Cour: Architecture des Ordinateurs (AO)

Akli ABBAS

Université Akli Mohand Oulhadj - Bouira
Département d'Informatique
2ème année - Licence informatique
2018 - 2019
Email: a_abbas@esi.dz
Disponible sur:
https://sites.google.com/a/esi.dz/a-abbas/

Contenu de la matière

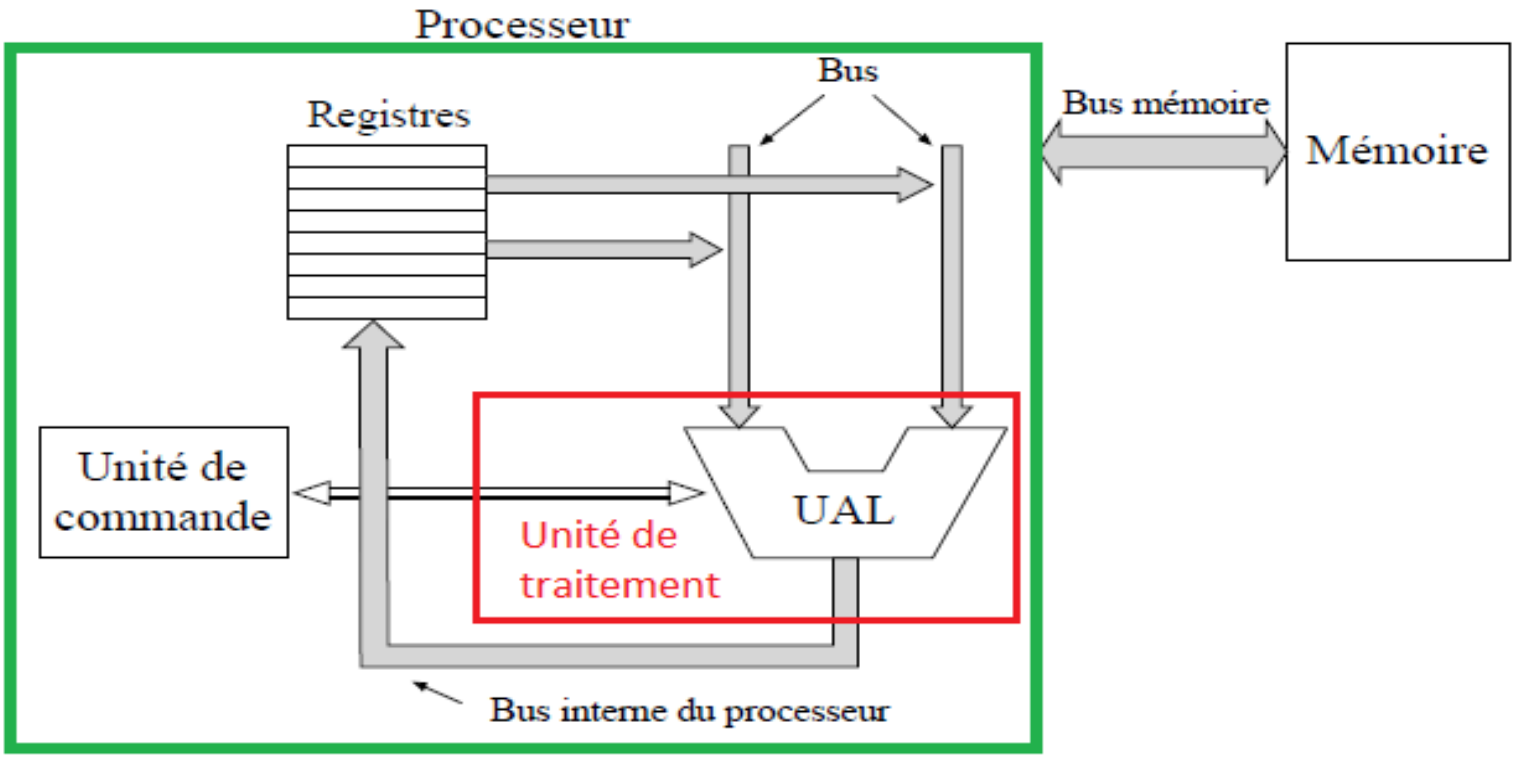
1. **Chapitre 1 :** Organisation générale de l'unité centrale d'un Ordinateur
   - **Généralités sur l'Ordinateur**
   - **Architecture de Base :**
     Le modèle de Harvard et de Von Neumann, Processeur, Mémoire et Bus
2. **Chapitre 2 :** Architecture Interne des Processeurs
   - **Introduction**
   - **Les Registres**
   - **Unité Arithmétique et Logique (UAL)**
   - **Unité de Commande (U.C)**
   - **Jeu d'instruction**
   - **Mode d'adressage**
   - **Étapes d'exécution d'un instruction**
3. **Chapitre 3 :** Étude des cas : Processeur 80x86
4. **Chapitre 4 :** Architectures des processeurs récents

Architecture interne des processeurs (Micro Architecture)

- **Introduction**
- **Les Registres**
- **Unité Arithmétique et Logique (UAL)**
- **Unité de Commande (U.C)**
- **Jeu d'instruction**
- **Mode d'adressage**
- **Étapes d'exécution d'un instruction**

Introduction

Un microprocesseur est constitué de quatre parties : **L'unité traitement** (UAL : **U**nité **A**rithmétique et **L**ogique), **l'unité commande**, **les registres** et **le bus interne**.



Les registres

- Pour assurer leur fonctionnement l'unité de traitement et l'unité de commande, utilisent des mémoires interne du processeur très rapides appelés **registre**.
- En général et pour un processeur donné, les registres sont associés à des noms court (ou mnémonique) indiquant leur rôle dans l'architecture de ce processeur.

Nom (mnémonique)	Désignation
RI	Registre Instruction
ACC	Registre Accumulateur
RAM	Registre Adresse Mémoire
PC	Compteur Programme
RE	Registre d'Etat

Les registres

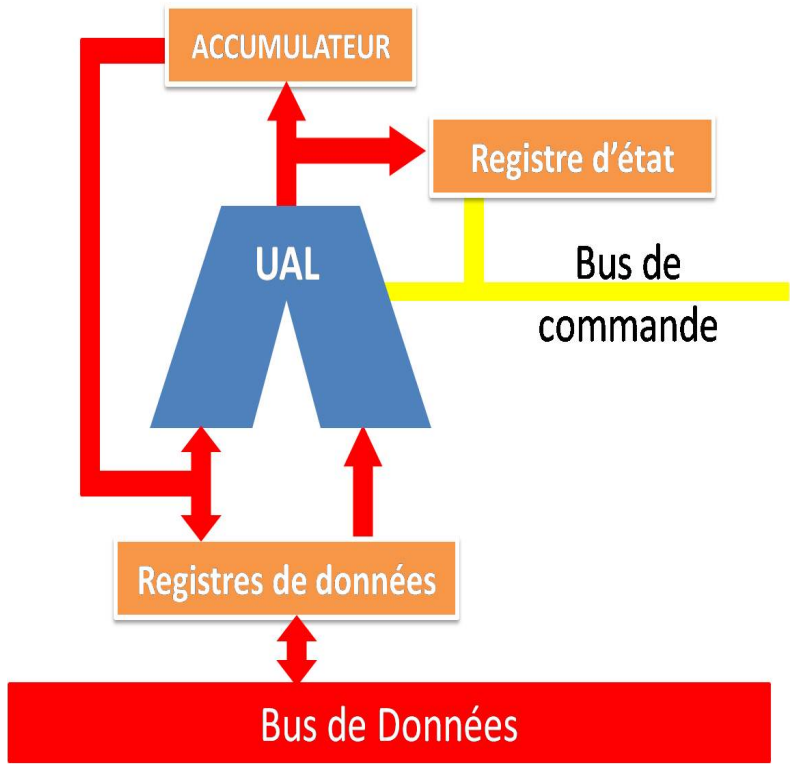
- Le processeur peut contenir d'autres registres autre que RI, ACC, RAM, PC et RE.
- Les registres sont plus rapide que la mémoire centrale, mais leur nombre est limité.
- Généralement, la taille d'un registre est égale à la taille d'un mot mémoire. La désignation d'un registre se fait en indiquant son nom (mnémonique) alors que la désignation d'une case mémoire se fait en indiquant la valeur de son adresse.

