Machine Pédagogiques

By L.ABADA

Machine Pédagogiques

Définition

- Un ordinateur est une machine électronique capable **de résoudre un problème** en traitant un certain nombre **d'instructions**, organisées en programmes.
- Dès sa mise sous tension, l'ordinateur **exécute**, l'une après l'autre **des instructions**.
- Ces instructions lui font **lire**, **manipuler**, puis **réécrire** un ensemble de données.
- Les données à manipuler se trouvent généralement dans **la mémoire**, elles sont obtenues par **la lecture** de composants **d'interface** (périphériques d'entrée)
- Une fois ces données **utilisées**, ou **manipulées**, les informations obtenues sont **transférées** vers des **périphériques de sortie**.

Machine Pédagogiques

Hiérarchie des langages

- La résolution d'un problème donné revient à écrire une suite d'instructions.
- Cette suite d'instructions est appelée programme.
- Pour être exécutable un programme doit être converti en Langage Machine.
- La Compilation est le fait de traduire un code source, écrit dans un langage de haut niveau facilement compréhensible par l'humain, vers un langage de plus bas niveau, l'assembleur ou le langage machine.

Langage évolué → Assembleur → Langage Machine

Machine Pédagogiques

Hiérarchie des langages

- La différence entre **l'Assembleur** et **le Langage Machine** est dans le fait que l'assembleur est **intelligible par l'humain** alors que **le langage machine** est purement **binaire** (***c'est le seul langage compréhensible par la machine***).
- Cependant il existe une correspondance directe entre ces 2 langages :
    - à chaque opération de l'Assembleur correspond un code binaire.
    - les données et les adresses (qui sont des valeurs numériques) sont automatiquement converties en binaire.

Machine Pédagogiques

Description d'un ordinateur

- La première description d'un ordinateur avec mémoire a été élaborée par le mathématicien **John Von Newman** au début du vingtième siècle.
- L'architecture de Von Newman est composée **d'une Mémoire Centrale** et **d'un Processeur** composé lui-même **d'une Unité de Contrôle** et **d'une Unité Arithmétique et Logique (UAL)**
- Toutes les architectures qui ont succédé à l'architecture décrite par Von Newman sont globalement basées sur cette dernière.

Ordinateur = Unité centrale de traitement (CPU) + Mémoire Centrale Unité de contrôle + UAL

Machine Pédagogiques

La mémoire centrale

- Elle est composée de cellules ou cases capables d'enregistrer une information binaire.
- Cet ensemble de cases est regroupé en mots mémoires. Chaque mot peut être contenu dans 8, 16, 32, … cases.
- Chaque mot est référencé par un numéro appelé « adresse ».
- La mémoire centrale peut contenir deux types d'informations :

  programmes et **données**.

Machine Pédagogiques

Unité Arithmétique et Logique

- Elle est composée de circuits dont le but est d'effectuer des opérations **arithmétiques** et **logiques** sur les **opérandes** (données) tels que **l'addition**, la **soustraction**, le **ET**, le **OU**…etc
- L'**UAL** comporte également deux registres :
  - l'**Accumulateur** : registre de travail qui sert à stocker un opérande en début et en fin d'opération.
  - Le **registre d'Indicateurs** : met à jour à la fin de chaque opération les principaux indicateurs tels que le débordement (overflow), la retenue (carry) ou encore le signe du résultat.
- Généralement tous les circuits de cette unité ont **2 entrées** et **une sortie**.
- Les **2 entrées** du circuit **reçoivent les données** d'une opération arithmétique ou logique et **la sortie renvoie le résultat** après exécution de l'opération.

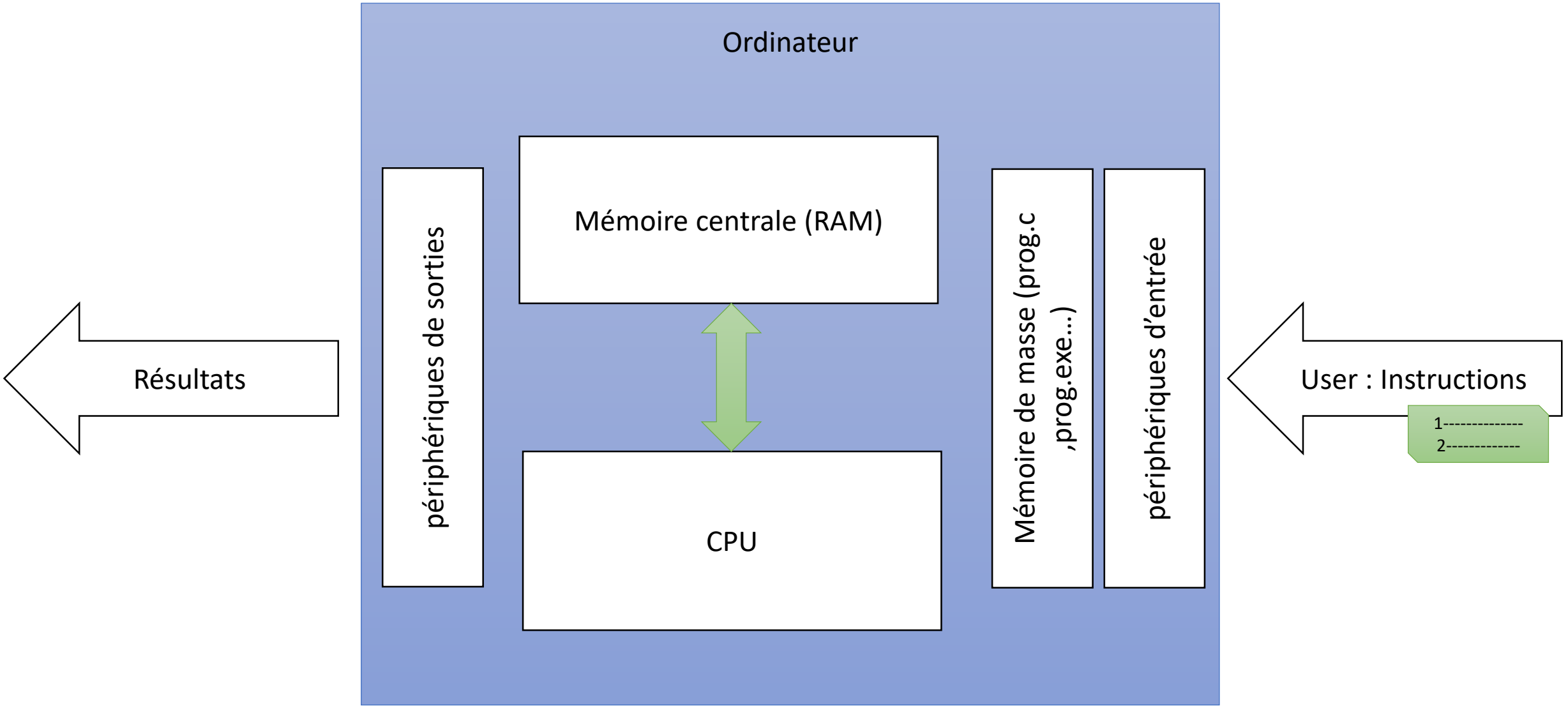
Machine Pédagogiques

Unité de contrôle :

Elle est composée de circuits logiques qui gèrent l'ensemble de l'ordinateur.

