOpérande 1	AdresseOpérande 2	AdresseRésultat

Exemple l'instruction :ADD A, B, C

Veut dire C←(A+B)

2°/ Machine à 2 adresses

Dans ce type de machine l'adresse l'emplacement du résultat est implicitement l'adresse du deuxième opérande

Code opération	Adresse Opérande 1	Adresse Opérande 2

Exemple l'instruction ADD A,B veut dire : B ← (A+B)

3°/ Machine à 1 adresse

Dans ce type de machine l'emplacement du résultat est dans l'accumulateur (Acc registre de l'UAL)

Code opération	Adresse Opérande 1

Exemple l'instruction ADD A veut dire ; Acc ←(Acc+A)

4°/ Machine à 0 adresses

Dans ce type de machine on additionne les deux opérandes se trouvant au sommet de la pile (SP ; Stack Pointer) et le résultat est empilé.

Code opération	

Remarque :

- Le champ opérande contient la donnée ou l'adresse de la donnée.
- Le mode d'adressage définit la manière dont le processeur va accéder à l'opérande.
- Le code opération de l'instruction comporte un ensemble de bits pour indiquer le mode d'adressage.

2.8 Les modes d'adressage

Les modes d'adressage les plus utilisés sont :Immédiat, Direct et Indirect.

2.8.1 Adressage Immédiat

Ce type d'adressage est noté par # et l'opérande est dans le champ adresse de l'instruction.

Exemple :

ADD	# 25

Dans ce cas, l'opérande =25

L'instruction ADD # 25 veut dire ; Acc (Acc+25)

2.8.2 Adressage Direct

Dans ce type d'adressage,l'adresse de l'opérande est dans le champ adresse de l'instruction.

Exemple :

ADD	25

Dans ce cas, l'adresse de l'opérande =25et l'opérande = [25]

L'instruction ADD 25 veut dire ; Acc←Acc+[25]

2.8.3 Adressage Indirect

Ce type d'adressage est noté par ( )

Exemple :

ADD	(25)

L'adresse de l'opérande = [25]

L'opérande = [[25]]

[[25]] : lire contenu du contenu de 25

L'instruction ADD (25) veut dire ; Acc←Acc + [[25]]

.

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

2.9 Étapes d'exécution d'une instruction

Le traitement d'une instruction est décomposé en trois phases :

- **Phase 1 :** rechercher l'instruction à traiter (PR)
- **Phase 2 :** Exécution de l'instruction (PE)
- **Phase 3 :** préparation de l'instruction suivante (PPIS)

Chaque phase comporte un certain nombre d'opérations élémentaires exécutées dans un ordre bien précis.

Ces opérations sont appelées microcommandes et elles sont générées par le séquenceur.

Remarque :

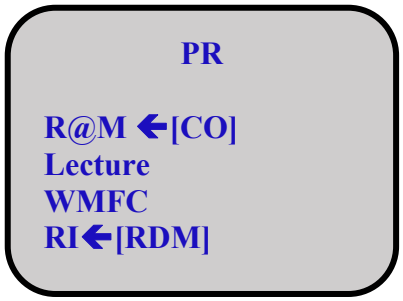
- La phase 1 et 3 ne change pas pour l'ensemble des instructions,
- La phase 2 change selon l'instruction et le mode d' addressage

2.9.1 Exemples d'exécution des instructions

Exemple 1 : Exécuter l'instruction suivante ; ADD #A, # B,C

Phase de Recherche :

1. Mettre le contenu du compteur Ordinal ([CO]) dans le registre d'adresse mémoire (R@M) :
2. Commande de lecture à la mémoire.
3. Attente du signal de contrôle de fonctionnement de la mémoire (WMFC)
4. Transfert du contenu du registre de mot (RDM) dans le registre RI :



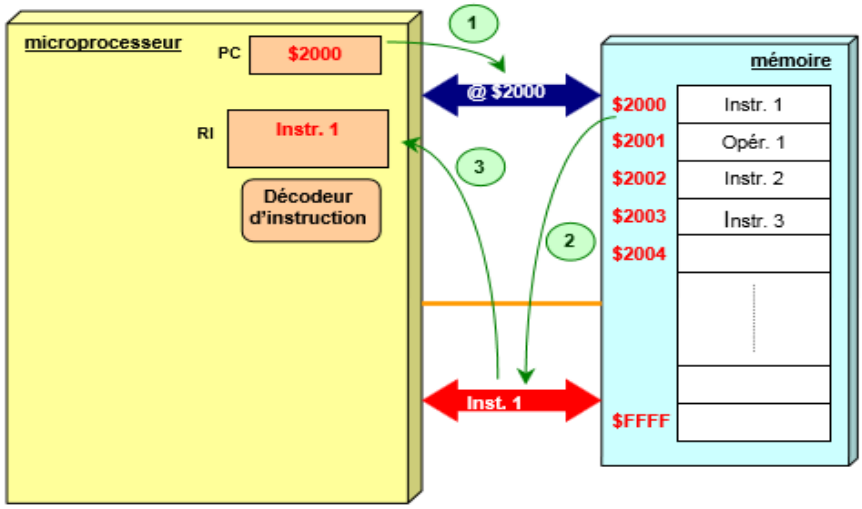


Figure 2.17. Phase Recherche

Phase d'Exécution

1. Analyse et décodage du Code Opération
2. Transfert de l'opérande A dans l'accumulateur 1 de l'UAL
3. Transfert de l'opérande B dans l'accumulateur 2 de l'UAL
4. Commande de l'exécution de l'opération d'addition
5. Transfert du résultat de l'addition dans registre de mot RDM
6. Transfert du contenu du 3ème champ d'adresse de RI (C) dans le registre d'adresse mémoire (R@M).
7. Commande d'écriture
8. Attente du signal de contrôle de fonctionnement de la mémoire WMFC

PE
Analyse du Copération
A cc 1 ← [1er chp d'@de RI]
A cc 2 ← [2ème chp d'@de RI]
Addition
RDM ← [Acc 3]
R@M ← [3ème chp d'@de RI]
Ecriture
WMFC

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

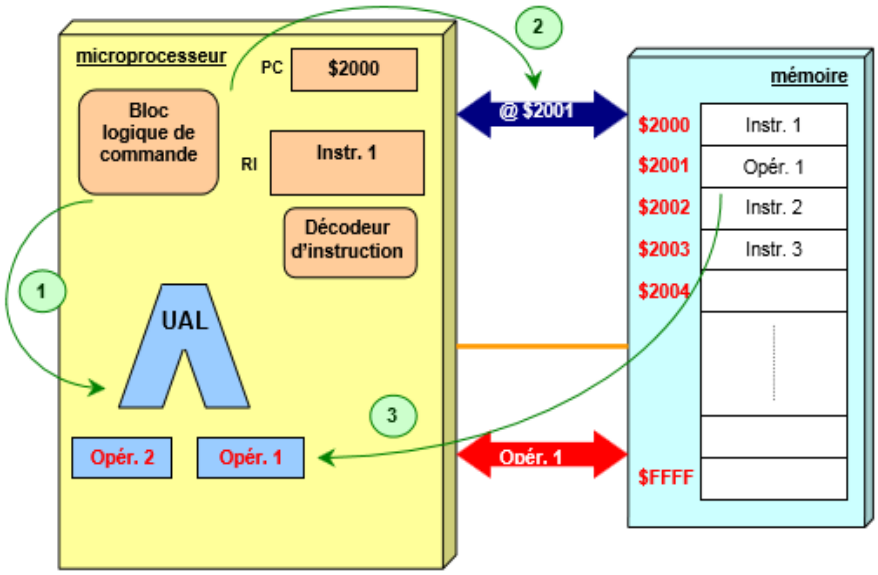
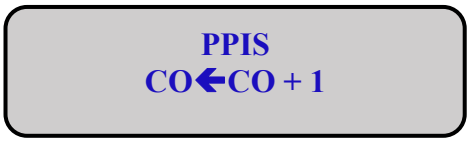


Figure 2.18 : Phase Exécution

Phase de Préparation de à l'Instruction Suivante

Elle consiste à incrémenter le compteur Ordinal

Instruction; ADD #A, # B,C



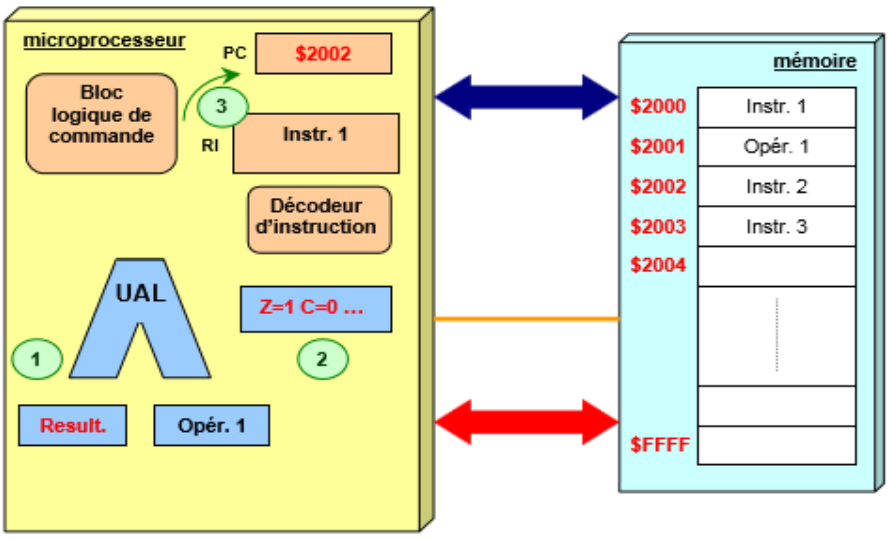


Figure 2.19 : Phase Préparation de l'Instruction Suivante

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

Exemple 2 : Exécuter l'instruction suivante : ADD B

Phase de Recherche :

Mettre le contenu du compteur Ordinal ([CO])dans le registre d'adresse mémoire (R@M) :

Commande de lecture à la mémoire.

