

## Chapitre 8 : L'architecture de base des ordinateurs

- Introduction
- Architecture de base d'une machine
  - La Mémoire Centrale
  - UAL ( unité arithmétique et logique )
  - UC ( unité de contrôle ou de commande )
- Jeu d'instructions , Format et codage d'une instruction
- Modes d'adressage
- Étapes d'exécution d'un instruction

1

## Objectifs

- Comprendre l'architecture d'une machine von newman.
- Comprendre les étapes de déroulement de l'exécution d'une instruction.
- Comprendre le principe des différents modes d'adressage.

2

### 1. Introduction

- Un programme est un ensemble d'instructions exécutées dans un ordre bien déterminé.
- Un programme est exécuté par un processeur ( machine ).
- Un programme est généralement écrit dans un langage évolué (Pascal, C, VB, Java, etc.).
- Les instructions qui constituent un programme peuvent être classifiées en 4 catégories :
  - Les Instructions d'affectations : permet de faire le transfert des données
  - Les instructions arithmétiques et logiques.
  - Les Instructions de branchement ( conditionnelle et inconditionnelle )
  - Les Instructions d'entrées sorties.

### 1. Introduction

- Pour exécuter un programme par une machine, on passe par les étapes suivantes :
  1. Édition : on utilise généralement un éditeur de texte pour écrire un programme et le sauvegarder dans un fichier.
  2. Compilation : un compilateur est un programme qui convertit le code source ( programme écrit dans un langage donné ) en un programme écrit dans un langage machine ( binaire ). Une instruction en langage évolué peut être traduite en plusieurs instructions machine.
  3. Chargement : charger le programme en langage machine dans mémoire afin de l'exécuter .

4

- Comment s'exécute un programme dans la machine ?
- Pour comprendre le mécanisme d'exécution d'un programme → il faut comprendre le mécanisme de l'exécution d'une instruction .
- Pour comprendre le mécanisme de l'exécution d'une instruction → il faut connaître l'architecture de la machine ( processeur ) sur la quelle va s'exécuter cette instruction.

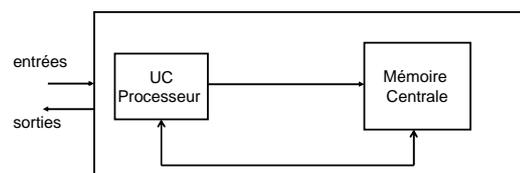
5

### 2. Architecture matérielle d'une machine ( architecture de Von Neumann )



L'architecture de Von Neumann est composée :

- D'une mémoire centrale,
- D'une unité centrale UC , CPU (Central Processing Unit), processeur , microprocesseur.
- D'un ensemble de dispositifs d'entrées sorties pour communiquer avec l'extérieur.
- Cette architecture est la base des architectures des ordinateurs.



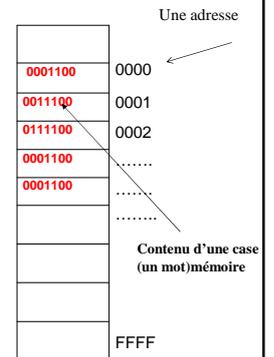
6

## 2.1 La mémoire centrale

- 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 ( performances d'une machine ).

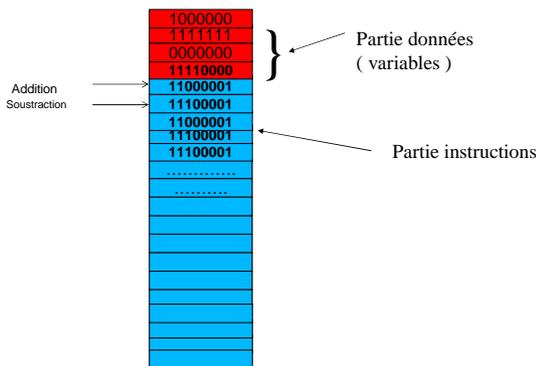
7

- La mémoire centrale peut être vue 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 les données utilisées par les programmes.