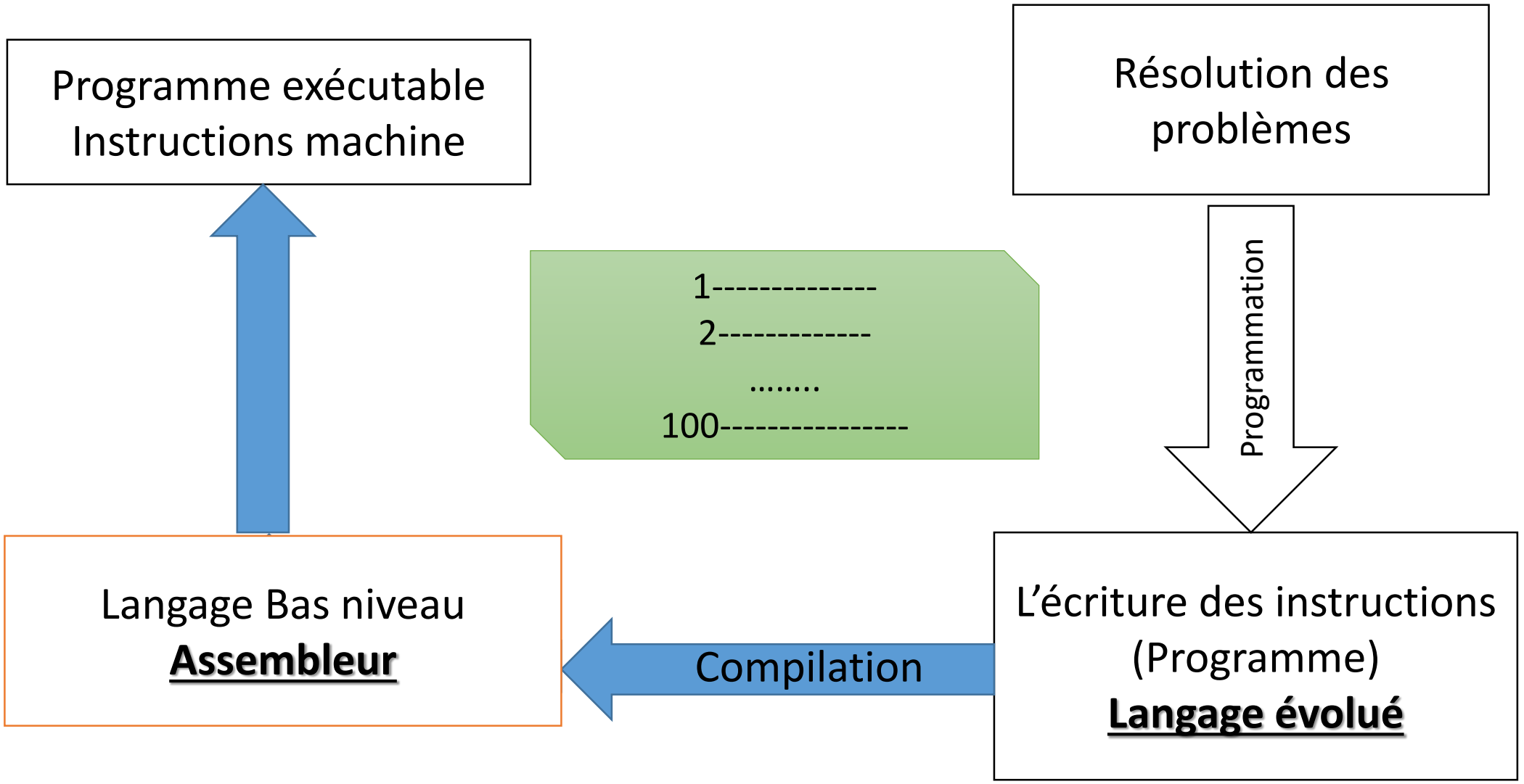
Cette unité nous permet de récupérer les instructions à partir de la mémoire centrale, de les traiter et de restituer le résultat.

Machine Pédagogiques



Machine Pédagogiques

Un langage évolué :
un langage de haut niveau facilement compréhensible par l'humain



Machine Pédagogiques

- assembleur est intelligible par l'humain -
- langage machine est purement binaire -