Attente du signal de contrôle de fonctionnement de la mémoire

Transfert du contenu du RDM dans le registre RI :

PR
R@M←[CO]
Lecture
WMFC
RI←[RDM]

Phase d'Exécution

1. Analyse et décodage du Code Opération
2. Transfert du contenu du champ d'adresse de RI dans le registre d'adresse mémoire
3. Commande de lecture
4. Attente du signal de contrôle de fonctionnement de la mémoire
5. Transfert du contenu de RDM vers l'UAL
6. Commande de l'exécution de l'opération (addition)

PE
Analyse du Cop
R@M←[chp d'@de RI]
Ecriture
WMFC
Addition

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

Phase de Préparation de à l'Instruction Suivante

Elle consiste à incrémenter le compteur Ordinal

PPIS
CO←CO + 1

Exemple 3 : Exécuter l'instruction suivante : ADD (B)

Phase de Recherche :

1. Mettre le contenu du compteur Ordinal ([CO])dans le registre d'adresse mémoire (R@M) :
2. Commande de lecture à la mémoire.
3. Attente du signal de contrôle de fonctionnement de la mémoire
4. Mettre le contenu du registre mot ([RDM])dans le registre RI

PR
R@M←[CO]
Lecture
WMFC
RI←[RDM]

Phase d'Exécution

1. Analyse et décodage du Code Opération
2. Transfert du contenu du champ d'adresse de RI dans le registre d'adresse mémoire
3. Commande de lecture
4. Attente du signal de contrôle de fonctionnement de la mémoire
5. Transfert du contenu de RDM dans le registre d'adresse mémoire
6. Commande de lecture
7. Attente du signal de contrôle de fonctionnement de la mémoire
8. Transfert du contenu de RDM vers l'UAL
9. Commande de l'exécution de l'opération (addition)

PE
Analyse du Cop
R@M←[chp d'@de RI]
Ecriture
WMFC
R@M←[RDM]
Lecture
WMFC
UAL←[RDM]
Addition

Phase de Préparation de à l'Instruction Suivante

Elle consiste à incrémenter le compteur Ordinal

PPIS
CO←CO + 1

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

2.10 Les entrées/sorties

Entrées/Sorties (E/S, Input/Output ou I/O en anglais) désigne l'ensemble des transferts de données qui permettent au microprocesseur et à la mémoire de communiquer avec le reste du monde :

- **Entrée** : une donnée allant du monde extérieur vers le microprocesseur
- **Sortie** : une donnée allant du microprocesseur vers le monde extérieur.
- Les Entrées/Sorties proviennent des périphériques.
- Les périphériques interagissent avec l'unité centrale et la mémoire.
- Certains périphériques sont branchés à l'intérieur de l'ordinateur (disques durs, carte réseau,…) alors que d'autres sont branchés sur des interfaces externes de l'ordinateur (clavier, écrans, souris, etc…)
- Les périphériques ont des modes de fonctionnement variés et ils ont souvent leurs propres formats de données.
- Les périphériques ne vont pas à la même vitesse que le CPU. Beaucoup sont très lents par rapport au CPU alors que certains sont plus rapides…
- Pour chaque périphérique, il existe une unité spéciale appelée module d'E/S (I/O module) qui sert d'interface entre le périphérique et le microprocesseur
- Ces modules sont habituellement appelés: contrôleurs.

Par exemple :

Le **module d'E/S** servant d'interface entre le microprocesseur et un disque dur sera appelé :**Contrôleur de Disque**

2.10.1 Fonctions principales des modules d'E/S

- Lire ou écrire des données du périphérique.
- Lire ou écrire des données du Microprocesseur/Mémoire.
- Certains modules d'E/S doivent générer des interruptions ou accéder directement à la mémoire.
- Contrôler le périphérique et lui faire exécuter des séquences de tâches.
- Mettre certaines données du périphérique ou du microprocesseur en mémoire tampon afin d'ajuster les vitesses de communication.
- Tester le périphérique de détection d'erreur.

2.10.2 Types de périphériques

Nom	Entrée ou Sortie	Vitesse des données
Clavier	Entrée	Très faible
Souris	Entrée	Faible
Ecran tactile	Entrée	Faible à Moyenne
Scanner	Entrée	Moyenne
Voix	Entrée	Faible à Moyenne
USB	Entrée - Sortie	Faible à très élevé
Son	Entrée - Sortie	Moyenne
Imprimante	Sortie	Faible à Moyenne
Flash disque	Stockage	Moyenne
Carte graphique	Sortie	Elevé

Tableau2. 8 : Exemple de périphériques

2.10.3 Adressage des composants d'E/S

Comme plusieurs composants sont connectés à un ordinateur, alors il faut pouvoir sélectionner un composant par une opération d'E/S

Dans ce cas on accède à un composant d'E/S comme on accède à une case mémoire.

L'UC sélectionne un composant en plaçant son adresse sur le bus d'adresse.

Seul le composant qui reconnait son code, répond à la commande de l'UC placée sur les lignes de contrôle.

Ce type d'adressage est appelé E/S à adresse mémoire(Memory Mapped I/O)

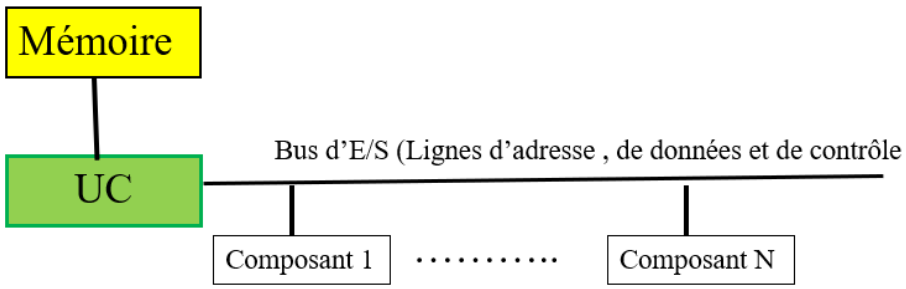


Figure 2.20 : Adressage des composants d'E/S

2.10.4 Classification des E/S

Il existe trois techniques principales pour communiquer à partir du CPU/Mémoire vers un périphérique à travers un module de I/O:

- E/S programmées
- E/S avec interruptions
- Le Direct Memory Access (DMA)

❖ **E/S programmées :**

Consistent à concevoir un programme, exécuté par le microprocesseur, qui communiquera avec le module d'E/S afin d'obtenir des données du périphérique.Le microprocesseur. peut tester, lire, écrire ou contrôler le périphérique à travers le module d'E/S.

Remarque: Pour accéder à un périphérique, le microprocesseur exécute le programme suivant: Pour accéder à un périphérique, le microprocesseur exécute le programme suivant:

1. Vérifier si le périphérique est prêt
2. Envoyer une requête au périphérique
3. Attendre que la requête soit finie en interrogeant les registres de statut du périphérique (polling)
4. Lorsque la requête est finie, lire le mot disponible dans le module d'E/S
5. Recommencer les opérations de 2 à 5 tant que le programme n'est pas terminé…

❖ **E/S avec interruptions**

La méthode d'E/S avec interruption consiste à accéder au module d'E/S lors d'interruptions provenant de ce dernier afin de signaler un évènement particulier.Une façon d'éliminer les délais d'attente du microprocesseur est d'utiliser des interruptions

Remarque: Le Traitement d'une interruption suit les étapes suivantes :

1. Terminer l'instruction en cours
2. Déterminer s'il faut traiter l'interruption.
3. Sauvegarder le contexte
4. Déterminer l'adresse de la routine de traitement de l'interruption
5. Exécuter cette routine
6. Restaurer le contexte
7. Reprendre là ou le processeur était rendu

❖ **Le Direct Memory Access DMA**

Le Direct Memory Access (DMA) est un transfert de données direct entre un périphérique et la mémoire ou vice versa, effectué sans intervention du microprocesseur.

Le contrôleur de DMA est un circuit intégré qui gère le transfert par DMA.

Le transfert par DMA se fait par bloc de données .

Remarque: Le microprocesseur configure le transfert de DMA en exécutant les instructions suivantes.

Le microprocesseur configure le transfert de DMA en exécutant les instructions.

Quigénéralement indiquent, le périphérique visé

La plage de mémoire visée (c.a.d l'adresse de départ et le nombrede données à transférer)
La vitesse de transfert, la taille des données transférées et le modede transfert (unique ou par bloc)

Une fois la configuration effectuée, le transfert par DMA débutelorsque le microprocesseur exécute une instruction initiant le transfert

Pendant l'initialisation du transfert par DMA, le contrôleur deDMA négocie l'accès au bus de données avec le microprocesseur.

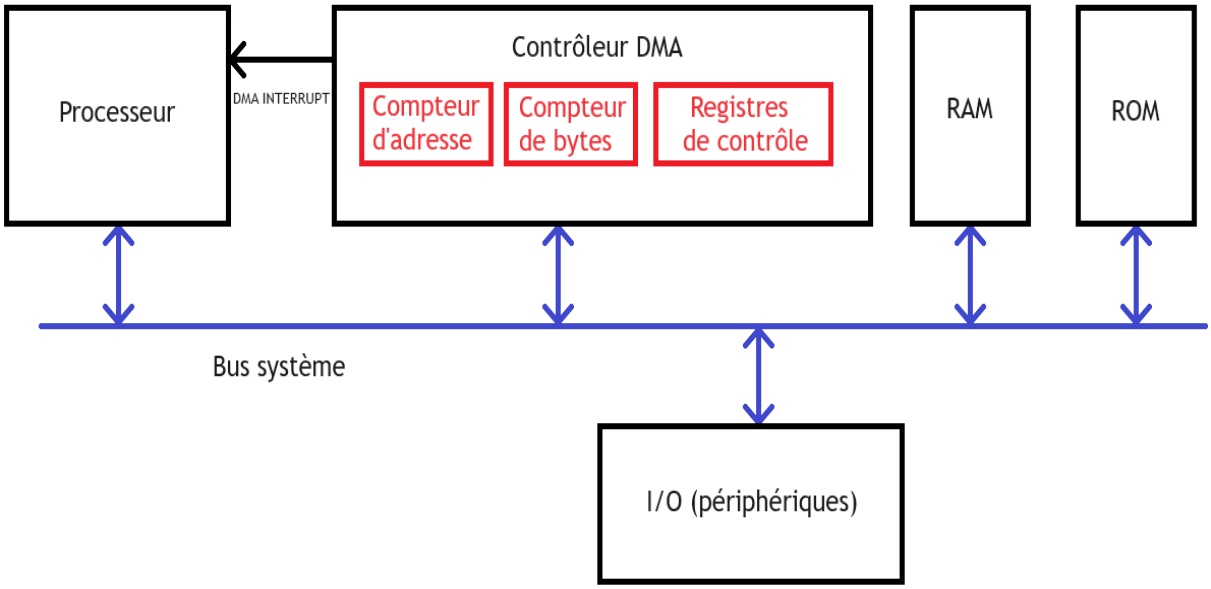


Figure2.21: Le Direct Memory Access DMA

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

CHAPITRE-3-

Notions sur les Instructions d'un Ordinateur

Les **instructions** d'un ordinateur sont les opérations de base que le processeur exécute pour faire fonctionner un programme. Chaque instruction indique au processeur une action précise, comme effectuer un calcul, déplacer des données ou réaliser une opération logique. Ces instructions sont codées en **langage machine** (binaire), mais elles peuvent aussi être représentées en **langage assembleur** pour être plus faciles à comprendre par les programmeurs. L'ensemble des instructions qu'un processeur peut exécuter est appelé **jeu d'instructions**.

