ARCHITECTURE DES ORDINATEURS

2ème Année Informatique

Chapitre3:

Notions sur les instructions d'un ordinateur

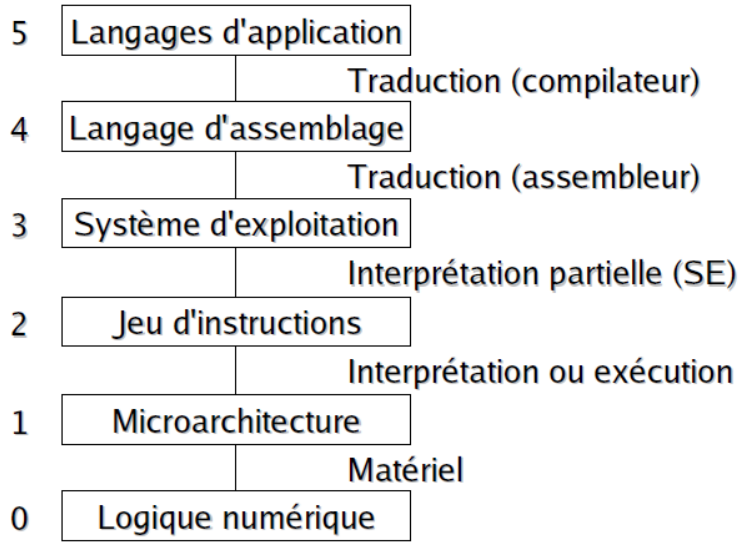
Centre universitaire Mila
2023-2024

1

Introduction

- Les ordinateurs modernes sont conçus comme un ensemble de couches
- Chaque couche représente une abstraction différente, capable d'effectuer des opérations et de manipuler des objets spécifiques
- L'ensemble des types de données, des opérations, et des fonctionnalités de chaque couche est appelée son architecture
- L'étude de la conception de ces parties est appelée « architecture des ordinateurs »

Machines multi-couches actuelles



3

Couche logique numérique

- Les objets considérés à ce niveau sont les portes logiques, chacune construite à partir de quelques transistors
- Chaque porte prend en entrée des signaux numériques (0 ou 1) et calcule en sortie une fonction logique simple (ET, OU, NON)
- De petits assemblages de portes peuvent servir à réaliser des fonctions logiques telles que mémoire, additionneur, ainsi que la logique de contrôle de l'ordinateur

4

Couche micro-architecture

- On dispose à ce niveau de plusieurs registres mémoire et d'un circuit appelé UAL (Unité Arithmétique et Logique, ALU) capable de réaliser des opérations arithmétiques élémentaires
- Les registres sont reliés à l'UAL par un chemin de données permettant d'effectuer des opérations arithmétiques entre registres
- Le contrôle du chemin de données est soit micro-programmé, soit matériel

5

Couche jeu d'instruction

- La couche de l'architecture du jeu d'instructions (Instruction Set Architecture, ISA) est définie par le jeu des instructions disponibles sur la machine
- Ces instructions peuvent être exécutées par microprogramme ou bien directement

6

Couche système d'exploitation

- Cette couche permet de bénéficier des services offerts par le système d'exploitation
- La plupart des instructions disponibles à ce niveau sont directement traitées par les couches inférieures
- Les instructions spécifiques au système font l'objet d'une interprétation partielle (appels système)

7

Couche langage d'assemblage

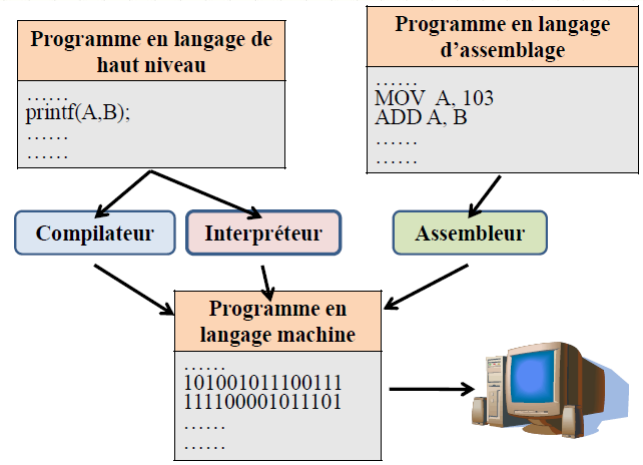
- Offre une forme symbolique aux langages des couches inférieures
- Permet à des humains d'interagir avec les couches inférieures