Exemple d'utilisation des registres :

1. Stockage temporaire des opérandes ramenés de la mémoire avant le lancement de l'opération.
2. Stockage temporaire du résultats d'une opération avant son transfert vers la mémoire

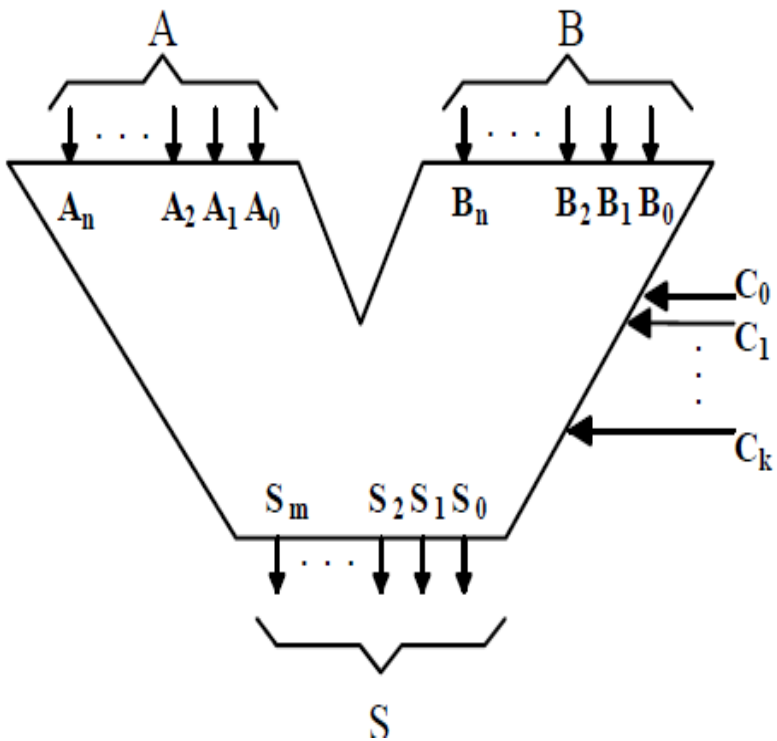
L'unité traitement
(UAL : **U**nité **A**rithmétique et **L**ogique)

- L'UAL réalise les opérations élémentaires (addition, ,soustraction,. . .).
- L'UAL regroupe les circuits qui assurent les fonctions logiques et arithmétiques de bases (ET,OU,ADD,SUS,. . .).



UAL

Reçoit deux opérandes A ($A_n...A_1A_0$) et B ($B_n...B_1B_0$) et produit le résultat S ($S_m...S_1S_0$) selon l'indication appliquée sur l'entrée C ($C_k...C_1C_0$).



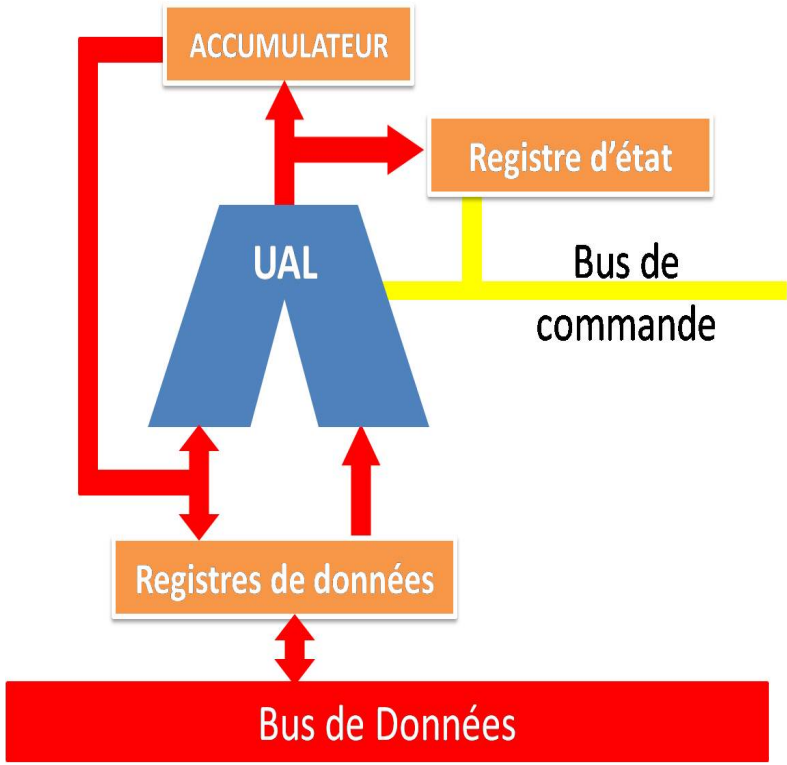
Architecture interne des processeurs

L'UAL utilise :

- **L'accumulateur** (ACC) : c'est un registre dédié à contenir le résultat d'une opération réalisée par l'UAL.
- **Les registres de données** : ces registres présentent à l'UAL les opérandes d'une opération et récupèrent le résultats se trouvant dans l'accumulateur.

Remarque : Le nombre de registres de données dépend du processeur ;

- **Un registre d'état** : Ce registre nous indique l'état du déroulement de l'opération .



UAL - Le **R**egistre d'**É**tat (RE)

- Ce registre est composé d'un ensemble de **bits** (8 bits en général). Ces bits sont appelés **indicateurs** (drapeaux ou flags).
- Les bit sont considérés individuellement.
- 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.
- Le résultat du test de leur état conditionne le déroulement de la suite d'un programme

UC : Unité de commande (ou contrôle)

Le rôle de l'unité de commande (ou unité de contrôle) est de :

- **coordonner** le travail de toutes les autres unités (UAL , mémoire,. . . .)
- et d'assurer la **synchronisation** de l'ensemble.