3.1 Introduction sur les instructions d'un ordinateur

Les processeurs doivent reconnaître des instructions codifiées sous la forme de groupes de bits. L'ensemble des instructions reconnues par un processeur et son système de codage forment ce qu'on appelle le langage machine du processeur.

3.2 Langage machine

Le langage machine désigne un jeu d'instruction et des règles syntaxique. Le langage machine est le langage compris par le microprocesseur.

Remarque :

- Afin de faciliter la tâche du programmeur, on a créé différents langages évolués.
- A l'aide d'un langage évolué, le programmeur écrit le code source. Ce dernier n'est pas directement exécutable, il faut le traduire en code objet ou code machine.
- Cette traduction se fait à l'aide d'un traducteur tel que:Assembleur, Compilateur,
- Le langage assembleur est le langage le plus proche du langage machine.
- Il est composé par des instructions que l'on appelle : Symboles mnémoniques. Chaque instruction représente un code machine différent.
- Chaque microprocesseur possède son propre langage assembleur.
- Chaque instruction en langage évolué correspond à une succession d'instructions en langage assembleur.
- Le programme en langage évolué n'est pas compréhensible par le microprocesseur.
- Il faut le compiler pour le traduire en assembleur puis l'assembler pour le convertir en code machine compréhensible par le microprocesseur.

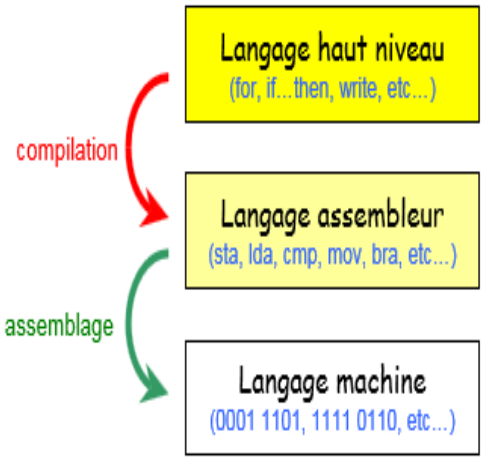


Figure 3.1:Langage machine

- On peut caractériser la puissance d'un microprocesseur par le nombre d'instructions qu'il est capable de traiter par seconde.

Pour cela, on définit :

- Le **CPI** (Cycle Par Instruction) c'est le nombre moyen de cycles d'horloge nécessaire pour l'exécution d'une instruction pour un microprocesseur donné. Ƒ
- Le **MIPS** (Millions d'Instructions Par Seconde) : représente la puissance de traitement du microprocesseur.

- La moyenne des cycles par instruction dans un processus donné est définie comme suit :

Le Cycle Par Instruction

$$CPI = \frac{\sum i(ICi)(Cci)}{IC}$$

Où
ICi est le nombre d'instructions pour un type d'instructiondonné i,
Cci est les cycles d'horloge pour ce type d'instruction et
IC est le nombre total d'instructions.

La sommation résume tous les types d'instructions pour un processus d'analyse comparative donné

- La puissance du processeur peut ainsi être caractérisée par le nombre d'instructions qu'il est capable de traiter par seconde.
- L'unité utilisée est le MIPS (Millions d'Instructions Par Seconde) correspondant à la fréquence (en MHz) du processeur que divise le CPI et il est défini comme suit :

$$MIPS = \frac{Fréquence}{CPI}$$

Conclusion

Pour augmenter les performances d'un microprocesseur, on peut donc :

- Soit augmenter la fréquence d'horloge(limitation matérielle),
- Soit diminuer le CPI(choix d'un jeu d'instruction adapté)

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

CHAPITRE 4

Étude de cas : processeur Intel 8086

Ce chapitre est une introduction au microprocesseur Intel 8086.

Parmi les microprocesseurs qui ont marqué l'histoire de l'informatique, le Intel 8086, développé par la société Intel et introduit en 1978

Le Intel 8086 est un microprocesseur 16 bits qui a donné naissance à la célèbre famille d'architectures x86, largement utilisée dans les ordinateurs personnels.

Il se distingue par son organisation interne basée sur deux unités principales :

1°/l'unité d'exécution (EU) : décode et exécute les instructions en utilisant l'UAL et les registres internes

2°/ l'unité d'interface de bus (BIU) : génère la communication avec la mémoire et les entrées / sortieset elle précharge les instructions dans une file d'attente.

Cette séparation permet un début de parallélisme ; pendant que l'EU exécute une instruction, la BIU peut charger la suivante.

L'étude de ce processeur permet de comprendre les principes fondamentaux de l'architecture des microprocesseurs, notamment l'organisation des registres, les modes d'adressage et le fonctionnement du cycle d'instruction.

4.1 Présentation de la famille 80x86

4.1.1 Historique & caractéristiques générales

Le microprocesseur Intel 8086 est apparu en 1978. C'est le premier microprocesseur de la famille Intel 80x86 (8086, 80186, 80286, 80386, 80486, Pentium, ...). Il se présente sous la forme d'un boitier DIP (Dual In-line Package) à 40 broches :

Figure 4.1 : Le microprocesseur Intel 8086

- C'est un microprocesseur ayant un bus de données 16 bits
- Ce processeur de 16 bits pouvait gérer 1 Mo de mémoire
- Bus d'adresse de 20 bits
- La fréquence choisie par IBM (4,77 MHz) était assez faible.
- Le processeur a atteint 10 MHz en fin de carrière

Le microprocesseur 8086 d'Intel se caractérise par :

- Structure 16 bits.
- Capacité d'adressage de 1 Mo.
- 14 registres internes de 16 bits.
- 7 modes d'adressage.
- Opérations sur des bits, des octets, des mots et des chaînes de caractères.
- Arithmétique signée ou non signée.
- Arithmétique binaire ou BCD, sur 8bits ou 16 bits.

4.2 Organisation interne du processeur Intel 8086

Le processeur 8086 dispose de quatorze registresgénéraux de 16 bits chacun. Ces registres sont AX, BX, CX et DX et ils sont composés de deux registres de 8 bits, par exemple le registre AX= AH+AL avec AH et AL des registres de 8 bits.

Le 8086 dispose de 3 jeux de registres spécifiques de 16 bits :

- Les registres généraux,
- Les pointeurs et index,
- Les registres de segments,

Les Registres Généraux	Accumulateur (AX, AH, AL)
	Base (BX, BH, BL)
	Compteur (CX, AH, CL)
	Données (DX, DH, DL)
Les pointeurs et index	pointeur de pile (SP)
	Pointeur de base (BP)
	Index de source (SI)
	Index de destination (DI)
Les registres de segments	segment de code (CS)
	Segment de données (DS)
	Segment de pile (SS)
	Segment supplémentaire(ES)
IP (*Compteur ordinal)*	Pointeur d'instruction

Tableau4.1 :Registres spécifiques au 8086

Remarque :
Les registres AX, BX, CX et DX sont des registres généraux de 16 bits chacun.Ces registres peuvent être considérés comme formés de 2 registres de 8 bits.

Exemple :

AX peut être décomposé en AH et AL,

AH (registre de 8 bits) est formé de la partie haute (high) de AX

AL (registre de 8 bits) de la partie basse (low) de AX

a) Registres spécifiques au 8086 (Pointeurs et Index)

- **SI** : Source Index ou index d'origine
- **DI :** Destination Index ou index de destination

  Ils sont associés à des opérations de traitement de chaînes mais peuvent également être considérés comme registres à usage général.
- **SP** (Stack Pointer ou indicateur de sommet de pile) ➔ne peut être utilisé comme registre à usage général.
- **BP** (Base Pointer ou registre de base)➔peut être comme registre à usage général

b) Registres spécifiques au 8086 (Registres de segments)

Quatre registres jouent un rôle fondamental lors des calculs d'adresse

- **CS :** Code Segment
- **DS :** Data Segment (ou Segment de Données)
- **ES :** Extra Segment
- **SS :** Stack Segment (ou segment de pile)

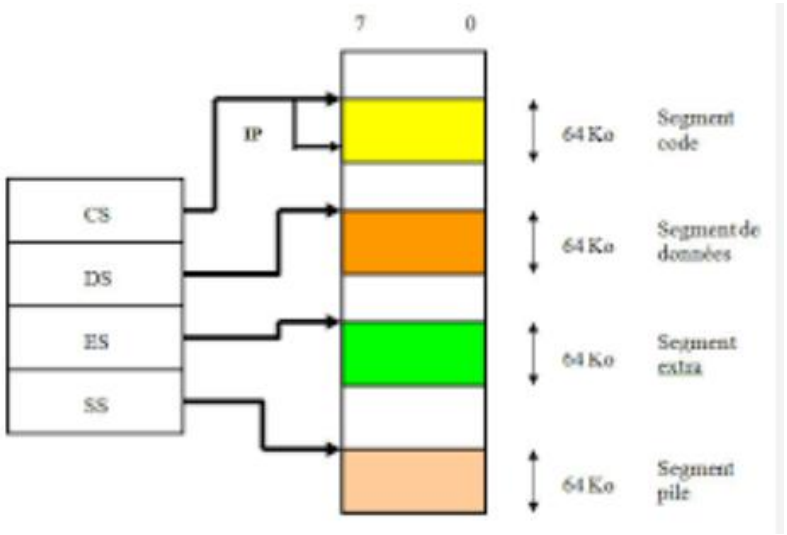


Figure 4.2 : Registres spécifiques au microprocesseur Intel 8086

Remarque :

Si le registre segment n'est pas spécifié alors le processeur l'ajoute par défaut selon l'offset choisit :

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

Offset utilisé	Valeur	DI	SI	BX	BP
Registre Segment par défaut qui sera utilisé par le CPU	DS	SS			

Tableau 4.2: segment par défaut

c) **Le registre IP (Instruction Pointer** « ou compteur ordinal) contient l'adresse de la prochaine instruction à effectuer.