8

Couche langages d'application

- Met à la disposition des programmeurs d'applications un ensemble de langages adaptés à leurs besoins
- Langages dits « de haut niveau »

Programmation



Langages de programmation

- Langage formel servant à l'écriture de programmes exécutables par l'ordinateur
- **Langages de bas niveau :**
  - Langages machine
  - Langages d'assemblage
- **Langages de haut niveau ou évolués**
  - Fortran, Basic, Pascal, C, C++, Visual Basic, Visual C++, Java...

11

Langages de bas niveau

- Etroitement liés à l'ordinateur utilisé
- Difficiles à lire et à écrire (des 0 et 1)
- Fortement exposés aux erreurs
- Programmes directement exécutables par la machine ou sont à assembler

12

Langages de haut niveau

- Indépendants de l'ordinateur (programmes portables)
- Faciles à utiliser (Instructions proches de la langue naturelle)
- Programmes facilement compréhensibles mais sont à compiler ou à interpréter

13

Langages d'assemblage

Le langage assemblage est le langage le plus « proche » du langage machine. Il est composé par des instructions en général assez rudimentaires que l'on appelle des mnémoniques. Ce sont essentiellement des opérations de transfert de données entre les registres et l'extérieur du microprocesseur (mémoire ou périphérique), ou des opérations arithmétiques ou logiques. Chaque instruction représente un code machine différent. Chaque microprocesseur peut posséder un assembleur différent.

14

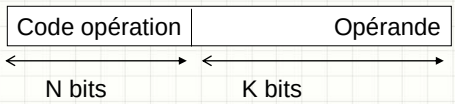
Jeu d'instructions et types d'instruction

- 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.

15

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

16

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	Opérande3

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.

17

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)

18

Machine à 1 adresses

- Dans de 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.

Code opération	Opérande2

Exemple :

ADD A (ACC←(ACC) + A)

Ce type de machine est le plus utilisé.

19

Mode d'adressage

- La champs 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 comportent un ensemble de bits pour indiquer le mode d'adressage.

- Les modes d'adressage les plus utilités sont :
  - Immédiat
  - Direct
  - Indirect
  - Indexé
  - relatif

20

Adressage immédiat

- L'opérande existent dans le champs adresse de l'instruction

Code opération	Opérande

Exemple :
ADD 150

ADD	150

Cette commande va avoir l'effet suivant : ACC←(ACC)+ 150

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

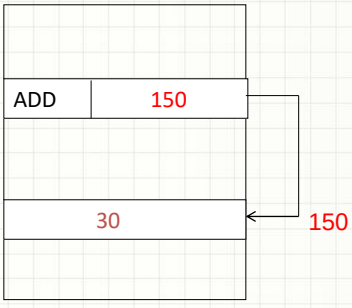
21

Adressage direct

- Le champs 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 ← (ACC)+ (ADR)

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

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



22

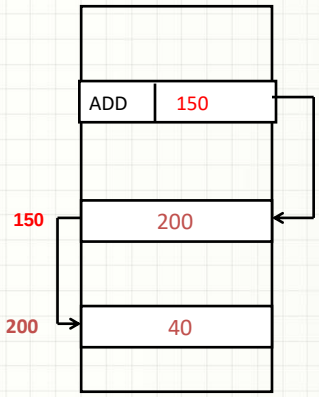
Adressage indirect

- La champs 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← (ACC)+ ((ADR))

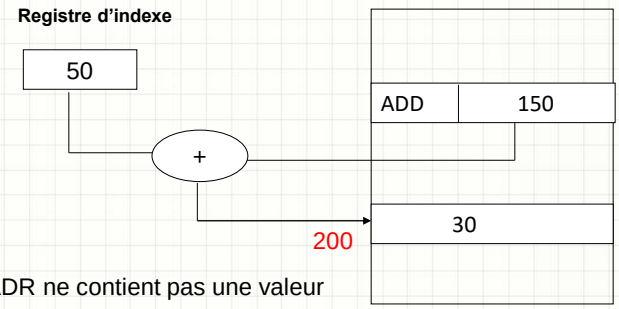
- Exemple :
- Initialement l'accumulateur contient la valeur 20
- Il faut récupérer l'adresse de l'adresse (150).
- Récupérer l'adresse de l'opérande à partir de l'adresse 150 (la valeur 200)
- Récupérer la valeur de l'opérande à partir de l'adresse 200 (**la valeur 40**)