Elle assure :

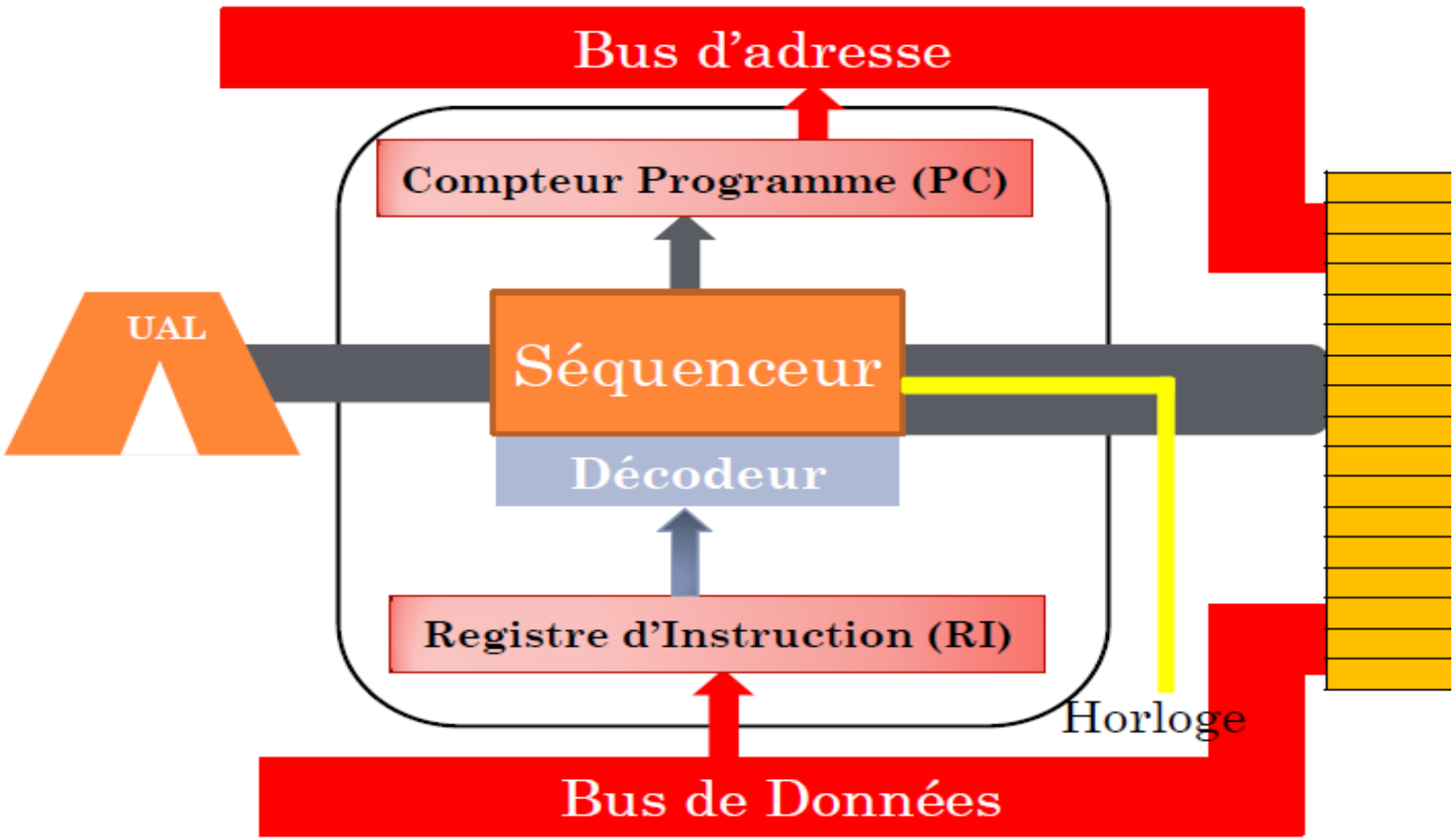
1. la **recherche** (lecture) de l'instruction et des données à partir de la mémoire,
2. le **décodage** de l'instruction et l'exécution de l'instruction en cours
3. et **prépare** l'instruction suivante.

UC : Unité de commande (ou contrôle)

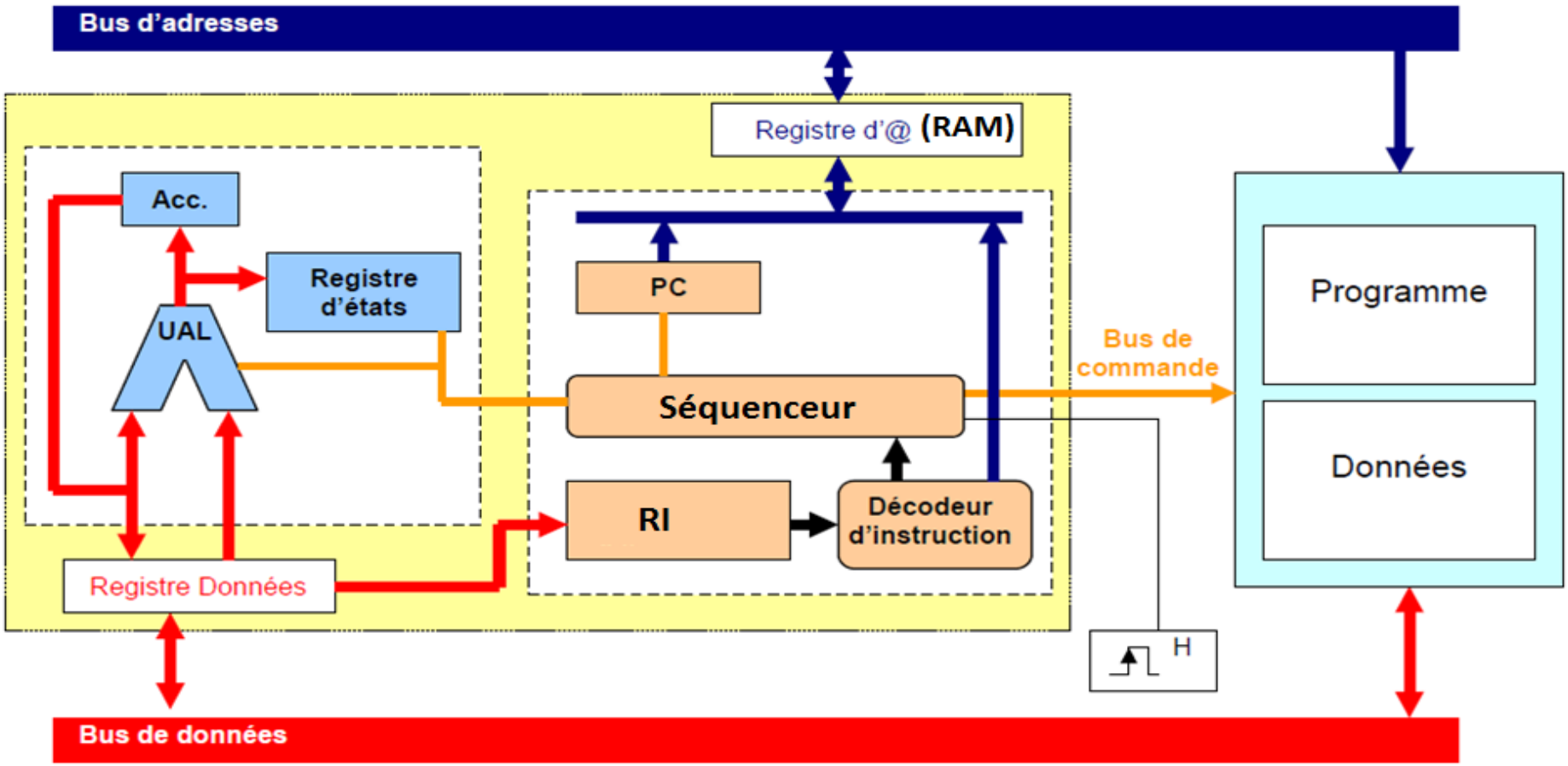
L'unité de contrôle comporte :

1. **Un registre instruction (RI)** : contient l'instruction en cours d'exécution. Chaque instruction est décoder selon sont code opération grâce à un **décodeur**.
2. 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 le première instruction du programme à exécuter.*
3. Un **séquenceur** : il organise (synchronise) l'exécution des instruction selon le rythme de l'horloge. Il élabore tous les signaux de commande des diverses parties du processeur en fonction du code de l'instruction

Schéma d'une UC



Exemple de micro architecture



Le jeu d'instructions

- Chaque processeur 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 processeur peut exécuter.