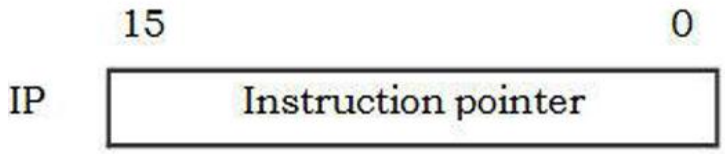


Figure 4.3 : Registre Instruction Pointer

d) **Le registre STATUS** ou registre d'ETAT FLAG contient les indicateurs d'état du microprocesseur. Il a le format suivant :

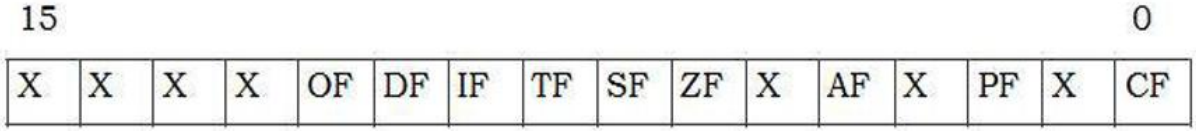


Figure 4.4 : Registre d'état Flag

- **O :** débordement (overflow)
- **D :** direction
- **I** : interruption
- **T** : fonctionnement pas à pas
- **S** : signe
- **Z** : zéro
- **A** :auxiliary carry
- **P** : parité
- **C** : retenue (carry)

4.3 Architecture interne du microprocesseur 8086

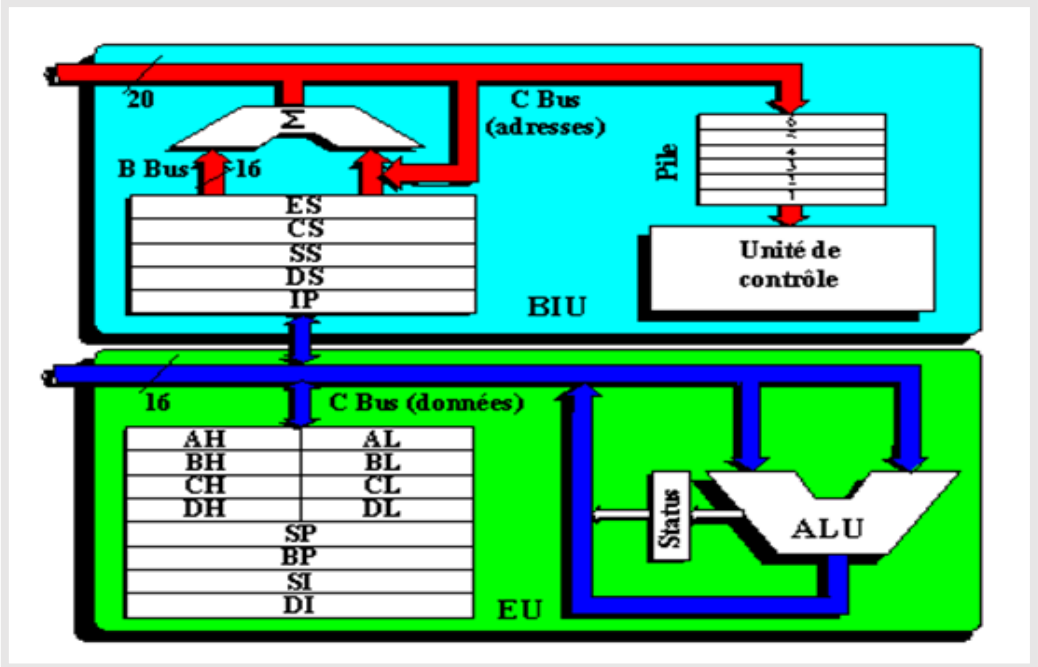


Figure 4.5 : Architecture du processeur 8086

4.3.1 Brochage du microprocesseur 8086

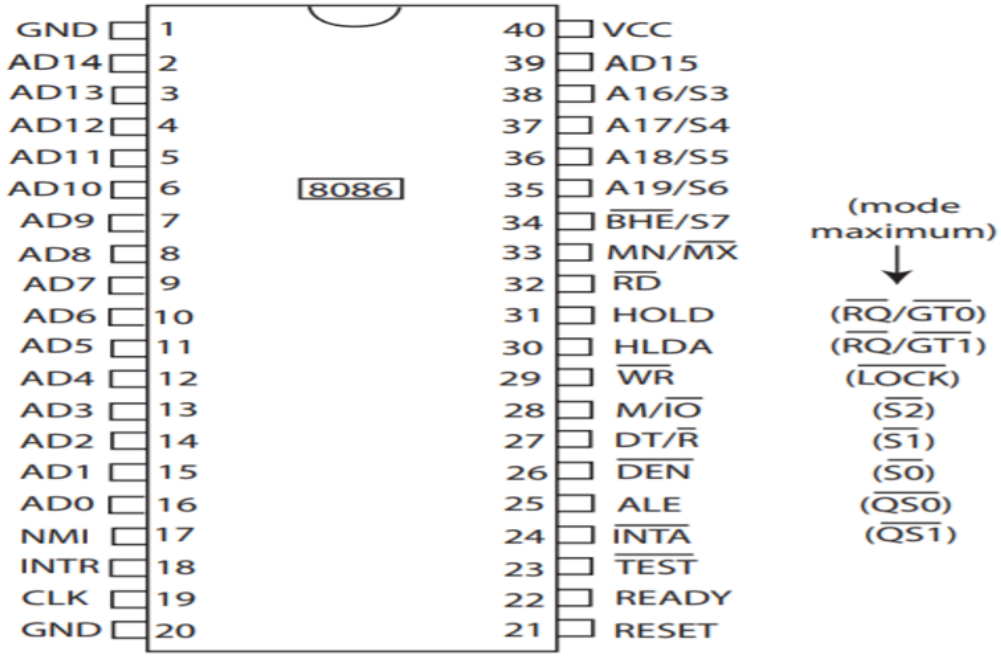


Figure 4.6 : Brochage du microprocesseur Intel 8086

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

4.4 Types d'instruction du 8086

- Chaque microprocesseur possède un certain nombre limité d'instruction appelé jeu d'instructions.
- Jeu d'instructions décrit l'ensemble des opérations élémentaires que le processeur peut exécuter.
- Les instructions peuvent avoir 0, 1, ou 2 opérandes.

Exemples :

- Instructions sans opérande : NOP, STI, CLI, PUSHF, CBW...;
- Instructions avec un seulopérande : INC, DEC, NEG, NOT ....;
- Instructions avec deux opérandes : CMP, ADD, MOV, LEA, XCHG, AND .....

Un opérande peut se trouver dans un registre du 8086 :Dans un octet, dans un mot 16 bits ou dans un double mot 32 bits.

Les opérations peuvent avoir lieu :

- Entre un registre et un autre registre,
- Entre un registre et un octet ou un mot en mémoire
- Mais pas entre 2 octets ou 2 mots en mémoire

Remarque : il faut passer dans ce cas par un registre.

- Les instructions peuvent être classées en 3 catégories:
  - Instructions de Transfert
  - Instructions d'Opération Arithmétique et Logique
  - Instructions de Contrôle de flux

4.4.1 Description des Instructions de Transfert

Les instructions de transfert permettent de déplacer des données d'une source vers une destination sans les modifier.

- Registre vers mémoire ;
- Registre vers registre ;
- Mémoire vers registre.

Instructions	Descriptions
MOV dest, src	Transfert entre registres ou mémoire
PUSH reg	Empile le registre sur la pile
POP reg	Dépile dans le registre
XCHG A, B	Échange les contenus de A et B
IN AL, port	Lit un octet depuis un port
OUT port, AL	Écrit un octet vers un port

Tableau4.3 : Instructions de transfert

Remarque : le microprocesseur 8086 n'autorise pas les transferts de mémoire vers mémoire donc, il faut passer par un registre intermédiaire.

Syntaxe :

MOV destination,source (MOV c'est to move = déplacer)

Destination désigne un registre ou bien une case mémoire même chose pour source.

Exemple

MOV AX, BX : copie le contenu du registre BX dans le registre AX

MOV [200], AX : stocke le contenu de registreAX dans l'adresse mémoire 200_{hexa}

4.4.2 Description des Instructions Arithmétiques

Instructions	Descriptions
ADD A, B	A ← A + B
SUB A, B	A ← A – B
MUL reg	Multiplication non signee
IMUL reg	Multiplication signee
DIV reg	Division non signee
INC reg	Incrémentation (+1)
DEC reg	Décrémentation (-1)
NEG reg	Negation (valeur opposée)
ADC / SBB	Addition / Soustraction avec retenue

Tableau 4.4 : Instructions Arithmétiques

Exemple d'Instructions Arithmétiques

a) Les instructions d'addition : ADD

```
ADD AX,BX   ; AX reçoit AX + BX
```

b) Les instructions de soustraction : SUB

```
SUB AX,DX  ; AX reçoit AX-DX
```

c) Les instructions de multiplication : MUL

MUL CL ; AX =AL * CL résultat sur 16 bits

d) Les instructions de division : DIV

DIV BL ; AX est divisé par BL, le reste est dans AH, le quotient dans AL.

4.4.3 Description des Instructions Logiques

Instructions	Descriptions
AND A, B	ET logique
OR A, B	OU logique
XOR A, B	OU exclusif
NOT A	Inversion de tous les bits
TEST A, B	Test logique sans modification
SHL / SAL	Décalage à gauche
SHR / SAR	Décalage à droite

Tableau4.5: Instructions Logiques

Exemple d'Instructions Logiques

Syntaxe :
AND Destination, S ource
OR Destination, S ource
XOR Destination, Source

Avec : Destination = registre/case mémoire
Source = registre/valeur immédiate/case mémoire
à condition que Destination ne soit pas une case mémoire.

Exemple :
AND AX, Masque ; l'instruction fait un ET logique entre le [AX]et le [Masque], Le résultat est dans AX

OR opérande1, opérande2L'opération effectuée est un OU logique :
opérande1 ← opérande 1 OU opérande 2.

4.4.4 Description des Instructions de contrôle de flux

Instructions	Descriptions
JMP addr	Saut inconditionnel
JE / JZ	Saut si égal / zéro
JNE / JNZ	Saut si non égal
JG / JL	Sauts conditionnels signés
LOOP, CALL, RET	Boucles, appel fonction, retour fonction

Tableau4.6 : Instructions de contrôle de flux

Résumé visuel :

Catégorie	Instructions clés
Transfert	MOV, XCHG, PUSH, POP, IN, OUT
Arithmétique	ADD, SUB, INC, DEC, MUL, DIV, NEG
Logique	AND, OR, XOR, NOT, TEST, SHL, SHR
Contrôle de flux	JMP, JE/JZ, LOOP, CALL, RET

Tableau 4.7 : résumé visuel des instructions

4.5 Mode d'adressage du 8086

Le mode d'adressage définie la manière dont le microprocesseur va accéder à l'opérande.Les champs d'adresse de l'instruction contiennent la donnée ou un certain nombre de bitsindiquant le mode d'adressage.

La structure la plus générale d'une instruction est la suivante :

```
Instruction   Opérande1,  Opérande 2
```

L'opération est réalisée entre les deux opérandes et le résultat est le résultat est toujours récupéré dans l'opérande1. Il y a des instructions qui agissent sur un seul opérande.

Les opérandes peuvent être des registres, des constantes, ou le contenu de cases mémoire. Ce principe est appelé mode d'adressage.

Les modes d'adressage les plus utilisés sont :

- Registre à Registre
- Immédiat (l'opérande existe dans le champ adresse de l'instruction)
- Direct (l'adresse de l'opérande existe dans le champ de l'instruction)

- Indirect ou Basé (l'adresse de l'adresse de l'opérande existe dans le champ de l'instruction)
- Indexé (l'adresse de l'opérande est dans une zone mémoire et l'adresse de cette zone est dans un registre.)