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



Adressage indexé

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



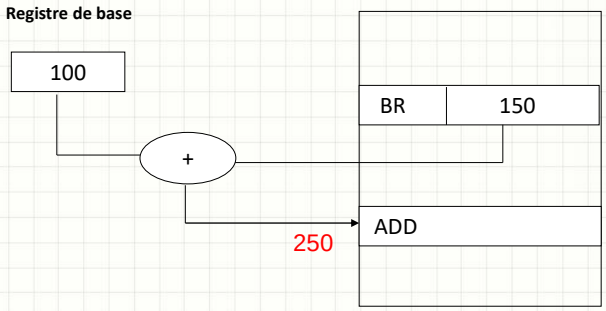
Remarque : si ADR ne contient pas une valeur immédiate alors

Adresse opérande = (ADR)+ (X)

Adressage relatif

- L'adresse effectif de l'opérande est relatif a une zone mémoire.
- L'dresse 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)



25

Cycle d'exécution d'une instruction

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

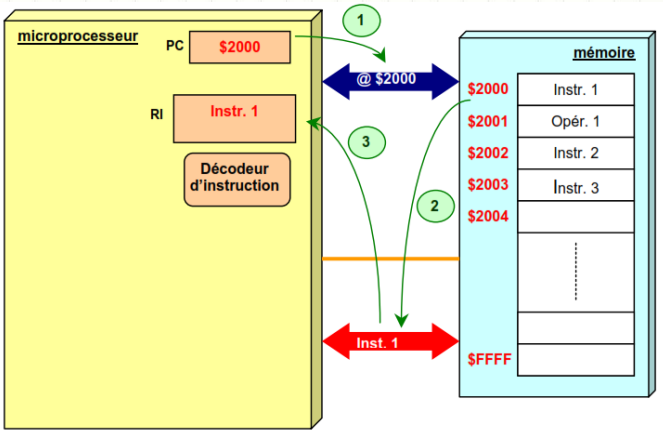
- **Phase 1:** ***Recherche de l'instruction à traiter***
- **Phase 2 :** ***Décodage de l'instruction et recherche de l'opérande***
- **Phase 3 :** ***Exécution de l'instruction***

- 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

26

Phase 1: Recherche de l'instruction à traiter

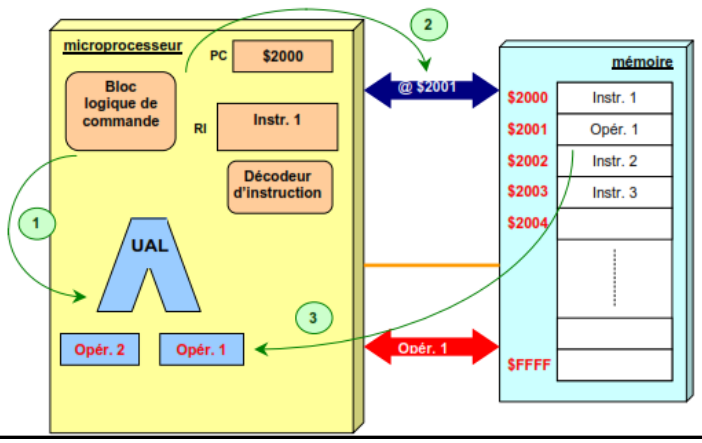
- La valeur du PC est placée sur le bus d'adresse par l'unité de commande qui émet un ordre de lecture.
- Après le temps d'accès à la mémoire, le contenu de la case mémoire sélectionnée est disponible sur le bus des données.
- L'instruction est stockée dans le registre d'instruction du processeur.