Les instructions qui constituent un programme peuvent être classifiées en 4 catégories :

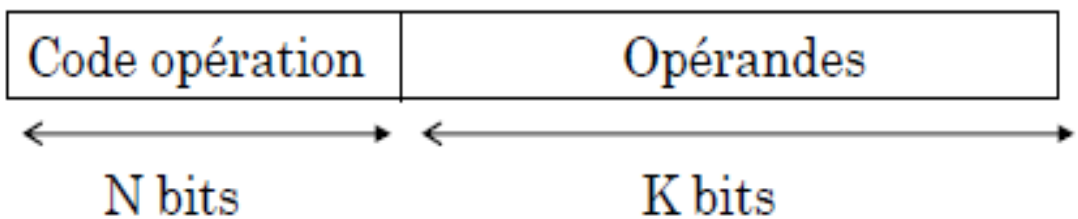
– **Les Instructions d'affectations** : permet de faire le transfert des données entre les registres et la mémoire

- Écriture : registre $\mapsto$ mémoire
- Lecture : mémoire $\mapsto$ registre

– **Les instructions arithmétiques et logiques** : ET , OU , ADD,...
– **Les Instructions de branchement** (conditionnelle et inconditionnelle)
– **Les Instructions d'entrées sorties**.

Format 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 composée de deux champs :
  1. **Code d'opération** (code instruction) : représentant l'action que le processeur doit accomplir.
  2. **Champ des opérandes** : définissant les paramètres de l'action. Un opérande peut s'agir d'une donnée ou bien d'une adresse (référence) à la donnée.



Format d'une instruction

• La taille d'une instruction dépend du **type de l'instruction** et du **type de l'opérande**.

• Les instructions et leurs opérandes sont stockés dans la mémoire.

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

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

Exemple du format d'une instruction

• **Instruction à trois opérandes** : Il faut préciser le premier opérande, le deuxième opérande et l'emplacement du résultat

Code opération	Opérande1	Opérande2	Résultat

Exemple : ADD A,B, C

ADD	A	B	C

$$C \longleftarrow A + B$$

La taille de l'instruction est grande $\Rightarrow$ Pratiquement il n'existe pas d'instruction de ce type.

Exemple du format d'une instruction

• **Instruction à deux opérandes** : Il faut préciser le premier opérande et le deuxième opérande. Le résultat est implicitement mis dans le deuxième opérande .

Code opération	Opérande1	Opérande2

Exemple : ADD A,B

ADD	A	B

$$B \longleftarrow A + B$$

Exemple du format d'une instruction

• **Instruction à un opérande** : il faut préciser uniquement le deuxième opérande. Le premier opérande existe dans le registre accumulateur. Le résultat est mis dans le registre accumulateur.

Exemple : ADD B

$$ACC \longleftarrow ACC + B$$

Ce type d'instruction est le plus utilisé.

Mode d'adressage

- 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

Adressage Immédiat

La valeur de l'opérande existe dans le champ opérande de l'instruction

Code opération	valeur

Exemple : ADD 150

ADD	150

Cette commande va avoir l'effet suivant : $ACC \longleftarrow ACC + 150$
Si le registre accumulateur contient la valeur 200 alors après l'exécution son contenu sera égale à 350.

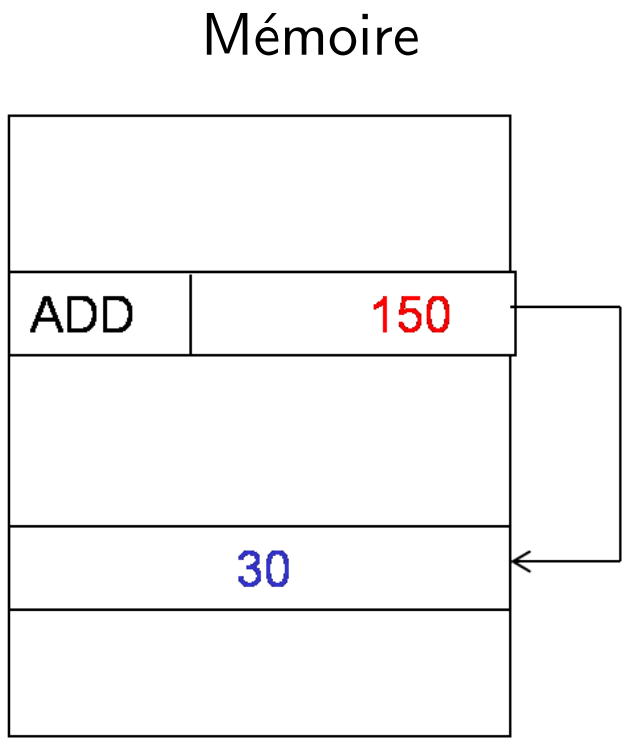
Adressage Direct

Le champs opérande contient l'adresse de l'opérande (emplacement en mémoire).

Code opération	adresse de l'opérande

Pour réaliser l'opération, il faut récupérer (lire) l'opérande à partir de la mémoire.

Exemple :
ADD 150
Cette commande va avoir l'effet suivant :
$ACC \longleftarrow ACC + (150)$
$ACC \longleftarrow ACC + 30$
Si le registre accumulateur contient la valeur 200 alors après l'exécution son contenu sera égale à 230



Adressage Indirect

Le champs opérande contient l'adresse de l'adresse de l'opérande.

Code opération	adresse d'adresse de l'opérande

Pour réaliser l'opération, il faut d'abord récupérer l'adresse de l'opérande à partir de la mémoire, ensuite, chercher l'opérande à partir de la mémoire.

Exemple :

$ADD\ 150 \Leftrightarrow ACC \longleftarrow (ACC) + ((ADR))$

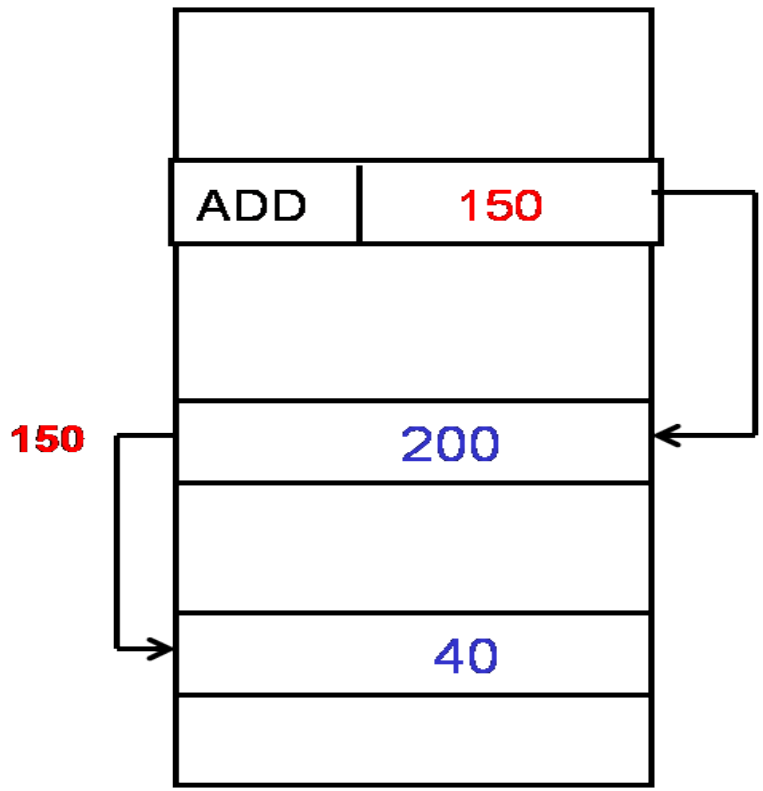
Cette commande va avoir l'effet suivant :