4.5.1 Adressage registre à registre.

Dans ce mode, la taille des deux registres doit être la même.

Exemples : MOV AX, BX ; opérandes 16 bits

L'instruction charge le contenu du registre BX dans le registre AX.

4.5.2 Adressage immédiat

Dans ce mode d'adressage, l'opérande est directement spécifié dans l'instruction. L'opérande est une valeur constante.

Syntaxe :MOV Registre, donnée immédiate

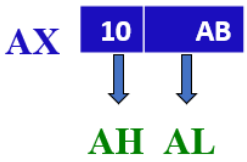
Exemples :

ADD AX, 25 Charger le registre AX par la valeur : [AX] + 25

MOV AL, $12_{(Hexa)}$: L'instruction charge le registre AL avec la valeur $12_{(Hexa)}$.

MOV [1100H], $40_{(Hexa)}$ L'instruction charge l'emplacement mémoire d'adresse 1100 avec la valeur 40.

MOV AX, $10AB_{(Hexa)}$Après exécution de cette instruction on charge l'opérande $10AB_{(Hexa)}$dans le registre AX comme suite :



Remarque : Le registre AX est formé de AH poids fort (Hight) , AL poids faible (low)

4.5.3 Mode d'adressage Registre : (registeraddressing mode)

L'opérande est dans un registre

Syntaxe : MOV Registre1, Registre 2

Exemple: MOV AX, BX *si on a*

AX	*FF*	*34*	**et**	**BX**	*10*	*AB*

Après exécution de cette instruction on charge le contenu de BX dans AX donc on aura le registre AX comme suite :

AX	*10*	*AB*

4.5.4 Adressage direct. (*Direct addressing mode)*

Dans ce type d'adressage un des deux opérandes se trouve en mémoire. L'adresse de la case mémoire ou plus précisément son offset est précisé directement dans l'instruction. Il doit être placé dans [ ]. Si le segment n'est pas précisé le segment DS est pris par défaut.

Le transfert se fait de registre à un emplacement mémoire, ou un emplacement mémoire vers un registre.

Syntaxe : MOVE Registre, [SI + offset]

Exemple :

MOV [452], AX Charger le contenu de AX dans la mémoire d'adresse DS:452

MOV AL, $[2400]_{Hexa}$L'instruction charge la donnée situé dans la case mémoire d'adresse 2400 dans le registre AL

Remarque : La valeur 2400 $_{(Hexa)}$représente l'offset de la case mémoire dans le segment de données (DS).(*Copier le contenu de la mémoire d'adresse DS:2400dans AL*).

Exemple 1 :

MOV AL, $[5000_H]$Transférer le contenu del'adresse **5000** dans le registre **AL**

Si on a par exemple :

5000_H	22
5001	34
5002	56
5003	2A

On a $[5000_H] = 22_H$

alors AL | | 22 |

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

Exemple 2 :
MOV AX, [5000$_H$] Ici on a le registre AX donc Il faut transférer les 2 octets à partir de l'adresse **5000$_H$** car AX est un registre de 16bits
d'où :

Exemple 3 :
MOV [5000$_H$], AL Transférer **[AL]** dans la case mémoire d'adresse **5000$_H$**
Donc Il faut transférer l'octet à partir de l'adresse 5000$_H$ d'où

Si le contenu deregistre **AL** est :

*A*lors le contenu de la casemémoire **5000 $_H$**est **4F$_H$**

4.5.5 Adressage indirect. (*Indirect addressing mode)*

Dans ce mode d'adressage le contenu du registre DS est combiné avec l'adresse effective pour obtenir l'adresse physique.L'adresse de l'opérande est stockée dans un registre qu'il faut bien le charger au préalable par la bonne adresse.

Un des deux opérandes se trouve en **mémoire**. L'offset de l'adresse n'est pas précisé directement dans l'instruction, il se trouve dans l'un des 4 registres d'offset BX, BP, SI ou DI et c'est le registre qui sera précisé dans l'instruction :

L'adressage indirect est divisé en 3 catégories selon le registre d'offset utilisé. On distingue ainsi, l'adressage indexé, l'adressage Basé et l'adressage basé indexé.

Syntaxe : MOV Registre1, [Adresse]

Exemples :

MOV AX, [BX]Charger AX par le contenu de la mémoire d'adresse DS:BX

MOV AX, [SI]Charger AX par le contenu de la mémoire d'adresse DS:SI

- **Adressageindexé. (Indexed addressing mode)**

L'offset se trouve dans l'un des deux registres d'index SI ou DI. On peut préciser un déplacement qui sera ajouté au contenu de Ri pour déterminer l'offset,

Syntaxe : MOVE Registre, [SI + offset]

Exemple :
MOV AX, [DI-100] ; Charger AX par la mémoire d'adresse **DS**:DI-100
MOV AX, [SI] ; Charger AX par le contenu de la mémoire d'adresse **DS**:SI

- **Adressagebasé : (Based addressing mode)**

L'offset se trouve dans l'un des deux registres de base BX ou BP. Donc l'adresse est formée d'un registre de base BX ou BP et d'un déplacement.

Syntaxe : MOVE AX, [BX+ déplacement]

Exemple :
MOV AX, [BX] Charger AX par le contenu de la mémoire d'adresse **DS**:BX
MOV AX, [BX+2]Charger AX par le contenu de la mémoire d'adresse **DS**:BX+2
MOV AX, [BP-26] Charger AX par le contenu de la mémoire d'adresse **SS**:BX-2

- **Adressagebaséindexé. (Based Indexed addressing mode)**

L'offset de l'adresse de l'opérande est la somme d'un registre de base (BX ou BP), d'un registre d'index (SI ouDI) et d'un déplacement optionnel.

Syntaxe : MOVE AX, [BX+ SI]

Exemples :
MOV AX,[BX+SI] AX est chargé par la mémoire d'adresse **DS**:BX+SI
MOV AX,[BX+DI+4] AX est chargé par la mémoire d'adresse **DS**:BX+DI+5

4.5.6 Résumé des modes d'adressage

Mode d'adressage	Exemple d'instruction
Immédiat	Move AX, 6
Registre	Move AX, BX
Direct	Move AX, [1234_h]
Indirect	Move AX, [BX]
Indexé	Move AX, [SI+5]
Basé	Move AX, [BX+10_h]
Basé Indexé	Move AX, [SI+SI]

Tableau4.7 : Les modes d'adressage du 8086

4.6 Gestion de la pile (Stack Management)

- **La** pile est une zone de la mémoire utilisée pour stocker temporairement des données (adresses de retour, variables locales, etc.).
- **Elle** fonctionne selon le principe LIFO (Last In, First Out).
- **Les**registresassociéssont :

- SP (Stack Pointer) : pointe vers le sommet de la pile
- **SS** (Stack Segment) : définit le segment de pile

➢ **Les**principales instructions

Instruction	Fonction
PUSH reg/mem	Empile une valeur dans la pile
POP reg/mem	Dépile une valeur depuis la pile
PUSHF	Empile le registre FLAGS
POPF	Dépile le registre FLAGS

Tableau4.8 : Instructions pour la gestion de la pile

4.7 Fonctions et sous - programmes

➢ **Les** sous-programmes sont des blocs de code que l'on peut appeler à plusieurs endroits dans un programme.
➢ **Leur** gestion repose sur l'utilisation de la pile pour stocker l'adresse de retour.
➢ Instructions utilisées :

Instruction	Description
CALL adresse	Appelle une procédure ; empile l'adresse de retour
RET	Retour de procédure ; dépile l'adresse de retour

Tableau 4.9: Instructions d'appel de sous-programme

4.8 Notions sur les interruptions

➢ **Une interruption** est un mécanisme qui permet d'interrompre l'exécution d'un processus suite à un événement extérieur ou intérieur et de passer le contrôle à une routine dite routine d'interruption ou traitement d'interruption.
➢ **Le système d'interruption** est un dispositif, incorporé au niveau du séquenceur, qui enregistre et traite les signaux d'interruption envoyés au microprocesseur.

- **Arrêter** le processus en cours
- **Sauvegarder** le contexte du processus interrompu
- **Exécuter** le programme de routine d'interruption
- **Restaurer** le contexte du processus interrompu
- **Reprendre** l'exécution du programme interrompu

4.8.1 Types interruptions:

- Interruptions matérielles (ex : clavier, timer)
- Interruptions logicielles (ex : INT n)

Instructions	Fonctions
INT n	Déclenche une interruption logicielle de numéro n
IRET	Retour d'interruption : restaure IP, CS et FLAGS depuis la pile

Tableau4.10 : Instructions matérielles et logicielles

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

CHAPITRE 5

Étude de cas : processeur de 32 bits

Un microprocesseur 32 bits est un processeur capable de traiter des données par blocs de 32 bits à la fois, limitant sa capacité d'adressage mémoire à environ 4 Go (Gigaoctets) de RAM, et il est aujourd'hui largement remplacé par les architectures 64 bits pour les tâches modernes.

Ces processeurs gèrent des adresses mémoire de 32 bits, ce qui est idéal pour les anciens systèmes et applications, mais inefficace pour les logiciels gourmands en mémoire actuelle.

5.1 Introduction générale aux architectures 32 bits et 64 bits

Les processeurs se ressemblent tous du point de vue de l'utilisateur. Ils comportent principalement une mémoire, un ensemble de registres et un jeu d'instructions

Les différences proviennent surtout du jeu d'instruction.

5.2 Capacité mémoire et architecture

- **Taille d'adresse mémoire 32 bits :**

Peut adresser au maximum 2^{32} octets soit un espace théorique de 4 Go de RAM.
En pratique, l'OS réserve une partie pour le noyau En pratique, les systèmes en utilisent souvent 3 à 3,5 Go.

- **Taille d'adresse mémoire 64 bits :**

Peut théoriquement adresser 2^{64} octets soit un espace théorique de 16 exa octets, mais limité par : l'architecture (x86-64 limite souvent à 48 bits d'adressage physique), et la carte mère et l'OS.
Systèmes actuels : support courant de 128 Go à plusieurs To.

Remarque :

- Les systèmes actuels en exploitent rarement plus de 128 Go, 256 Go ou plusieurs téraoctets, selon l'architecture
- Plus d'adresses disponibles → possibilité d'utiliser plus de RAM, d'allouer de grands buffers, d'exécuter de nombreuses applications lourdes (IA, bases de données, machines virtuelles).

5.3 Performance et micro-architecture

➢ **Taille des registres 32 bits :**

Registres généraux de 32 bits donc capacité de calcul limitée à cette taille. Ce qui entraine des limites sur les calculs entiers et les opérations nécessitant des valeurs > 2^{32}.

➢ **Taille des registres 64 bits :**

Les registres généraux de 64 bits ce qui facilite des calculs plus rapides pour les entiers et traitement plus efficace des données volumineuses.