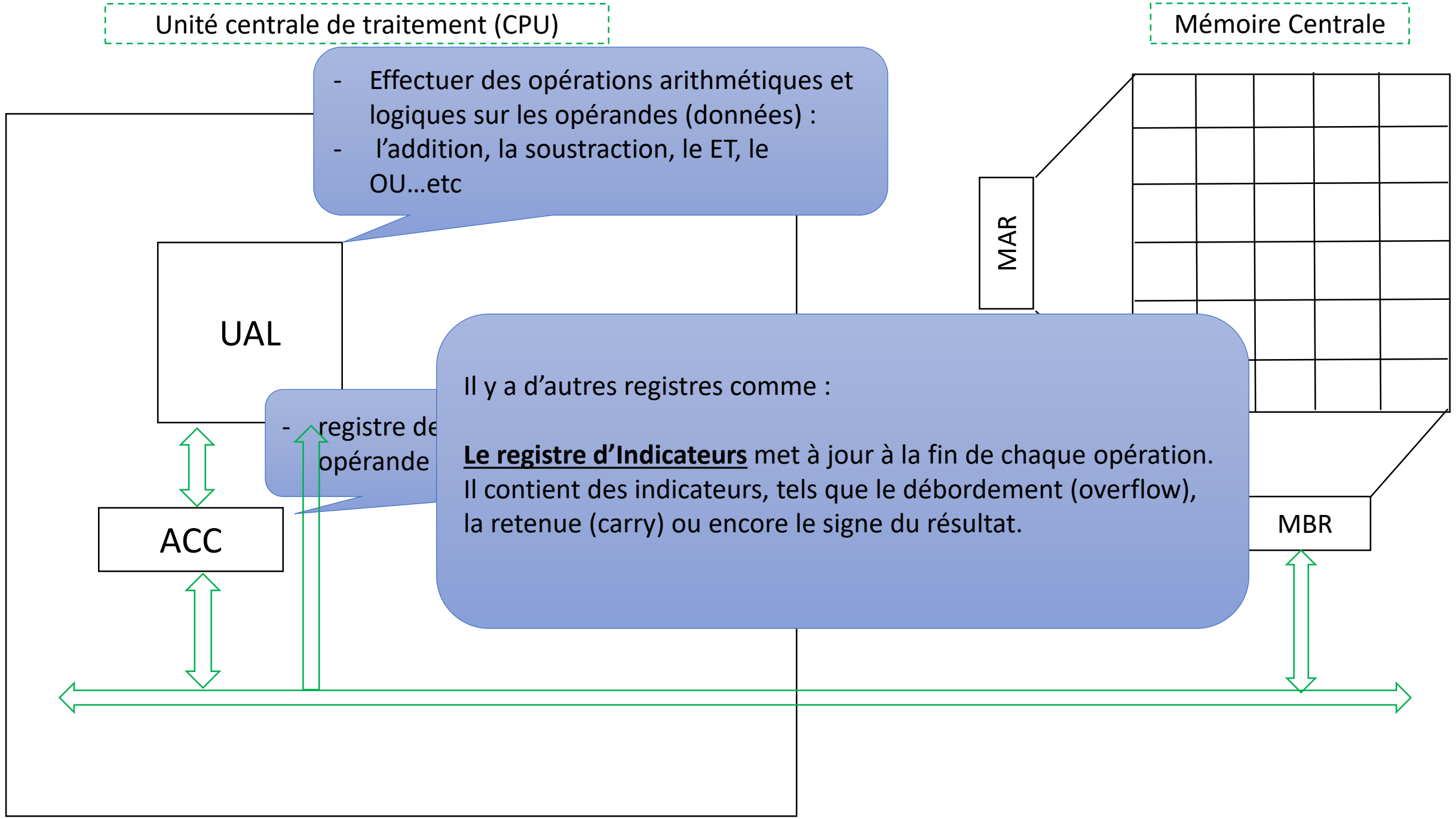
Il existe une correspondance directe entre ces 2 langages :

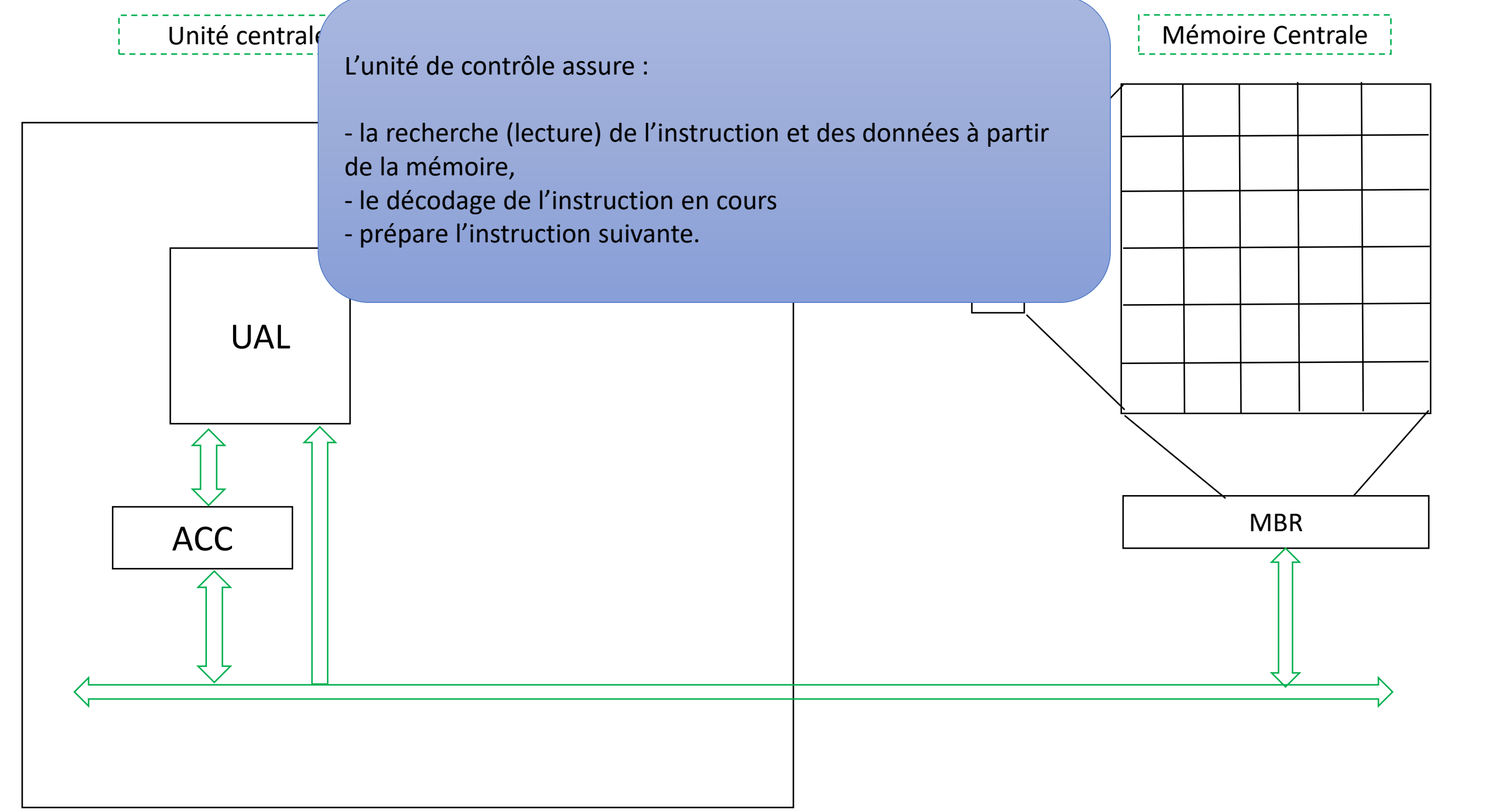
- chaque opération de l'Assembleur correspond un code binaire.
- les données et les adresses sont automatiquement converties en binaire.