27

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

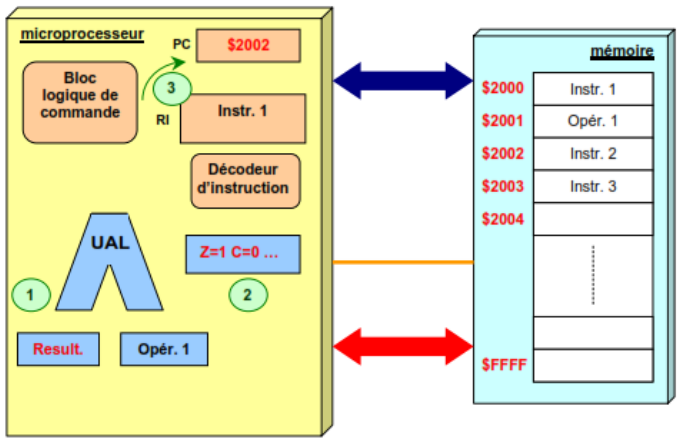
- L'unité de commande transforme l'instruction en une suite de commandes élémentaires nécessaires au traitement de l'instruction.
- Si l'instruction nécessite une donnée en provenance de la mémoire, l'unité de commande récupère sa valeur sur le bus de données.
- L'opérande est stocké dans le registre de données.



28

Phase 3 : Exécution de l'instruction

- Le séquenceur réalise l'instruction.
- Les drapeaux sont positionnés (registre d'état).
- L'unité de commande positionne le PC pour l'instruction suivante.



Cycle d'exécution d'une instruction: Exemple1

- déroulement de l'instruction d'addition en mode immédiat ACC←(ACC)+ Valeur
  - Phase 1 : (rechercher l'instruction à traiter)
    - Mettre le contenu du PC dans le registre RAM RAM← (PC)
    - Commande de lecture à partir de la mémoire
    - Transfert du contenu du RIM dans le registre RI RI← (RIM)
    - Analyse et décodage
  - Phase 2 : (traitement)
    - Transfert de l 'opérande dans l'UAL UAL ← (RI).ADR
    - Commande de l'exécution de l'opération (addition)
  - Phase 3 : (passer à l'instruction suivante)
    - PC← (PC)+ 1

Cycle d'exécution d'une instruction: Exemple 2

- Exemple 2 : déroulement de l'instruction d'addition en mode direct ACC←(ACC)+ (ADR)
  - Phase 1 : (rechercher l'instruction à traiter)
    - Mettre le contenu du PC dans le registre RAM RAM← (PC)
    - Commande de lecture à partir de la mémoire
    - Transfert du contenu du RIM dans le registre RI RI← (RIM)
    - Analyse et décodage
  - Phase 2 : (décodage et traitement)
    - Transfert de l'adresse de l 'opérande dans le RAM RAM← (RI).ADR
    - Commande de lecture
    - Transfert du contenu du RIM vers l'UAL UAL ←(RIM)
    - Commande de l'exécution de l'opération (addition)
  - Phase 3 : (passer à l'instruction suivante)
    - PC ← (PC)+ 1

31

Cycle d'exécution d'une instruction: Exemple 3

- Exemple 3 : Déroulement de l'instruction d'addition en mode indirect ACC←(ACC)+ ((ADR))
  - Phase 1 : (rechercher l'instruction à traiter)
    - Mettre le contenu du PC dans le registre RAM RAM← (PC)
    - Commande de lecture à partir de la mémoire
    - Transfert du contenu du RIM dans le registre RI RI← (RIM)
    - Analyse et décodage
  - Phase 2 : (décodage et traitement)
    - Transfert de l'adresse de l 'opérande dans le RAM ← (RI).ADR
    - Commande de lecture /* récupérer l'adresse */
    - Transfert du contenu du RIM vers le RAM RAM←(RIM)
    - Commande de lecture /* récupérer l'opérande */
    - Transfert du contenu du RIM vers l'UAL UAL ← (RIM)
    - Commande de l'exécution de l'opération (addition)
  - Phase 3 : (passer à l'instruction suivante)
    - PC ← (PC)+ 1

32

Temps d'exécution

- L'exécution d'une instruction nécessite plusieurs temps de cycle, c'est ce que l'on appelle le **CPI** (Cycles per Instruction ou nombre de cycles par instruction).
- Le temps d'exécution d'un programme est alors donné par la formule suivante (si on considère que toutes les instructions ont le même CPI) :

Texec = Nins x CPI x Tcycle

avec :

Texec: temp d'exécution du programme
Nins: nombre d'instructions
CPI : nombre de cycles par instructions
Tcycle: temps de cycle

33

Améliorations de l'architecture de base

- L'ensemble des améliorations des microprocesseurs visent à diminuer le temps d'exécution du programme.
- Deux types d'améliorations sont possibles :

- la première consiste à diminuer le temps de cycle, pour cela il suffit d'augmenter la fréquence de fonctionnement du processeur.