8

## Structure d'un programme en MC

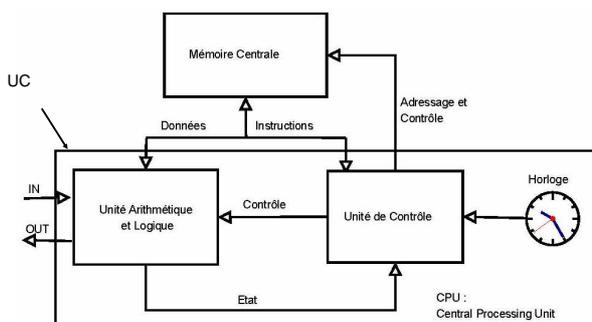


## 2.2 L'Unité Centrale ( UC )

- L'unité centrale (appelée aussi processeur , microprocesseur ) a pour rôle d'exécuter les programmes.
- L'UC est composée d'une unité arithmétique et logique (UAL) et d'une unité de contrôle.
  - L'unité arithmétique et logique réalise les opérations élémentaires (addition, soustraction, multiplication, . . . ) .
  - L'unité de commande contrôle les opérations sur la mémoire (lecture/écriture) et les opérations à réaliser par l'UAL selon l'instruction en cours d'exécution.

10

## Architecture matérielle d'une machine Von Neumann



## 2.2.1 L'UAL

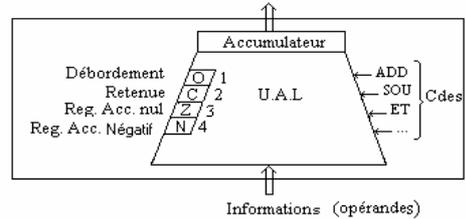
- L'unité arithmétique et logique réalise une opération élémentaire (addition, soustraction, multiplication, . . . ) .
- L'UAL regroupe les circuits qui assurent les fonctions logiques et arithmétiques de bases ( ET,OU,ADD,SUS,.....).
- L'UAL comporte un registre accumulateur ( ACC ) : c'est un registre de travail qui sert à stocker un opérande (données) au début d'une opération et le résultat à la fin.

12

- L'UAL comporte aussi un **registre d'état** : Ce registre nous indique l'état du déroulement de l'opération .
- Ce registre est composé d'un ensemble de **bits**. Ces bits s'appels **indicateurs** (drapeaux ou flags).
- Ces indicateurs sont **mis à jours ( modifiés )** après la fin de l'exécution d'une opération dans l'UAL.
- Les principaux indicateurs sont :
  - Retenue : ce bit est mis à 1 si l'opération génère une retenue.
  - Signe : ce bit est mis à 1 si l'opération génère un résultat négative.
  - Débordement : ce bit est mis à 1 s'il y a un débordement.
  - Zero : ce bit est mis à 1 si le résultat de l'opération est nul.

13

## Schéma d'une UAL



14

## 2.2.2 Unité de contrôle

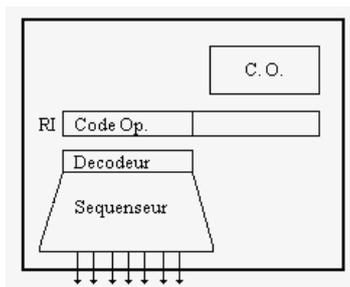
- Le rôle de l'unité de contrôle (ou unité de commande ) est de :
  - **coordonner** le travail de toutes les autres unités ( UAL , mémoire,..... )
  - et d'assurer la **synchronisation** de l'ensemble.
- Elle assure :
  - la **recherche** ( lecture ) de l'instruction et des données à partir de la mémoire,
  - le **décodage** de l'instruction et l'exécution de l'instruction en cours
  - et **prépare** l'instruction suivante.