$ACC \longleftarrow ACC + ((150))$

$ACC \longleftarrow ACC + (200)$

$ACC \longleftarrow ACC + 40$

Si le registre accumulateur contient la valeur 200 alors après l'exécution son contenu sera égale à 240

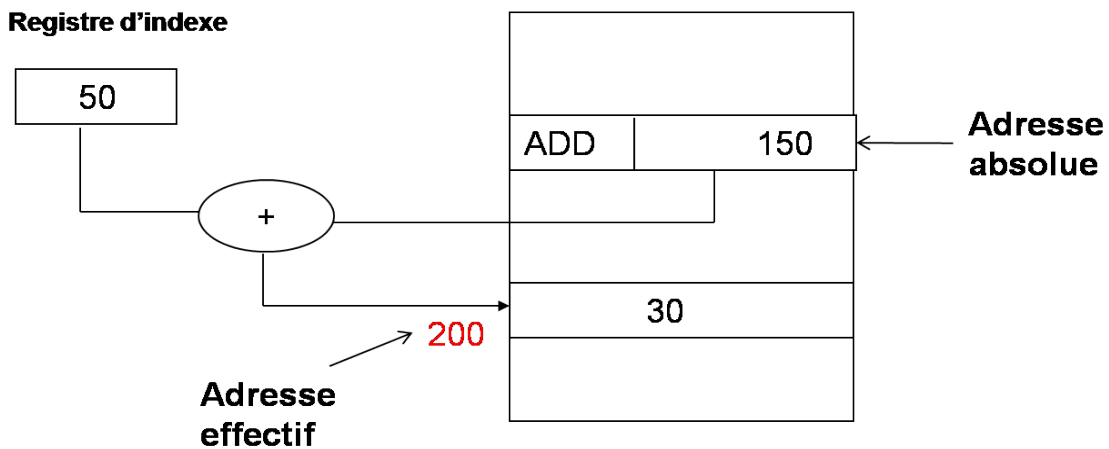


Adressage Indexé

Le champs opérande contient l'adresse **absolue** de l'opérande. L'adresse **effectif** de l'opérande est relatif à une zone mémoire. L'adresse de cette zone se trouve dans un registre spécial appelé **registre index**.

Code opération	adresse absolue de l'opérande

Adresse effectif de l'opérande = ADR + R_Index



Remarque :

si ADR ne contient pas une valeur immédiate alors

Adresse opérande = (ADR)+ (X)

Adressage relatif

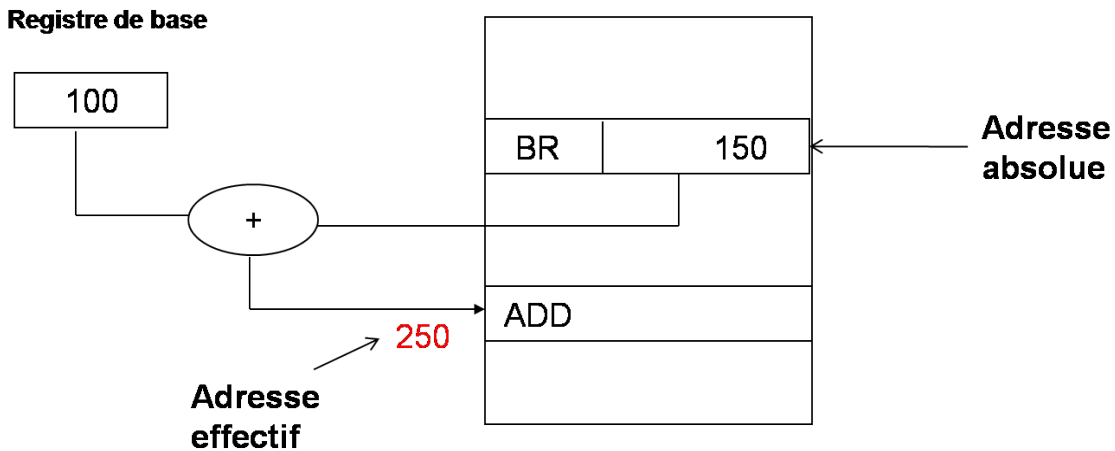
Le champs opérande contient l'adresse absolue de l'opérande. L'adresse effectif de l'opérande est relatif à une zone mémoire. L'adresse de cette zone se trouve dans un registre spécial appelé registre de base.

Code opération	adresse absolue de l'opérande

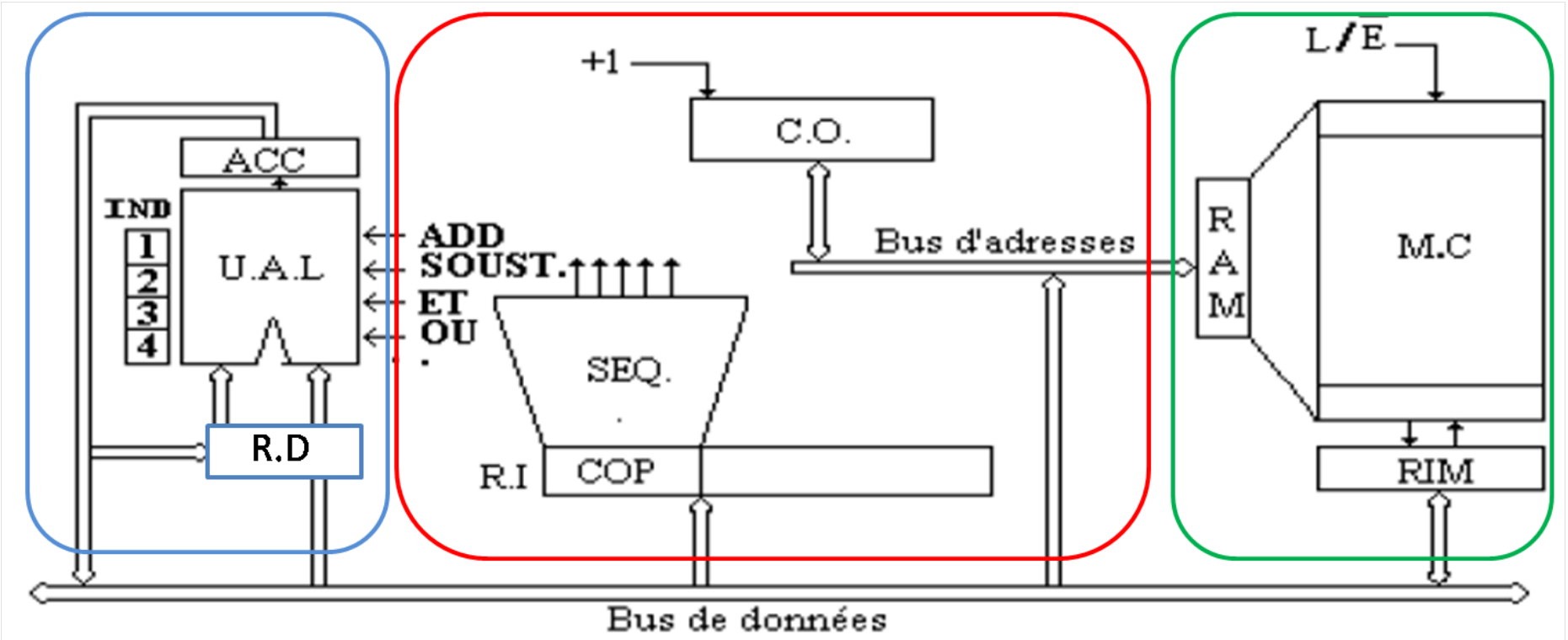
Adresse effectif de l'opérande = ADR + R_Base

Remarque :

Ce mode d'adressage est utilisée pour les instructions de branchement.