Machine Pédagogiques

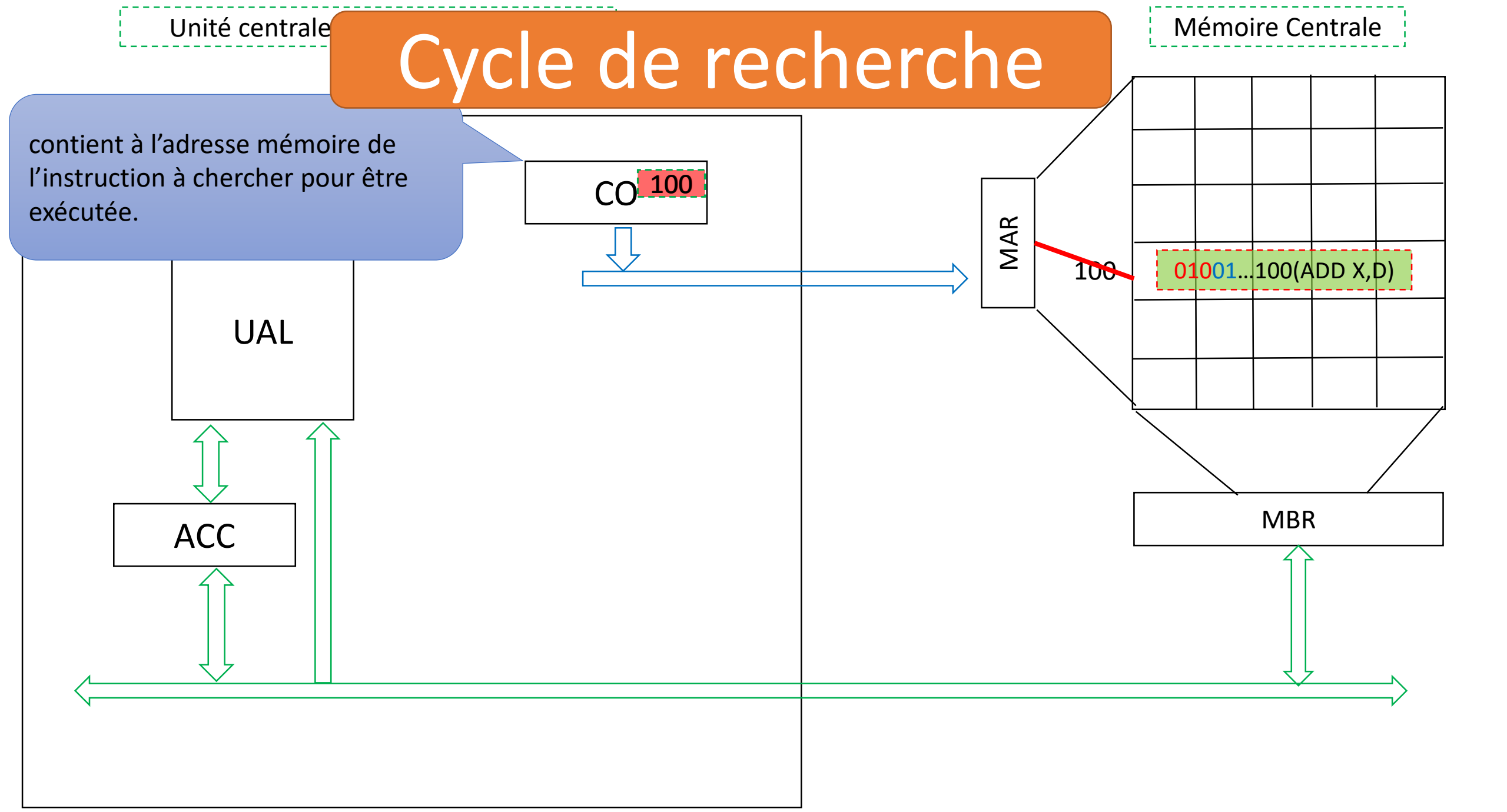
Architecture **Von Newman**

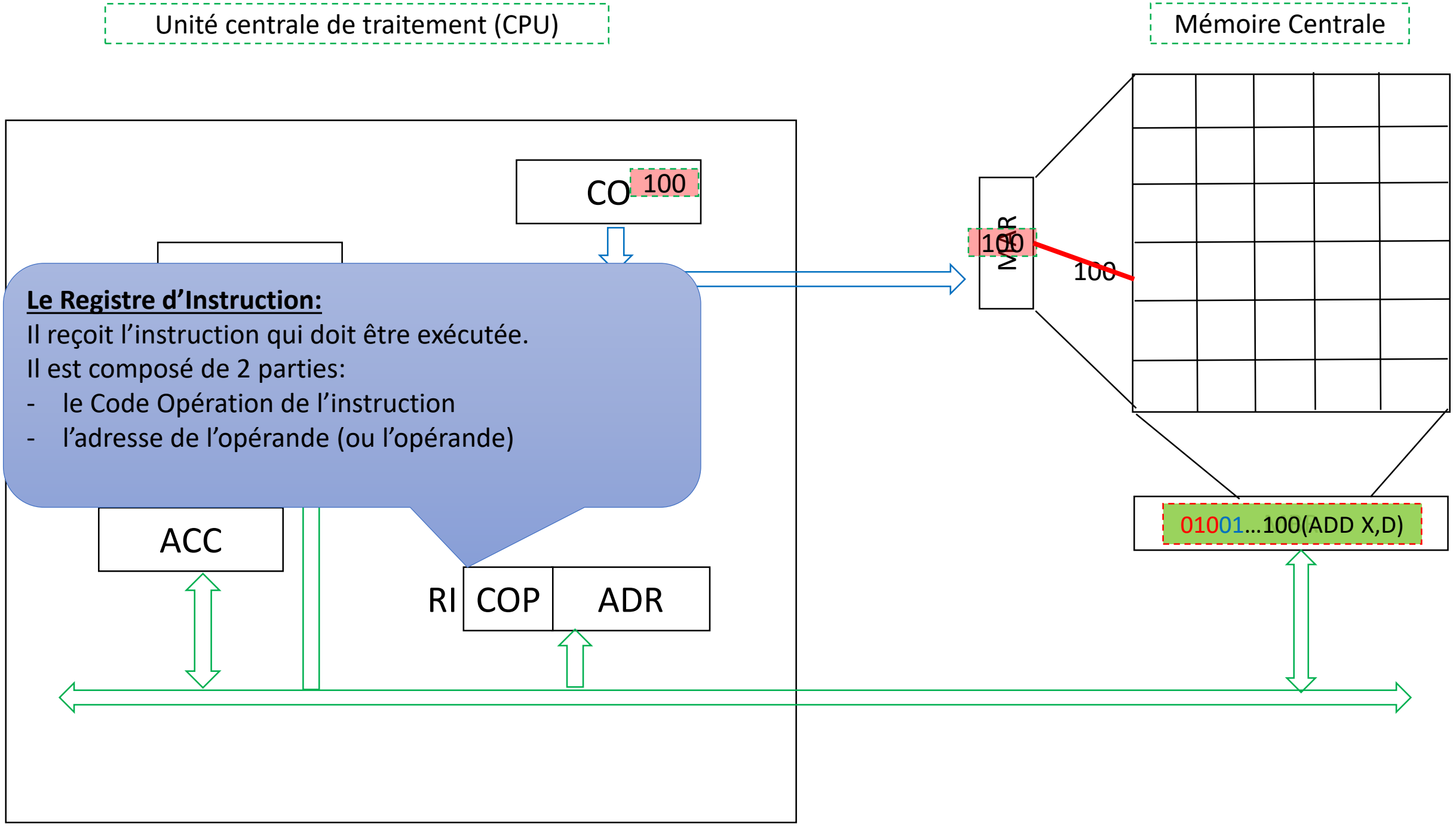
By L.ABADA


Unité centrale de traitement (CPU)
Mémoire Centrale
- Effectuer des opérations arithmétiques et logiques sur les opérandes (données) :
- l'addition, la soustraction, le ET, le OU…etc
MAR
UAL
- registre de
opérande
Il y a d'autres registres comme :