15

- L'unité de contrôle comporte :
  - Un **registre instruction (RI)** : contient l'instruction en cours d'exécution. Chaque instruction est décodée selon son code opération grâce à un décodeur.
  - Un registre qui s'appel **compteur ordinal (CO)** ou le **compteur de programme (CP)** : contient l'adresse de la prochaine instruction à exécuter ( pointe vers la prochaine instruction à exécuter ). Initialement il contient l'adresse de la première instruction du programme à exécuter.
  - Un **séquenceur** : il organise ( synchronise ) l'exécution des instruction selon le rythme de l'horloge, il génère les signaux nécessaires pour exécuter une instruction.

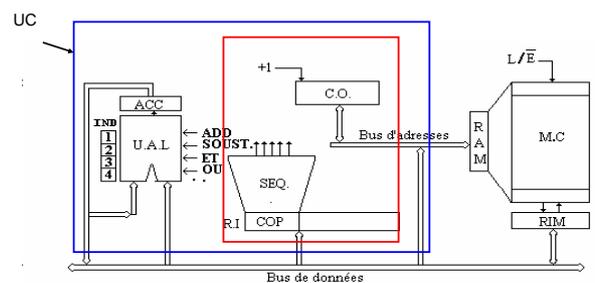
16

## Schéma d'une UC



17

## Schéma détaillé d'une machine



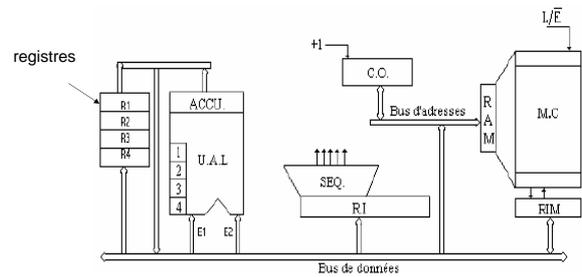
18

## Remarque

- Le microprocesseur peut contenir **d'autres registres** autre que CO, RI et ACC.
- Ces registres sont considérés comme une **mémoire interne** ( registre de travail ) du microprocesseur.
- Ces registres sont plus rapide que la mémoire centrale , mais le nombre de ces registre est **limité**.
- Généralement ces registres sont utilisés pour sauvegarder les données avant d'exécuter une opération.
- Généralement la taille d'un registre de travail est égale à la taille d'un mot mémoire

19

## Une machine avec des registres de travail



20

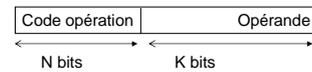
## 3. Jeu d'instructions

- Chaque microprocesseur possède un **certain nombre limité** d'instructions qu'il peut exécuter. Ces instructions s'appelles **jeu d'instructions**.
- Le jeu d'instructions décrit l'ensemble des opérations élémentaires que le microprocesseur peut exécuter.
- Les instructions peuvent être classifiées en 4 catégories :
  - Instruction d'affectation : elle permet de faire le transfert des données entre les registres et la mémoire
    - Écriture : registre → mémoire
    - Lecture : mémoire → registre
  - Les instructions arithmétiques et logiques ( ET , OU , ADD,....)
  - Instructions de branchement ( conditionnelle et inconditionnelle )
  - Instructions d'entrées sorties.

21

## 3.1 Codage d'une instruction

- Les **instructions et leurs opérandes** ( données ) sont stocké dans la mémoire.
- La taille d'une instruction ( nombre de bits nécessaires pour la représenter en mémoire ) dépend du type de l'instruction et du type de l'opérande.
- L'instruction est découpée en deux parties :
  - Code opération** ( code instruction ) : un code sur N bits qui indique quelle instruction.
  - La champs opérande** : qui contient la donnée ou la référence ( adresse ) à la donnée.



•Le format d'une instruction peut ne pas être le même pour toutes les instructions.

•Le champs opérande peut être découpé à sont tours en **plusieurs champs**

22

## Machine à 3 adresses

- Dans ce type de machine pour chaque instruction il faut préciser :
  - l'adresse du premier opérande
  - du deuxième opérande
  - et l'emplacement du résultat

Code opération	Opérande1	Opérande2	Résultat
----------------	-----------	-----------	----------

Exemple :

ADD A,B,C ( C ← B+C )

- Dans ce type de machine la taille de l'instruction est grand .
- Pratiquement il n'existent pas de machine de ce type.