La machine pédagogique MIASM



Calcul | Contrôle | Mémoire

Étapes d'exécution d'une instruction

Étape 1 : **Rechercher (ou charger) l'instruction à traiter** ;

Étape 2 : **Décoder l'instruction chargée** ;

Étape 3 : **Rechercher (ou charger) l'opérande** ;

Étape 4 : **Exécuter l'instruction** ;

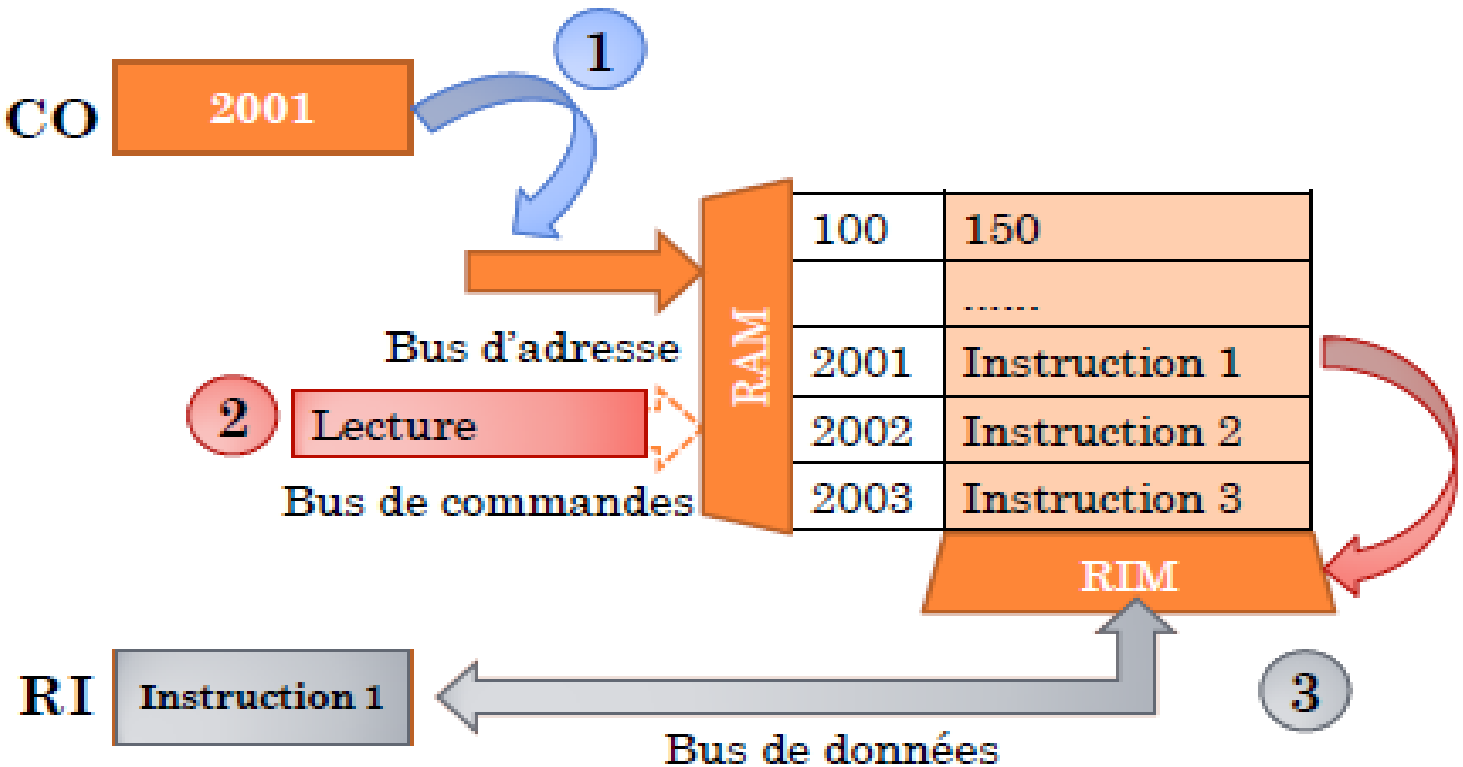
Étape 5 : **Passer à l'instruction suivante**.

Chaque étape comporte un certain nombre d'opérations élémentaires (micro-commandes) exécutées dans un ordre bien précis (elle sont générées par le séquenceur).

Étapes d'exécution d'une instruction

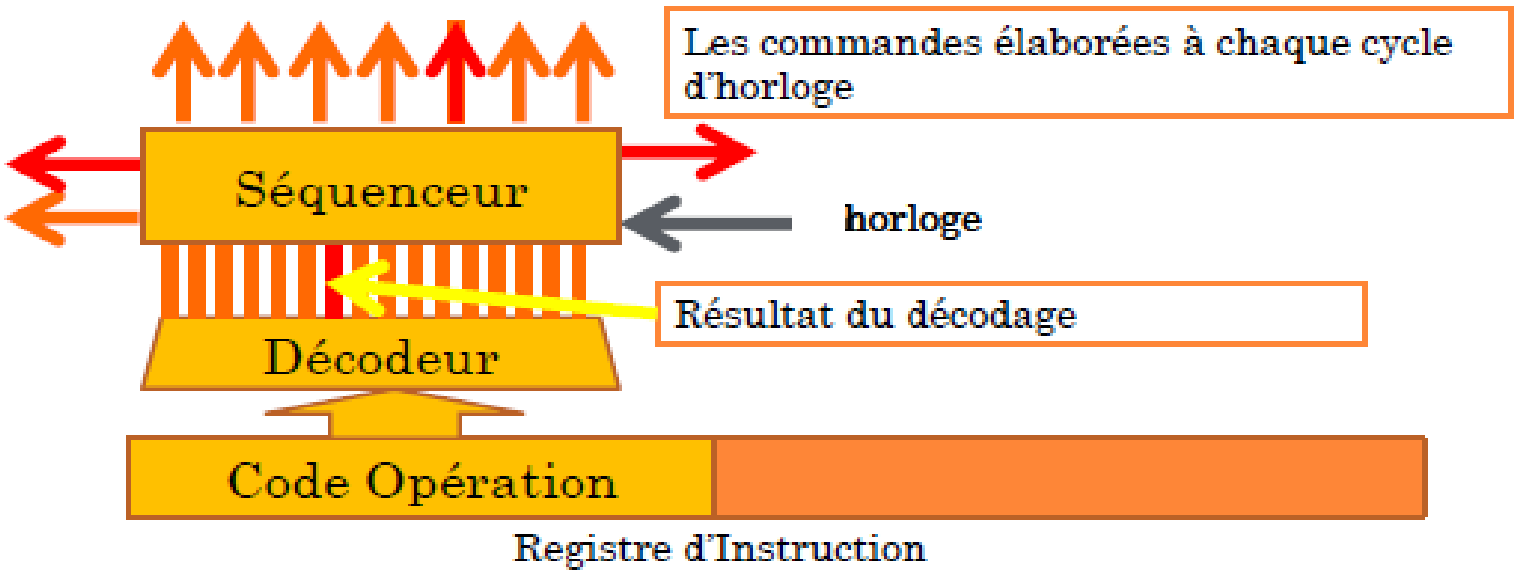
Étape 1 :Rechercher (ou charger) l'instruction à traiter :