➢ **Evolution des architectures 64 bits (AMD64 / Intel 64)**

Cette architecture ajoute 8 registres généraux supplémentaires, ce qui réduit l'accès à la mémoire et augmente la vitesse donc un gain de performance

➢ **Bus de données et pipeline**

Les CPU 64 bits modernes profitent de :

- busmémoire plus larges,
- caches plus grands,
- exécution superscalaire améliorée.

***Re*marque :**
Les systèmes 64 bits permettent :

- Une gestion plus rapide des applications multitâches,
- Des instructions plus puissantes pour cryptographie, compression, multimédia, un meilleur parallélisme.

5.4 Compatibilité système

➢ **Systèmes d'exploitation**

Un processeur 32 bits ne peut exécuter que un OS 32 bits.

Un processeur 64 bits peut exécuter un OS 64 bits, et parfois un OS 32 bits (selon les fabricants).

➢ **Pilotes**

Les pilotes 32 bits ne fonctionnent généralement pas sur un OS 64 bits. Ils nécessitent des pilotes spécifiques (drivers 64 bits.)

Remarque :

Les systèmes 64 bits sont aujourd'hui les standards, mais certains logiciels très anciens peuvent poser problème.

5.5 Cas Particuliers Le MIPS R3000

Le MIPS R3000 de l'anglais Microprocessor without Interlockeed Pipeline Stage est conçu en 1985 par la société Computer System.

Ce type de processeur est utilisé surtout dans les systèmes embarqués comme les ordinateurs de poche, les consoles de jeux, les routeurs Cisco etc….

5.5.1 Description du MIPS R3000

- L'architecture **MIPS** de l'anglais **M**icroprocessorwithout**I**nterlockeed**P**ipeline **S**tage est une architecture de processeur du type ;**RISC** :**R**educed**I**nstruction **S**et **C**omputer développée par la société MIPS Technologiesnommée MIPS Computer Systems basée en Californie
- L'architecture RISC est une structure interne d'un processeur basé sur un langage assembleur comportant un nombre minimum d'instructions simples de taille fixe (3 adresses) et un certain nombre de registres.
- Le processeur MIPS R3000 est un processeur 32 bits industriel
- Contrairement aux langages évolués (C, Java, etc…) MIPS n'utilise pas de variables. **Pourquoi pas ?** Conserver le hardware aussi simple que possible
- Les opérandes de MIPS sont les registres
- Un nombre limité (32) d'emplacements spéciaux, construits directement dans le hardware
- Les opérations ne s'appliquent qu'aux registres

Avantage : l'accès des registres est très rapide (moins de 10s).

Inconvénient : les registres faisant partie du processeur, ils sont en nombre prédéterminé

- Un des enjeux de la programmation MIPS est l'utilisation efficace des registres
  - 32 registres en MIPS
  - Pourquoi 32 ? Moins il y en a, plus ça va vite
  - La taille de chaque registre MIPS est de 32 bits
  - Un groupe de 32 bits est appelé un mot de MIPS

5.5.2 Registres du MIPS R3000

- Le processeur MIPS R3000 possède 32 registres généraux visibles du logiciel c'est-à-dire dont la valeur peut être modifiée par les instructions.
- Chaque registre est connu par son numéro, qui varie entre 0 et 31, et est préfixé par un $.
- Par exemple, le registre 31 sera noté $31 dans l'assembleur.
- En dehors du registre $0, tous les registres sont identiques du point de vue de la machine.
- Le processeur MIPS R3000 possède deux modes de fonctionnement : utilisateur/superviseur.

  Registres Protégés et Registres non Protégés

$0	Vaut zéro en lecture, non modifié par écriture
$1	Réservé à l'assembleur. Ne doit pas être employé dans les programmes utilisateur
$2	Valeur de retour des fonctions
$3, $4	Argument des syscall
$5, ..., $26	Registres de travail à sauver
$27, ..., $28	Registres réservés aux procédures noyau. Ils ne doivent pas être employés dans les programmes utilisateur
$29	Pointeur de pile
$31	Adresse de retour d'appel de fonction

Tableau5.1 :Registres protégés et Registres non protégés

5.5.2.1 Registres non protégés du MIPS R3000

Le processeur possède **35 registres** manipulés par les instructions standards c'est à dire les instructions qui peuvent s'exécuter aussi bien en mode utilisateur qu'en mode superviseur.

- R_i ($0 \leq i \leq 31$) 32 registres généraux : Ces registres sont directement adressés par les instructions, et permettent de stocker des résultats de calculs intermédiaires.

- **R_0** est un registre particulier appelé Zéro Il contient la valeur zéro même après une écriture (l'écriture ne modifie pas son contenu) .

- **R_{31}** est utilisé par les instructions d'appel de procédures (instructions BGEZAL, BLTZAL, JAL et JALR) pour sauvegarder l'adresse de retour.

- **PC**Program Counter : Ce registre contient l'adresse de l'instruction en cours d'exécution.

- **HI** et **LO** Registres pour la multiplication ou la division Ces deux registres 32 bits sont utilisés pour stocker le résultat d'une multiplication ou d'une division, qui est un mot de 64 bits

5.5.2.2 Registres protégés du MIPS R3000

En pratique, cette version du processeur MIPS R3000 en utilise 4 pour la gestion des interruptions et des exceptions

1) **SR** Registre d'état (Status Register). Il contient en particulier le bit qui définit lemode : superviseur ou utilisateur, ainsi que les bits de masquage des interruptions.
   ➔**Ce registre possède le numéro 12**
2) **CR** Registre de cause (Cause Register). En cas d'interruption ou d'exception, son contenu définit la cause pour laquelle on fait appel au programme de traitement des interruptions et des exceptions.
   ➔**Ce registre possède le numéro 13**
3) **EPC Registre** d'exception (Exception Program Counter). Il contient l'adresse de retour (PC + 4) en cas d'interruption. Il contient l'adresse de l'instruction fautive en cas d'exception (PC).
4) **BAR Registre**d'adresseillégale (Bad Address Register). Cas d'exception de type "adresse illégale", il contient la valeur de l'adresse mal formée.
   ➔**Ce registre possède le numéro 8**

5.5.3 Commentaires en MIPS

En langage **MIPS**, il est possible d'ajouter des commentaires afin d'expliquer lefonctionnement du programme et de rendre le code plus lisible. Les commentaires ne sont pas exécutés par le processeur ; ils servent uniquement aux programmeurs pour documenter le code. Pour écrire un commentaire en MIPS, on utilise le symbole #. Tout le texte qui suit ce symbole jusqu'à la fin de la ligne est considéré comme un commentaire et sera ignoré par l'assembleur.

Par exemple :add $t0, $t1, $t2 # additionne les valeurs de $t1 et $t2
Dans cet exemple, la partie après # explique simplement l'instruction.

Remarque :

Il est important de noter que les commentaires en MIPS ne peuvent pas s'étendre automatiquement sur plusieurs lignes : chaque ligne de commentaire doit commencer par #. Cela diffère du langage **C**, où les commentaires peuvent être écrits sous la forme /*commentaire */, ce qui permet d'écrire un commentaire sur plusieurs lignes.

Par exemple :

```
/* Ceci est un commentaire
qui peut s'étendre
sur plusieurs lignes */
```

D'autres langages de programmation utilisent également des symboles différents pour les commentaires. Par exemple, en **Python**, on utilise aussi le symbole # pour écrire un commentaire sur une seule ligne :

```
# Ceci est un commentaire en Python
x=10
```

En **Java** ou **JavaScript**, on peut écrire des commentaires sur une seule ligne avec // ou sur plusieurs lignes avec /* ... */ :

```
// Commentaire sur une seule ligne
```

```
/* Commentaire
   sur plusieurs lignes */
```

Ainsi, bien que le principe des commentaires soit le même dans la plupart des langages — expliquer le code sans influencer son exécution — la syntaxe utilisée pour les écrire peut varier d'un langage à un autre.

5.6 Organisation de la mémoire

Dans l'architecture MIPS R3000, l'espace adressable est divisé en deux segments :

- Le segment utilisateur, (adr 31 = 0)
- Le segment noyau (ou segment système) (adr 31 = 1)

➢ Quand le processeur est en mode superviseur, les 2 segments sont accessibles.

➢ Quand le processeur est en mode utilisateur, seul le segment utilisateur est accessible.

Un programme utilisateur utilise généralement trois sous-segments (appelés sections) dans le segment utilisateur :

La section text :contient le code exécutable en mode utilisateur. Elle est implantée conventionnellement à l'adresse 0x00400000.

La section data :contient les données globales manipulées par le programme utilisateur. Elle est implantée conventionnellement àl'adresse 0x10000000.

La section stack :contient la pile d'exécution du programme utilisateur. Elle est implantée conventionnellement à l'adresse 0x7FFFF000.

Segment Noyau	**Resevé**	0xFFFFFFFF 0xFFFFF000
	Kstack	0xFFFFEFFF
	Kdata	0xC0000000
	Ktext	0xBFFFFFFF 0x80000000
Segment utilisateur	**Resevé**	0x7FFFFFFF 0x7FFFF000
	stack	0x7FFFEFFF
	data	0x10000000
	text	0x0FFFFFFF 0x00400000
	Resevé	0x00000000

Tableau5.2 : Les segments dans l'architecture MIPS R3000

Remarque : Trois sections sont également définies dans le segment noyau :

- **La section Ktext**
- **La section Kdata**
- **La section Kstack**

5.6.1 Adressage Mémoire

- Toutes les adresses émises par le processeur sont des adresses octets,
- Les adresses sont codées sur 32 bits.
- Les instructions sont codées sur 32 bits.
- Les échanges de données avec la mémoire se font par :

  ➔Mot (4 octets consécutifs),

  ➔Demi-Mot (2 octets consécutifs),

  ➔Octet.

Pour les transferts de mots et **de demi-mots,** le processeur respecte la convention **"littleendian".**

Remarque :

L'adresse d'un mot de donnée ou d'une instruction doit être multiple de 4. L'adresse d'un demi-mot doit être multiple de 2.

5.6.2 Calcul d'adresse

Il existe un seul mode d'adressage, consistant à effectuer la somme entre le contenu d'un registre général Ri, défini dans l'instruction, et d'un déplacement qui est une valeur immédiate signée (positif ou négatif), sur 16 bits, contenue également dans l'instruction :

Adresse = Ri + Déplacement

Exemples :

Lw $12, 13($10) $12←[$10 + 13]

Le registre 12 reçoit le contenu de l'adresse calculée à partir du contenu du registre 10 en ajoutant le déplacement égal à 13

Sw $20, - 60($22) [$22 – 60]←$20

S'il n'y a pas d'entier devant la parenthèse ouvrante, le déplacement est nu

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

5.7 JEU d'INSTRUCTIONS DU MIPS R3000

❖ **Addition registre registre signée (*add)***

Syntaxe : **add $rr, $ri, $rj**

Description : Les contenus des **registres $ri et $rj** sont ajoutés pour former un résultat sur 32 bits qui est placé dans **le registre$rr**

$rr ⟵ $ri + $rj

❖ **Addition registre immédiat signée (*addi)***

Syntaxe **:** **addi $rr, $ri, imm**

Description : La valeur immédiate sur 16 bits subit une extension de signe, et est ajoutée au contenu du **registre $ri** pour former un résultat sur 32 bits qui est placé dans le **registre $rr.**

$rr imm*(16bits)* + $ri

❖ **Soustraction registre registresignée (*sub*)**

Syntaxe : **sub $rr, $ri, $rj**

Description : Le contenu du **registre $rj** est soustrait du contenu du **registre $ri** pour former un résultat sur 32 bits qui est placé dans **le registre $rr**.