23

## Machine à 2 adresses

- Dans de type de machine pour chaque instruction il faut préciser :
  - l'adresse du premier opérande
  - du deuxième opérande ,
- l'adresse de résultat est implicitement l'adresse du deuxième opérande .

Code opération	Opérande1	Opérande2
----------------	-----------	-----------

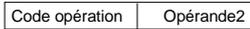
Exemple :

ADD A,B ( B ← A + B )

24

## Machine à 1 adresses

- Dans ce type de machine pour chaque instruction il faut préciser uniquement l'adresse du **deuxième opérande**.
- Le **premier opérande** existe dans le registre **accumulateur**.
- Le **résultat** est mis dans le **registre accumulateur**.



Exemple :  
ADD A (ACC ← (ACC) + A)

Ce type de machine est le plus utilisé.

25

## 4. Mode d'adressage

- Le champ opérande contient **la donnée** ou la **référence** (adresse) à la donnée.
- Le mode d'adressage définit la manière dont le microprocesseur va **accéder à l'opérande**.
- Le code opération de l'instruction comporte un ensemble de bits pour indiquer le **mode d'adressage**.
- Les modes d'adressage les plus utilisés sont :
  - Immédiat
  - Direct
  - Indirect
  - Indexé
  - relatif

26

### 4.1 Adressage immédiat

- L'opérande existe dans le **champs adresse** de l'instruction



Exemple :  
ADD 150

ADD	150
-----	-----

Cette commande va avoir l'effet suivant :  $ACC \leftarrow (ACC) + 150$

Si le registre accumulateur contient la valeur 200 alors après l'exécution son contenu sera égale à 350

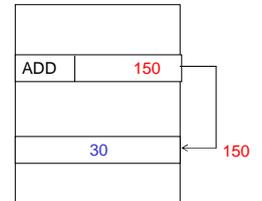
27

### 4.2 Adressage direct

- Le champ opérande de l'instruction contient **l'adresse de l'opérande** (emplacement en mémoire)
- Pour réaliser l'opération il faut le récupérer (lire) l'opérande à partir de la mémoire.  $ACC \leftarrow (ACC) + (ADR)$

Exemple :  
On suppose que l'accumulateur contient la valeur 20.

A la fin de l'exécution nous allons avoir la valeur 50 ( $20 + 30$ )



28

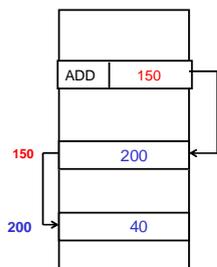
### 4.3 Adressage indirect

- Le champ adresse contient **l'adresse de l'adresse** de l'opérande.
- Pour réaliser l'opération il faut :
  - Récupérer **l'adresse de l'opérande** à partir de la mémoire.
  - Par la suite il faut chercher l'opérande à partir de la mémoire.

$$ACC \leftarrow (ACC) + ((ADR))$$

- Exemple :
- Initialement l'accumulateur contient la valeur 20
- Il faut récupérer l'adresse de l'adresse (150).
- Récupérer l'adresse de l'opérande à partir de l'adresse 150 (la valeur 200)
- Récupérer la valeur de l'opérande à partir de l'adresse 200 (la valeur 40)

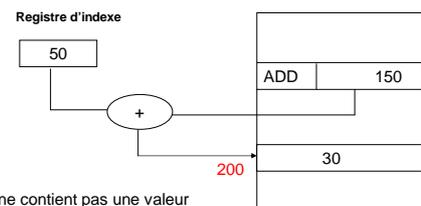
Additionner la valeur 40 avec le contenu de l'accumulateur (20) et nous allons avoir la valeur 60



29

### 4.4 Adressage indexé

- L'adresse effective de l'opérande est relatif à une zone mémoire.
- L'adresse de cette zone se trouve dans un registre spécial (registre indexé).
- Adresse opérande =  $ADR + (X)$