Le registre d'Indicateurs met à jour à la fin de chaque opération. Il contient des indicateurs, tels que le débordement (overflow), la retenue (carry) ou encore le signe du résultat.
ACC
MBR

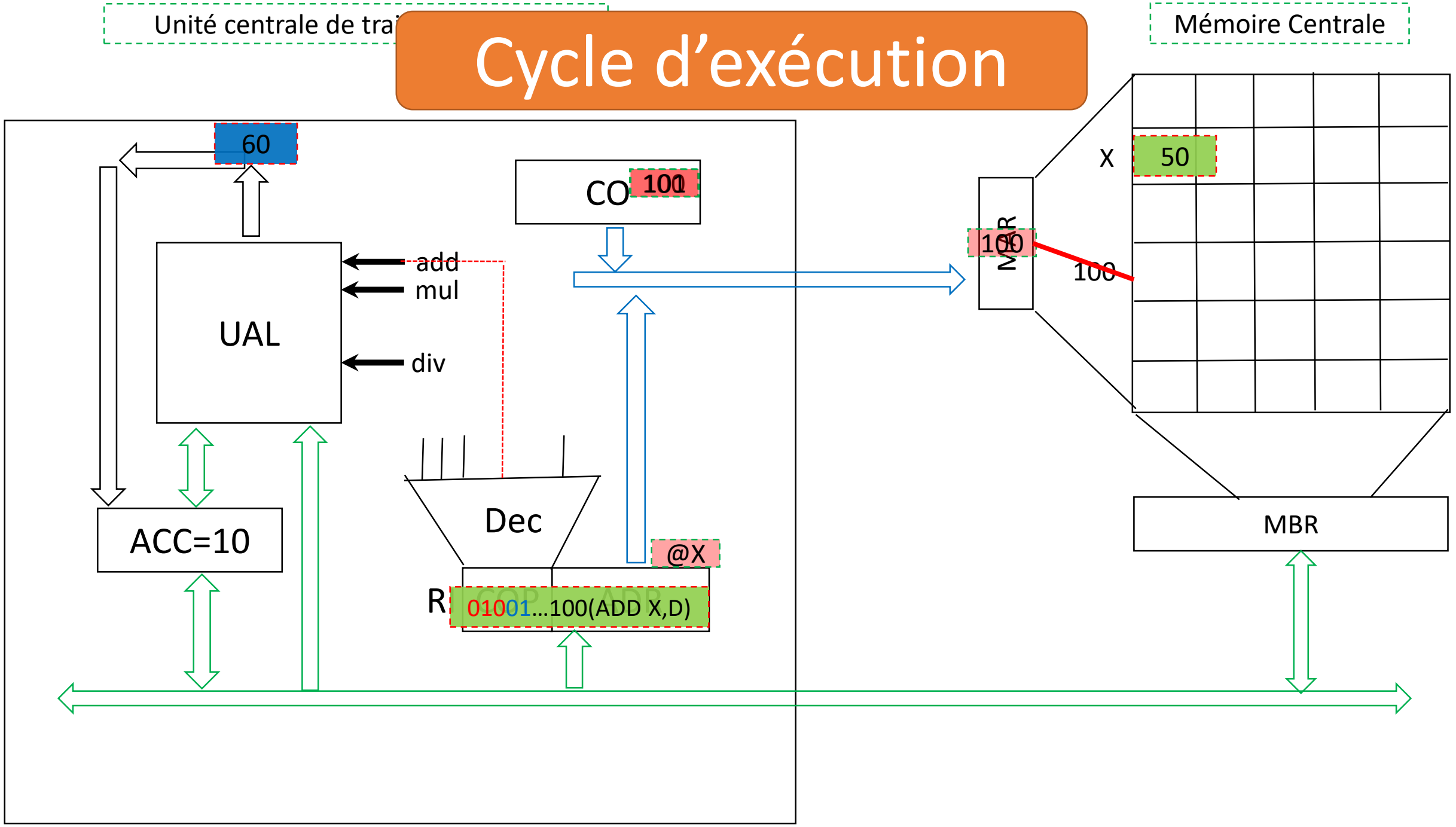

Unité centrale
Mémoire Centrale
L’unité de contrôle assure :
- la recherche (lecture) de l’instruction et des données à partir de la mémoire,
- le décodage de l’instruction en cours
- prépare l’instruction suivante.
UAL
ACC
MBR