$rr ⟵ $ri - $rj

❖ **Division entière signée (*div*)**

Syntaxe **:** **div $ri, $rj**

Description : Le contenu du **registre $ri** est divisé par le contenu du **registre $rj**, le contenu des deux registres étant considéré comme des nombres en complément à deux.

- Le **quotient** de la division est placé dans le **registre spécial lo,**
- Le**reste** dans le **registre spécial hi**.

❖ **Multiplication signée (*mult)***

Syntaxe : **mult $ri, $rj**

Description : Le contenu du **registre $ri** est multiplié par le contenu du **registre $rj**, le contenu des deux registres étant considéré comme des nombres en complément à deux.

Remarque :

Les 32 bits de **poids fort**du résultat sont placés dans le **registre hi**

Les 32 bits de **poids faible** sont placés **dans lo.**

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

$lo ($ri x $rj) 31…0

$hi ($ri x $rj) 63…32

❖ **Branchement inconditionnel registre (jr)**

Syntaxe : **jr $ri**

Description : Le programme saute à l'adresse contenue dans le **registre $ri**. .

pc $ri

❖ **Lecture d'un mot de la mémoire (lw)**

Syntaxe : **Lw $rr, imm($ri)**

Description : L'adresse de chargement est la somme de la valeur immédiate et du contenu du registre **$ri.**

$rr [imm + $ri]

❖ **Écriture d'un mot en mémoire**

Syntaxe : sw $rj, imm($ri)

Description : L'adresse d'écriture est obtenue par la somme de la valeur immédiate et du contenu du registre **$ri.** Le contenu du registre $rj est écrit en mémoire à l'adresse ainsi calculée.

[imm + $ri] $rj

❖ **syscall***Appel à une fonction du système (en mode noyau).*

Syntaxe : syscall

Description : Un appel système est effectué, par un branchement inconditionnel au gestionnaire d'exception.

❖ **asciiz chaîne**

Description : Cette directive place à partir de l'adresse du compteur d'adresse de la section concernée la suite de caractères entre guillemets. Elle ajoute un zéro binaire à la fin de chaque chaîne..

Exemple message: Bonjour Monsieur

asciiz "Bonjour, Monsieur''

➢ Les APPELS SYSTEMES :

Ils permettent l'interaction avec le système d'exploitation.

- Le MIPS communique avec le système par la commande **syscall**
- La fonction utilisée est déterminée selon la valeur du registre $V0

Le tableau suivant résume les principaux appels

Service	N°d'appel	Arguments	Effet
Print_int	1	$a0 = integer	Imprime l'entier contenu dans a0
Print_string	4	$a0 = string	Imprime la chaine en a0 jusqu'à'/000'
Read_int	5		Integer (in $v0) Li un entier et le place dans v0
exit	10		Arrêt du programme en cours d'exécution

Tableau5.3 : les principaux appels

Quelques exemples dujeu d'instruction du MIPS R3000

1°/ Lecture

- **Lw $rt, déplacement ($rs) ⟹ $rt ← [$rs]+ déplacement**

(*Le déplacement peut êtrepositif ou négatif ou nul)*

Exemple : lw $12, 13 ($10) ⟹ $12 ← [[$10] +13]

- **Li $rt, constante ⟹ $rt ← constante**

2°/ Stockage

- **sw $rt, déplacement ($rs) ⟹ $rt ← [$rs] + déplacement**

  Exemple : sw $20, -60 ($22) ⟹ [$22] – 66 ← $20

- **sw $rt, adradr [⟹ ←**
- **move $rt, source ⟹ $rt [source]**

3°/ Addition des entiers

- **Add $rd, $rs, $rt, ⟹ $rd [$rs]+ [$rt]]**
- **Addi $rd, $rs, constante, ⟹ $rd ← [[$rs]+ constante]**

4°/ Soustraction

- **Sub $rd, $rs, $rt, ⟹ $rd ← [[$rs] - [$rt]]**

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

5°/ Multiplication

- **MULT $rd, $rs, $rt, ⟹ $rd [[$rs] * [$rt]]**

Remarque :
Si le résultat de la multiplication dépasse 32 bits, Mips utilise deux autres registres : $Hi et $Lo.
Si on veut voir le résultat, il faut écrire dans le programme les instructions suivantes:

mflo $rt2 ⟹ [$rt2] pour partie 1
mfhi $rt3 ⟹ [$rt3] pour partie 2

6°/ Division

- **Div $rd, $rs, $rt ⟹ $rd ⟵ [[$rs] / [$rt]]**

Remarque *:*
Si on veut récupérer le résultat, il faut écrire dans le programme les instructions suivantes *:*

mflo $rt2 ⟹ [$lo] = quotient
mfhi $rt3 ⟹ [$Hi] = reste

7°/ Saut Conditionnel

On six instructions de saut conditionnel :

- Branch on equalBeq = 0 ⟹
- Branch on not equal ⟹ Bne≠ 0
- Branch on less than ⟹ Blt< 0
- Branch on less or equal than ⟹ Ble ≤ 0
- Branch on greater than ⟹ Bgt> 0
- Branch on greater or equal than ⟹ Bge≥ 0

Exemple : Beq **$r1, $r2, adr** ⟹ saut à l'adresse **adr** si**[$r1] = [$r2]**

8°/ Saut Inconditionnel

On quatre instructions de saut inconditionnel :

- **Jump ⟹ J**
- **Jump and link ⟹ Jal**
- **Jump register ⟹ Jr**
- **Jump and linkregisterJa⟹**

Exemple :
J adr*saut à l'adresse adr*
Jal adr *saut à l'adresse adr avec sauvegarde du [***$PC***] dans le registre* **$ra**
Jr $t0 *saut à l'adresse contenue dans* **$t0**
Jalr $t0 *saut à l'adresse contenue dans* **$t0** *et sauvegarde du [***$PC***] dans le registre* **$ra**

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

9°/ Appel système

- Le MIPS communique avec le système par la commande **syscall**
- La fonction utilisée est déterminée selon la valeur du registre $V0

$V0	Commande	Arguments	résultat
1	Print	*[**$a0**]=Entier à lire*	
4	Print-string	*[**$a0**]=adresse de la chaine de caractère*	
5	Read		*[**$V0**]=entier lu*
8	Read-string	*[**$a0**]= adresse de la chaine de caractère*	
10	Exit		

Tableau5.4 : Appel système

PR .H FIZAZI
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF

CHAPITRE6

Architectures des processeurs récents

L'étude de l'architecture des processeurs récents permet ainsi de comprendre les principes et les technologies qui déterminent les performances des systèmes informatiques actuels.

L'évolution rapide de l'informatique et des applications numériques a conduit au développement de processeurs de plus en plus performants et sophistiqués. Les architectures des processeurs récents intègrent de nombreuses innovations visant à améliorer la vitesse d'exécution, l'efficacité énergétique et la capacité de traitement des données.

Contrairement aux premières générations de microprocesseurs comme Intel 8086, les processeurs modernes reposent sur des techniques avancées telles que le pipeline, l'exécution superscalaire, le multicœur et la gestion optimisée de la mémoire cache. Ces architectures modernes permettent d'exécuter plusieurs instructions simultanément et d'optimiser l'utilisation des ressources matérielles.

6.1 Les architectures des processeurs récents

Les processeurs constituent le cœur des systèmes informatiques modernes. Ils sont responsables de l'exécution des instructions et du traitement des données dans les ordinateurs, les smartphones et les serveurs. Avec l'évolution rapide des technologies et l'augmentation des besoins en performance, les architectures des processeurs ont connu des améliorations importantes.

6.1.1 Concept de multi-cœur

Un processeur **multi-cœur** est un processeur qui contient plusieurs unités de traitement appelées **cœurs** dans une seule puce. Chaque cœur est capable d'exécuter des instructions de manière indépendante, comme s'il s'agissait d'un processeur séparé. Cette architecture permet d'exécuter plusieurs tâches simultanément, ce qui améliore considérablement les performances du système. Par exemple, un ordinateur peut exécuter un programme, lire une vidéo et télécharger des fichiers en même temps grâce aux différents cœurs du processeur.

➢ Avantages des architectures multi-cœurs

- Amélioration des performances globales (exécution parallèle)
- Meilleure efficacité énergétique (chaque cœur peut fonctionner à plus basse fréquence)
- Idéal pour les applications multithreadées (jeux, rendu 3D, IA)

➢ Défis et limitations

- Complexité du développement logiciel (programmation parallèle, synchronisation)
- Accès concurrent à la mémoire partagée (problèmes de cohérence de cache)
- Diminution du rendement au-delà d'un certain nombre de cœurs

➢ Gestion de la mémoire cache dans les architectures récentes

Les caches sont des mémoires très rapides utilisées pour réduire le temps d'accès aux données fréquemment utilisées. Ils sont organisés en niveaux hiérarchiques (L1, L2, L3).

➢ Hiérarchie typique des caches

- **Cache L1 :** très rapide, petite taille (16–64 Ko), dédié à chaque cœur
- **Cache L2 :** plus grand (256 Ko–1 Mo), souvent dédié ou semi-partagé
- **Cache L3** : partagé entre tous les cœurs (4 Mo – 64 Mo)

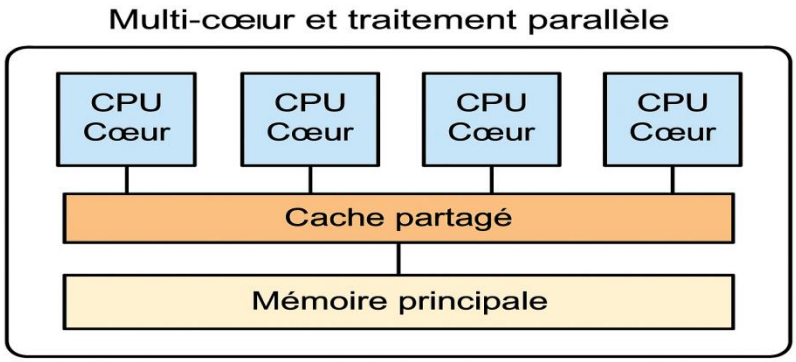


Figure 6.1: Hiérarchie type des caches

6.1.2 Des exemples concrets de processeurs multi-cœurs :

Nous allons présenterquelques exemples concrets de processeurs multi-cœurs que nous pouvons trouver dans nos appareils en 2026, classés par usage.