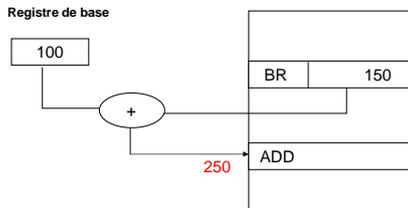
Remarque : si ADR ne contient pas une valeur immédiate alors

$$\text{Adresse opérande} = (ADR) + (X)$$

30

## 4.5 Adressage relatif

- L'adresse effectif de l'opérande est relatif a une zone mémoire.
- L'adresse de cette zone se trouve dans un registre spécial ( registre de base ).
- Ce mode d'adressage est utilisée pour les instructions de branchement.  
Adresse = ADR + (base)



31

## 5. Cycle d'exécution d'une instruction

- Le traitement d'une instruction est décomposé en trois phases :
  - Phase 1 : rechercher l'instruction à traiter et décodage
  - Phase 2 : rechercher de l'opérande et exécution de l'instruction
  - Phase 3 : passer à l'instruction suivante
- Chaque phase comporte un certain nombre d'opérations élémentaires ( microcommandes ) exécutées dans un ordre bien précis ( elle sont générées par le séquenceur ).
- La phase 1 et 3 ne change pas pour l'ensemble des instructions , par contre la phase 2 change selon l'instruction et le mode d'adressage

32

- Exemple 1 : déroulement de l'instruction d'addition en mode immédiat  
 $ACC \leftarrow (ACC) + \text{Valeur}$

- Phase 1 : ( rechercher l'instruction à traiter )
  - Mettre le contenu du CO dans le registre RAM  $RAM \leftarrow (CO)$
  - Commande de lecture à partir de la mémoire
  - Transfert du contenu du RIM dans le registre RI  $RI \leftarrow (RIM)$
  - Analyse et décodage
- Phase 2 : (traitement )
  - Transfert de l'opérande dans l'UAL  $UAL \leftarrow (RI).ADR$
  - Commande de l'exécution de l'opération ( addition )
- Phase 3 : ( passer à l'instruction suivante )
  - $CO \leftarrow (CO) + 1$

33

- Exemple 2 : déroulement de l'instruction d'addition en mode direct  
 $ACC \leftarrow (ACC) + (ADR)$

- Phase 1 : ( rechercher l'instruction à traiter )
  - Mettre le contenu du CO dans le registre RAM  $RAM \leftarrow (CO)$
  - Commande de lecture à partir de la mémoire
  - Transfert du contenu du RIM dans le registre RI  $RI \leftarrow (RIM)$
  - Analyse et décodage
- Phase 2 : ( décodage et traitement )
  - Transfert de l'adresse de l'opérande dans le RAM  $RAM \leftarrow (RI).ADR$
  - Commande de lecture
  - Transfert du contenu du RIM vers l'UAL  $UAL \leftarrow (RIM)$
  - Commande de l'exécution de l'opération ( addition )
- Phase 3 : ( passer à l'instruction suivante )
  - $CO \leftarrow (CO) + 1$

34

- Exemple 3 : Déroulement de l'instruction d'addition en mode indirect  
 $ACC \leftarrow (ACC) + ((ADR))$

- Phase 1 : ( rechercher l'instruction à traiter )
  - Mettre le contenu du CO dans le registre RAM  $RAM \leftarrow (CO)$
  - Commande de lecture à partir de la mémoire
  - Transfert du contenu du RIM dans le registre RI  $RI \leftarrow (RIM)$
  - Analyse et décodage
- Phase 2 : ( décodage et traitement )
  - Transfert de l'adresse de l'opérande dans le RAM  $RAM \leftarrow (RI).ADR$
  - Commande de lecture /\* récupérer l'adresse \*/
  - Transfert du contenu du RIM vers le RAM  $RAM \leftarrow (RIM)$
  - Commande de lecture /\* récupérer l'opérande \*/
  - Transfert du contenu du RIM vers l'UAL  $UAL \leftarrow (RIM)$
  - Commande de l'exécution de l'opération ( addition )
- Phase 3 : ( passer à l'instruction suivante )
  - $CO \leftarrow (CO) + 1$

35