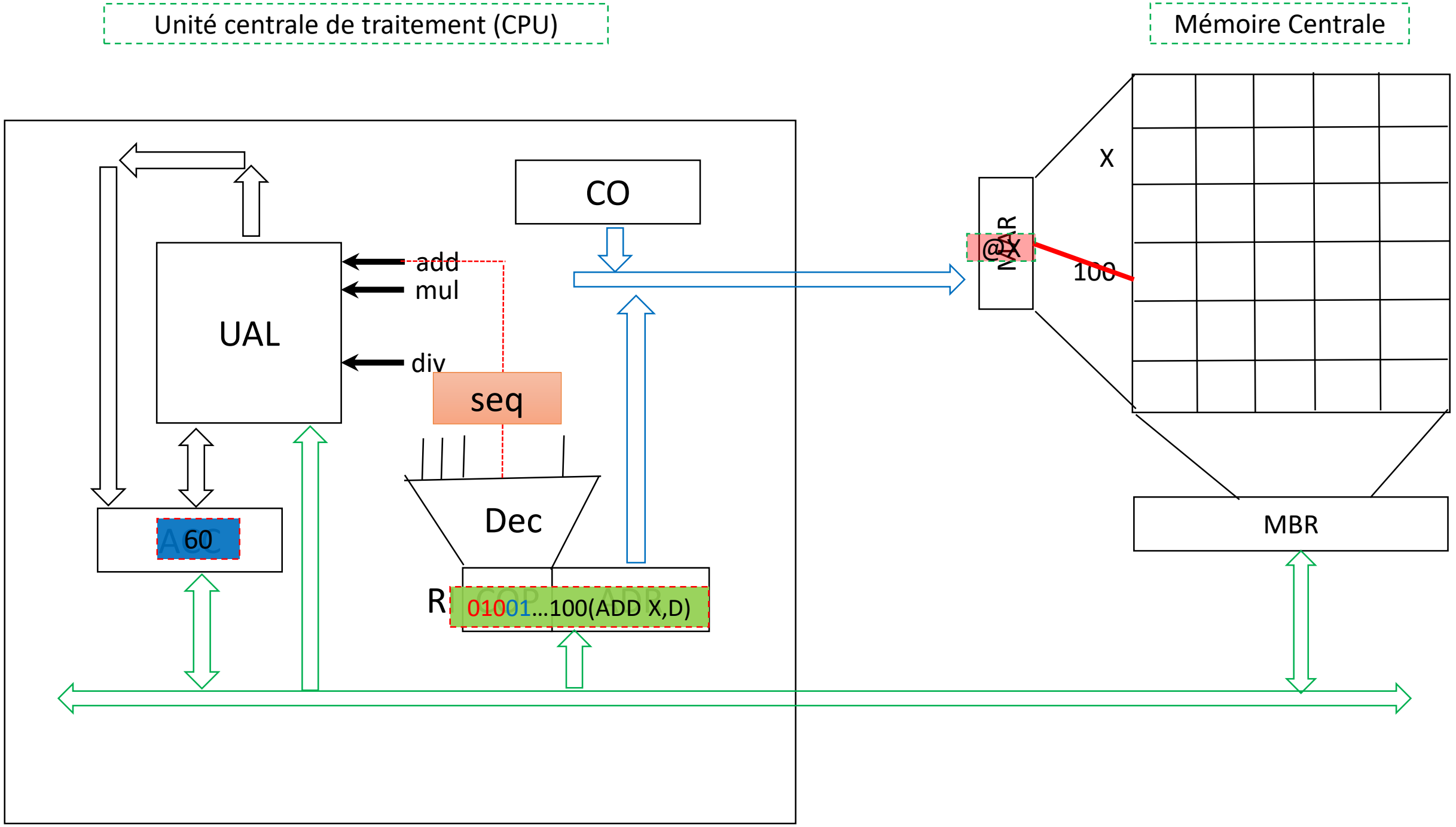
Cycle de recherche

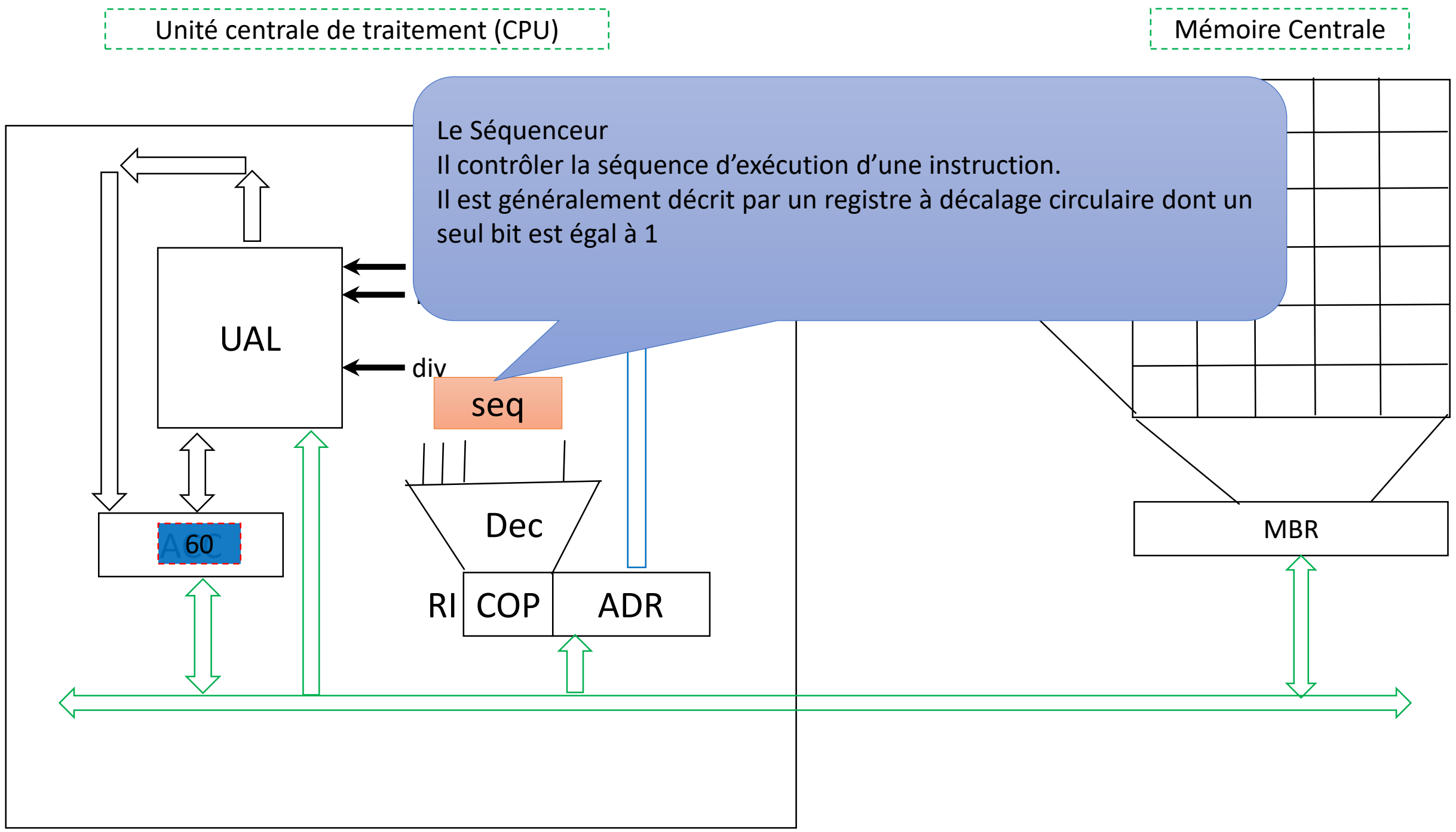


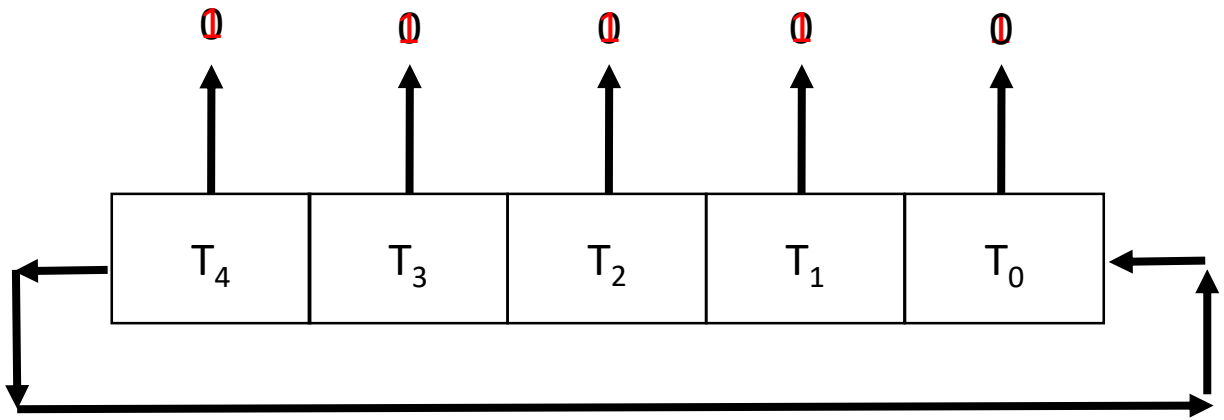

Unité centrale de traitement (CPU)
Mémoire Centrale
CO
100
MAR
100
100
01001…100(ADD X,D)
Le Registre d'Instruction:
Il reçoit l'instruction qui doit être exécutée.
Il est composé de 2 parties:
- le Code Opération de l'instruction
- l'adresse de l'opérande (ou l'opérande)
ACC
RI
COP
ADR

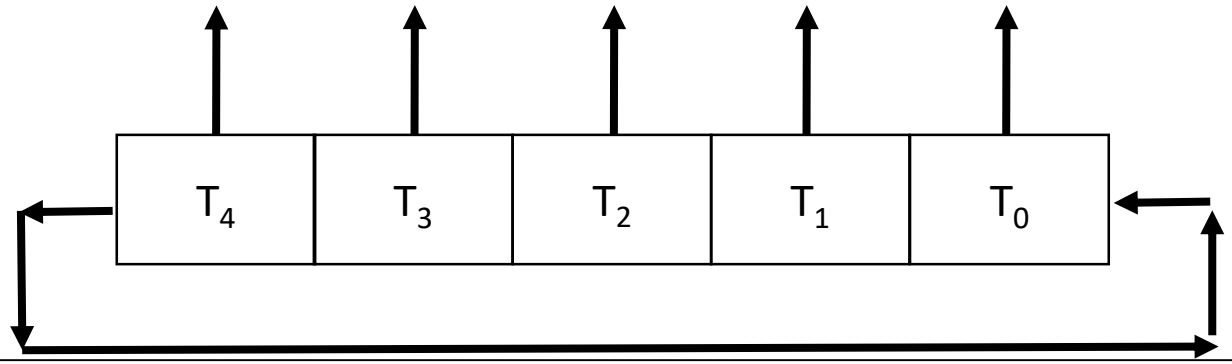
Cycle d'exécution




Unité centrale de traitement (CPU)
Mémoire Centrale