1. Dans les Smartphones (Le standard : 8 cœurs)

Presque tous les smartphones modernes (Android et iPhone) ont **8 cœurs**, mais organisés différemment.

- **Apple A18 Pro (iPhone 16/17)**
  - **Total :** 6 cœurs.
  - **Répartition :** 2 cœurs de performance (très rapides) + 4 cœurs d'efficacité (pour économiser la batterie).
  - *Usage :* Gère les jeux lourds et la photo avec les 2 gros cœurs, et garde le téléphone en veille ou gère les messages avec les 4 petits.
- **Qualcomm Snapdragon 8 Gen 4 / 5 (Samsung S25, Xiaomi, etc.)**
  - **Total :** 8 cœurs.
  - **Répartition :** Souvent 2 cœurs "Ultra Performance" + 4 cœurs "Performance" + 2 cœurs "Efficiency".
  - *Usage :* Permet de faire du multitâche intensif (streaming + jeu + téléchargement) sans surchauffe.

2. Dans les PC Portables (L'équilibre Performance/Batterie)

C'est ici que l'architecture **hybride** est la plus visible.

- **Intel Core Ultra 7 "Series 2" (Lunar Lake)**
  - **Total :** 8 cœurs physiques.

- **Répartition :** 4 cœurs Performance (Lion Cove) + 4 cœurs Efficiency (Skymont).
- *Exemple concret :* Si vous montez une vidéo sur Premiere Pro, les 4 gros cœurs travaillent à fond. Si vous tapez simplement un texte Word pendant que Spotify tourne, seuls les 4 petits cœurs s'activent pour préserver l'autonomie de la batterie.

- **AMD Ryzen AI 9 HX 370 (Série Strix Point)**
  - **Total :** 12 cœurs physiques.
  - **Répartition :** 4 cœurs Performance (Zen 5) + 8 cœurs Efficiency (Zen 5c).
  - *Particularité :* AMD a misé sur un très grand nombre de petits cœurs efficaces pour gérer le multitâche de fond (notifications, mises à jour, IA locale) tout en gardant 4 cœurs puissants pour les applications lourdes.
- **Apple M3 / M4 / M5 (MacBook Air & Pro)**
  - **Version de base (ex: MacBook Air) :** 8 cœurs (4 Performance + 4 Efficiency).
  - **Version Pro (ex: MacBook Pro 14") :** Jusqu'à 14 ou 16 cœurs (10 ou 12 Performance + 4 Efficiency).
  - *Usage :* Sur un Mac Studio avec la puce M2 Ultra (qui fusionne deux puces), on peut avoir jusqu'à **24 cœurs**, permettant de rendre des films en 3D en temps réel.

3. Dans les PC de Bureau (La puissance brute)

Ici, on cherche le nombre maximum de cœurs pour le gaming extrême, le montage 4K/8K ou la compilation de code.

- **Intel Core i9-14900K / Core Ultra 9 (Arrow Lake)**
  - **Total :** 24 cœurs physiques.
  - **Répartition :** 8 cœurs Performance + 16 cœurs Efficiency.
  - *Pourquoi autant ?* Les 16 petits cœurs gèrent Windows, les antivirus, les navigateurs en arrière-plan, laissant les 8 gros cœurs libres à 100% pour le jeu vidéo ou le calcul lourd.
- **AMD Ryzen 9 9950X (Architecture Zen 5)**
  - **Total :** 16 cœurs physiques.
  - **Répartition :** 16 cœurs "pleine puissance" (tous identiques).
  - *Différence avec Intel :* Contrairement à l'approche hybride d'Intel, AMD ici utilise 16 cœurs identiques très puissants. C'est idéal pour les professionnels qui

ont besoin que *toutes* les tâches soient rapides (ex: rendu 3D Blender, serveur virtuel).

- *Note :* Grâce au "SMT" (SimultaneousMulti-Threading), ces 16 cœurs peuvent traiter 32 flux de données en même temps.

- **AMD Threadripper 7995WX (Station de travail extrême)**
  - **Total :** 96 cœurs physiques !
  - **Usage :** Ce n'est pas pour les gamers. C'est pour les studios de cinéma (effets spéciaux), la recherche scientifique (simulations climatiques) ou l'entraînement de modèles d'IA massifs. Imaginez 96 chefs dans la cuisine !

CONCLUSION GÉNÉRALE

Au terme de ce polycopié, il apparaît clairement que la maîtrise de l'architecture des ordinateurs ne se limite pas à la simple connaissance de composants matériels isolés, mais repose sur la compréhension profonde de leur interaction dynamique au sein d'un système cohérent. Ce cours a permis de poser les jalons fondamentaux nécessaires à tout futur ingénieur pour appréhender la complexité croissante des systèmes informatiques modernes.

Nous avons d'abord établi que l'organisation générale d'un ordinateur, qu'elle suive le modèle de Von Neumann ou celui de Harvard, repose sur un équilibre subtil entre la puissance de calcul du processeur, la rapidité d'accès de la hiérarchie mémoire et l'efficacité des mécanismes d'entrée/sortie. L'étude détaillée de l'architecture interne du processeur a ensuite mis en lumière le rôle central de l'unité de commande et de l'unité arithmétique et logique, ainsi que l'importance cruciale des registres et des bus dans le flux de l'information.

Par ailleurs, l'analyse du langage machine et des modes d'adressage a offert une vision concrète du lien indissociable entre le matériel et le logiciel. Elle a démontré comment chaque instruction exécutée par l'utilisateur est, in fine, traduite en une série d'opérations élémentaires régies par la logique électronique de la machine. Enfin, l'étude de cas du processeur Intel 8086 a servi de pont essentiel entre la théorie abstraite et la réalité technologique, illustrant comment les concepts fondamentaux s'incarnent dans une architecture réelle ayant marqué l'histoire de l'informatique.

En somme, les connaissances acquises dans ce module constituent le socle indispensable pour aborder des disciplines plus avancées telles que les systèmes d'exploitation, la compilation, l'optimisation de code ou encore la conception de systèmes embarqués. Comprendre « comment ça marche » à ce niveau fondamental permet non seulement de mieux exploiter les ressources matérielles actuelles, mais aussi d'anticiper les évolutions futures, telles que le multi-cœur, l'intégration de l'IA ou les nouvelles architectures RISC et hybrides.

Pour l'étudiant ingénieur, cette compréhension de l'architecture n'est pas une fin en soi, mais un outil puissant lui permettant de concevoir des solutions logicielles plus efficaces, de diagnostiquer des problèmes complexes et d'innover dans un domaine où la frontière entre le hardware et le software devient chaque jour plus ténue.

RÉFÉRENCES

➢ **Livres :**

- **Structured Computer Organization**, Auteur: Andrew S. Tanenbaum, Édition : Pearson, 2012
- **Computer Organization and Design**: The Hardware/Software Interface; Auteurs: David A. Patterson, John L.Hennessy Edition : Morgan Kaufmann.
- **Architecture et technologie des ordinateurs** Cours et exercices corrigés - 6e édition, Paolo Zanella, Yves Ligier,Emmanuel Lazard, 2018
- **The Intel Microprocessors:** 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro Processor,Pentium II, Pentium III, Pentium 4, and Core2 with 64-bit Extensions, Auteur : Barry B. BreyÉdition : Pearson
- **Tanenbaum, A. S., & Austin, T.** (2013). *Structured Computer Organization* (6th ed.). Pearson.
- **Stallings, W.** (2016). *Computer Organization and Architecture* (10th ed.). Pearson.
- **Muller, J.-P.** (2006). *Architecture des ordinateurs : Cours et exercices corrigés*. Dunod.
- **Brey, B. B**. (2009). The Intel Microprocessors: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Pro Processor (8th ed.). Pearson Prentice Hall.
- **Hall, K. J**. (2006). Microprocessors and Interfacing: Programming and Hardware. McGraw-Hill Education.
- **Intel Corporation**. (1980s-1990s). iAPX 86, 88, 186 and 188 User's Manual. IntelDocumentation.
- **Apple Inc.** (2025). *Apple M5 Chip TechnicalSpecifications*. [En ligne]. Disponible sur : apple.com/mac/chips.
- **Intel Corporation.** (2025). *Intel Core Ultra Processors (Series 2 - Lunar Lake/Arrow Lake) Architecture Overview*. White Paper.
- **Advanced Micro Devices (AMD).** (2025). *AMD Ryzen 9000 Series and Zen 5 Core Architecture*. Technical Brief.
- **ARM Holdings.** (2024). *ARMv9 Architecture Reference Manual*.
- **Waterman, K., &Asanović, K.** (2024). *The RISC-V Reader: An Open Architecture Atlas* (2nd ed.). Strawberry Canyon LLC.

➢ **Autres Sources Pédagogiques :**

- EMU8086 -MICROPROCESSOR EMULATOR https://emu8086-microprocessor-emulator.fr.softonic.com/

- NASM (Netwide Assembler) https://www.nasm.us/
- MASM (Microsoft Macro Assembler) https://learn.microsoft.com/en-us/cpp/assembler/masm/microsoft-macroassembler-reference?view=msvc-17
- GAS (GNU Assembler) https://www.gnu.org/software/binutils/
- YASM https://yasm.tortall.net/
- AnandTech& Tom's Hardware. (2025). Deep Dives: CPU Microarchitecture Analysis. [Sites web].
- Coursera / edX. Cours : "Build a Modern Computer from First Principles: Nand to Tetris" (Shimon Schocken & Noam Nisan).

RÉSUMÉ

Ce polycopié est destiné aux étudiants de première année ingénieur informatique. Il vise à expliquer comment les composants matériels collaborent pour traiter l'information, en alliant théorie et exemples pratiques.

, il couvre quatre axes essentiels :

Les Fondamentaux : La structure de base d'un ordinateur (CPU, mémoire, E/S) et les modèles d'organisation de référence (Von Neumann et Harvard).

Le Cœur du Système : L'architecture interne du processeur, détaillant le rôle de l'unité de commande, de l'UAL, des registres, du cache et des bus, ainsi que le cycle d'exécution des instructions.

L'Interface Matériel-Logiciel : Le décodage du langage machine, des modes d'adressage et la traduction des programmes au niveau le plus bas.

Études de Cas : L'analyse concrète du processeur historique Intel 8086, mise en perspective avec un aperçu des architectures modernes.

Ce cours fournit le socle technique indispensable pour aborder des disciplines avancées comme les systèmes d'exploitation, l'optimisation de code et les systèmes embarqués. Nous invitons les étudiants à exploiter activement ce document, en cours comme en auto-apprentissage, pour consolider leurs acquis et réussir leurs évaluations.