- la seconde consiste à diminuer le nombre d'instructions ou diminuer le nombre de cycles par instruction.

34

Pipeline

Idée générale

- Lancer le traitement d'une instruction avant que la précédente ne soit terminée

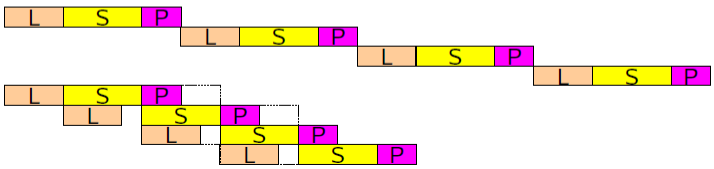
- Découpage des instructions en sous-parties élémentaires
  - En relation avec les étapes de traitement de l'instruction
  - Définition des étages du pipeline
  - «travail à la chaine»

- Exécution des sous-parties élémentaires dans les étages correspondants du pipeline.

35

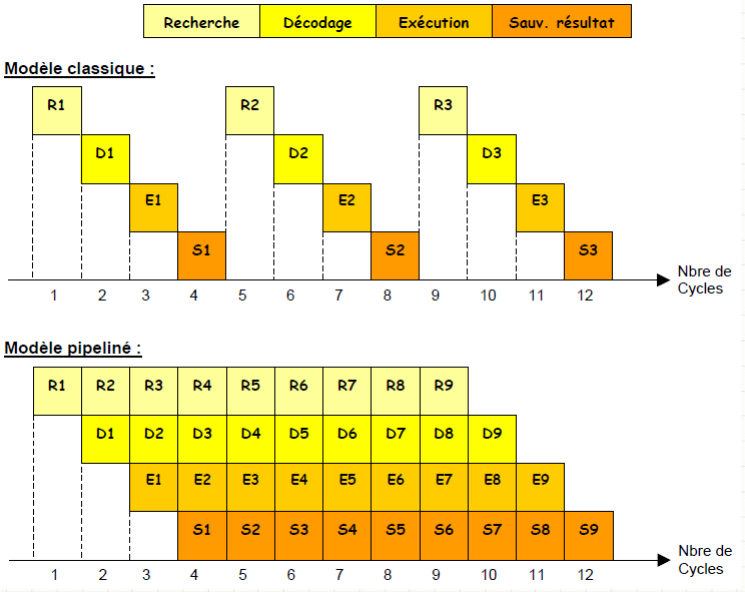
Pipeline

- Exemple : le lavomatique
  - Lavage : 30 minutes L
  - Séchage : 40 minutes S
  - Pliage : 20 minutes P



36

Pipeline



37

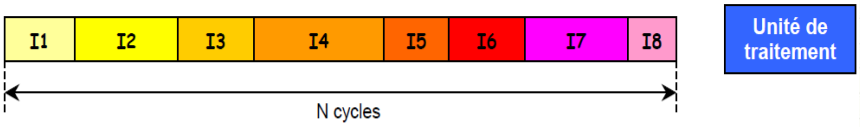
Architecture superscalaire

- Une autre façon de gagner en performance est d'exécuter plusieurs instructions en même temps.
- L'approche superscalaire consiste à doter le microprocesseur de plusieurs unités de traitement travaillant en parallèle.
- Les instructions sont alors réparties entre les différentes unités d'exécution.
- Il faut donc pouvoir soutenir un flot important d'instructions et pour cela disposer d'un cache performant.

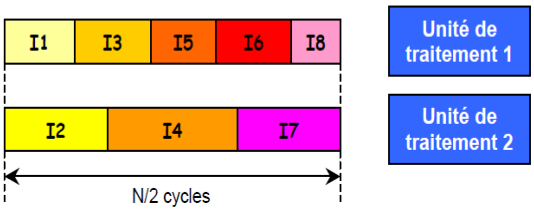
38

Architecture superscalaire

Architecture scalaire :



Architecture superscalaire :



39