CO
UAL
add
mul
div
seq
Dec
ACC
60
R
COP
ADR
01001…100(ADD X,D)
MAR
@X
X
100
MBR


Unité centrale de traitement (CPU)
Mémoire Centrale
Le Séquenceur
Il contrôler la séquence d'exécution d'une instruction.
Il est généralement décrit par un registre à décalage circulaire dont un seul bit est égal à 1
UAL
div
seq
Dec
ACC
60
RI
COP
ADR
MBR


0
0
0
0
0
T4
T3
T2
T1
T0



Cycle Recherche *(rechercher l'instruction à traiter)*

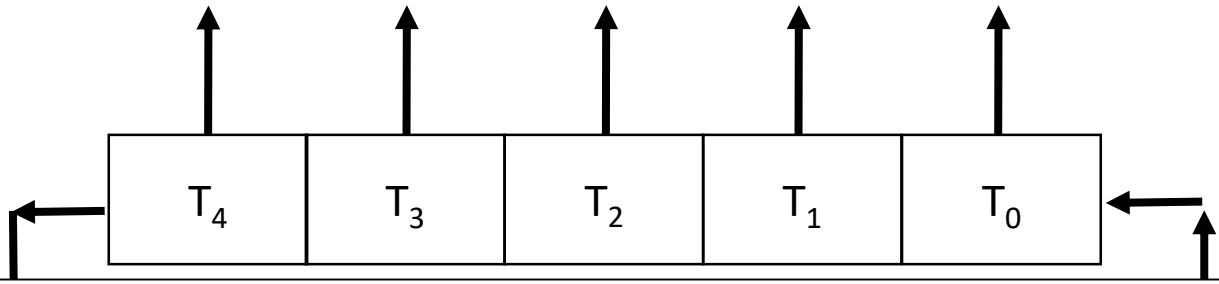
$\overline{F}$ t_0 : MAR ← (CO) *(mettre le contenu du CO dans le registre MAR)*