1. Mettre le contenu du Compteur Ordinal (CO) dans le Registre d'Adresse Mémoire (RAM) : $RAM \longleftarrow CO$;
2. Commande de lecture à partir de mémoire ;
3. Transfert du contenu du Registre d'Instruction Mémoire (RIM) dans le Registre d'Instruction (RI) : $RI \longleftarrow RIM$



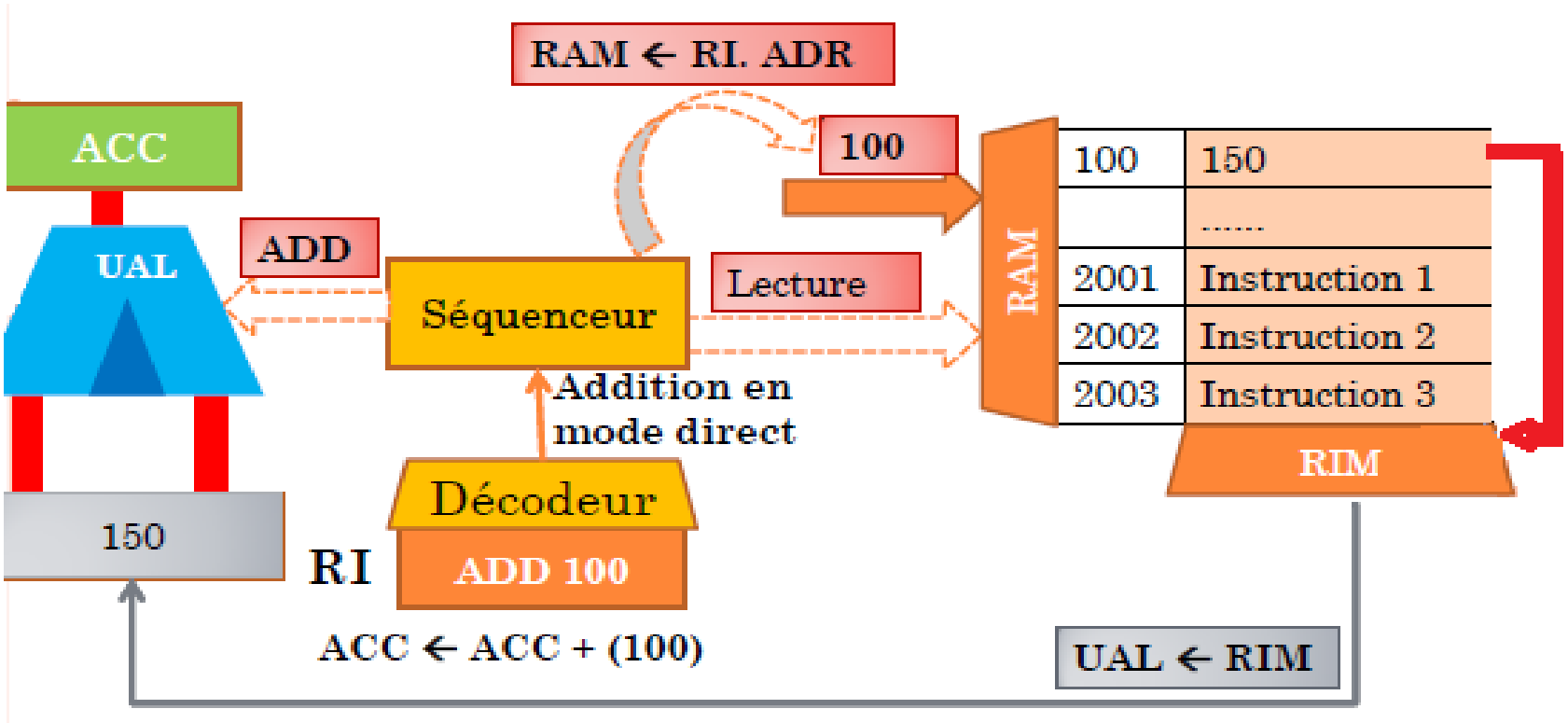
Étapes d'exécution d'une instruction

Étape 2 :Décoder l'instruction chargée : Le code d'opération indique la nature de l'opération à effectuer (addition, soustraction,...) et le nombre de mots de l'instruction ;
A la base du code de l'instruction le séquenceur élabore une suite de commandes élémentaires.



Étapes d'exécution d'une instruction

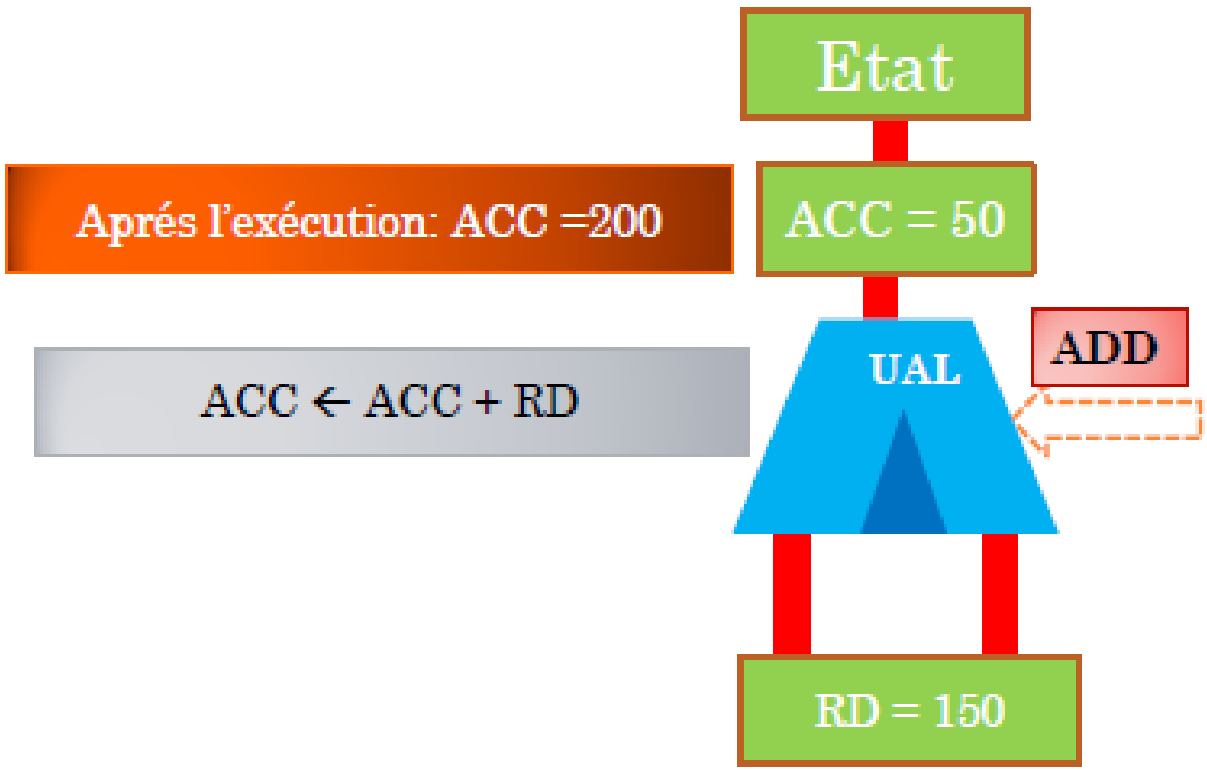
Étape 3 : Rechercher (ou charger) l'opérande : Si l'instruction nécessite une donnée qui se trouve en mémoire, le séquenceur émet les commandes pour récupérer cette donnée. *RAM* ⟵ *RI.ADR* et *UAL* ⟵ *RIM*



Étapes d'exécution d'une instruction

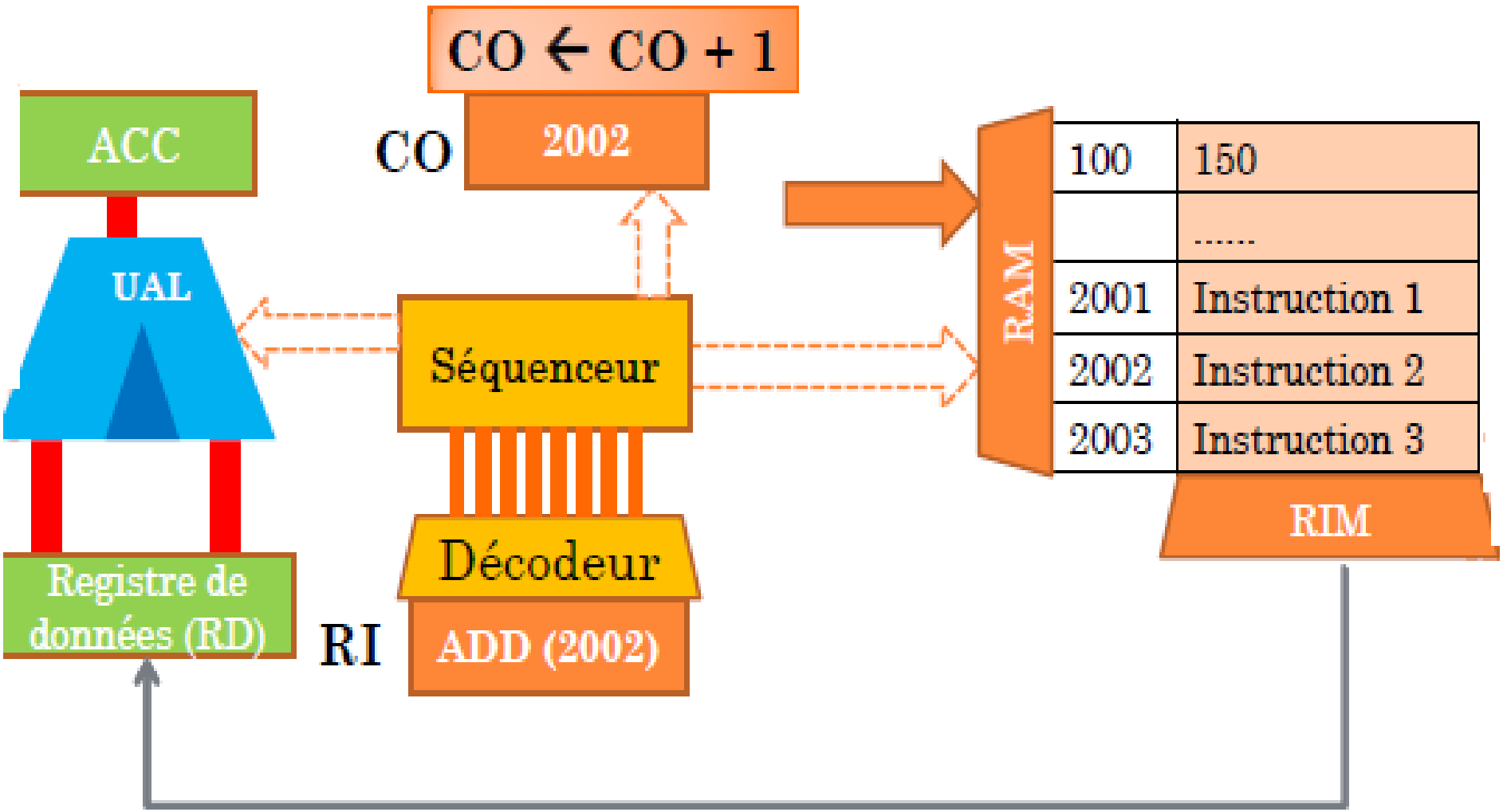
Étape 4 :Exécuter l'instruction :

Les bits d'état sont positionnés S = 0, Z = 0, ...



Étapes d'exécution d'une instruction

Étape 5 : Passer à l'instruction suivante : Le Compteur Ordinal (CO) est mis à jour avec l'adresse de l'instruction suivante.



Étapes d'exécution d'une instruction

Exemple 1 : déroulement de l'instruction d'addition en mode immédiat :
ADD valeur $\Leftrightarrow$ ($ACC \longleftarrow (ACC) + Valeur$)
Étapes 1 : Charger l'instruction

- $RAM \longleftarrow CO$
- Lecture
- $RI \longleftarrow RIM$

Étapes 2 : Décoder l'instruction
Étapes 4 : Exécuter l'instruction

- $RD \longleftarrow valeur$
- $ACC \longleftarrow ACC + RD$

Étapes 5 : Passer à l'instruction suivante

- $CO \longleftarrow CO + 1$

Étapes d'exécution d'une instruction

Exemple 2 : déroulement de l'instruction d'addition en mode direct :

ADD Adr $\Leftrightarrow (ACC \longleftarrow (ACC) + (Adr))$

Étapes 1 : Charger l'instruction

- $RAM \longleftarrow CO$
- Lecture
- $RI \longleftarrow RIM$

Étapes 2 : Décoder l'instruction

Étapes 3 : Charger l'opérande

- $RAM \longleftarrow Adr$
- Lecture
- $RD \longleftarrow RIM$

Étapes 4 : Exécuter l'instruction

- $ACC \longleftarrow ACC + RD$

Étapes 5 : Passer à l'instruction suivante

- $CO \longleftarrow CO + 1$

Étapes d'exécution d'une instruction

Exemple 3 : déroulement de l'instruction d'addition en mode indirect :
ADD Adr $\Leftrightarrow (ACC \longleftarrow (ACC) + ((Adr)))$

Étapes 1 : Charger l'instruction

- $RAM \longleftarrow CO$
- Lecture
- $RI \longleftarrow RIM$

Étapes 2 : Décoder l'instruction

Étapes 3 : Charger l'opérande

- $RAM \longleftarrow Adr$
- Lecture
- $RAM \longleftarrow RIM$
- Lecture
- $RD \longleftarrow RIM$

Étapes 4 : Exécuter l'instruction

- $ACC \longleftarrow ACC + RD$

Étapes 5 : Passer à l'instruction suivante

- $CO \longleftarrow CO + 1$

Lien du cours

Disponible sur :

`https://sites.google.com/a/esi.dz/a-abbas`