$\overline{F}$ t_1 : MBR ← Mot *(lecture de l'instruction à partir de la mémoire)*

$\overline{F}$ t_2 : RI ← (MBR) *(transfert du contenu du MBR dans le registre RI)*

$\overline{F}$ t_3 : Décodage *(décoder le code opération qui se trouve dans le RI)*

$\overline{F}$ t_4 : CO ← (CO) + 1 et F ← 1 *(passer au cycle exécution)*



Cycle exécution *(exécution de l'instruction)*

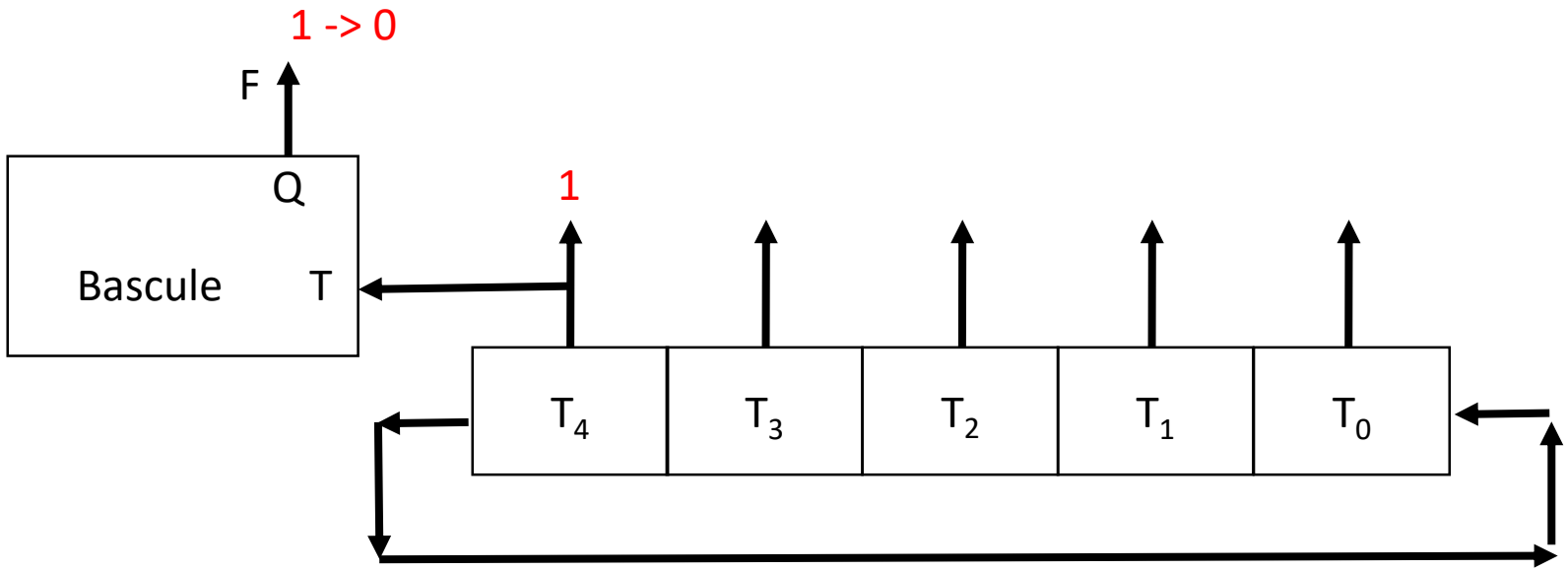
F t_0 : MAR ← (RI : ADR) *(mettre la partie Adresse du RI dans le registre MAR)*

F t_1 : MBR ← Opérande *(lecture de l'opérande à partir de la mémoire)*

F t_2 : UAL ← (MBR) *(envoi de l'opérande vers l'UAL)*

F t_3 : Acc ← (Acc) + (MBR) *(additionner le contenu de l'Acc et le contenu du MBR)*

F t_4 : F ← 0 *(passer au cycle recherche)*


1 -> 0
F
Q
Bascule
T
1
T4
T3
T2
T1
T0

Les instructions

Chaque microprocesseur possède un nombre limité d'instructions qu'il peut exécuter.

Ces instructions s'appellent jeu d'instructions.

Les instructions peuvent être classifiées en 4 catégories :

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

Codage d’une instruction :

Les instructions sont stockées dans la mémoire.

La taille d’une instruction est égale à la taille d’un mot mémoire.

L’instruction est découpée en deux parties :

- Code opération qui indique quelle instruction exécuter.
- L’opérande : qui contient la donnée ou l’adresse de la donnée concernée par l’opération à exécuter.

Code opération	L’opérande

Machine à 3 opérandes :

Code opération	L'opérande1	L'opérande2	L'opérande3
ADD	A	B	C

C ← A+B

Machine à 2 opérandes :

Code opération	L'opérande1	L'opérande2
ADD	A	B

B ← A+B

Machine à 1 opérande :

Code opération	L'opérande
ADD	A

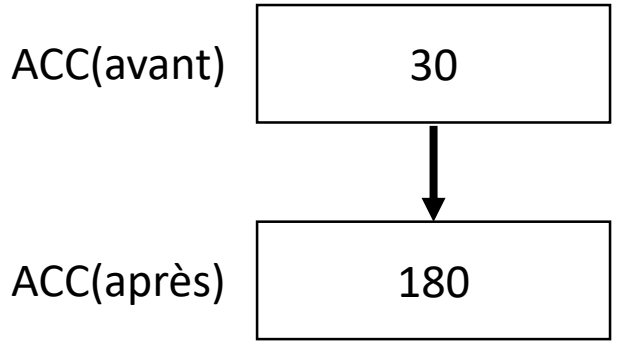
ACC ← ACC+A

Mode d’adressage

1- Adressage immédiat

Code opération	L’opérande1
ADD	150,IMM

ACC ← (ACC) + 150

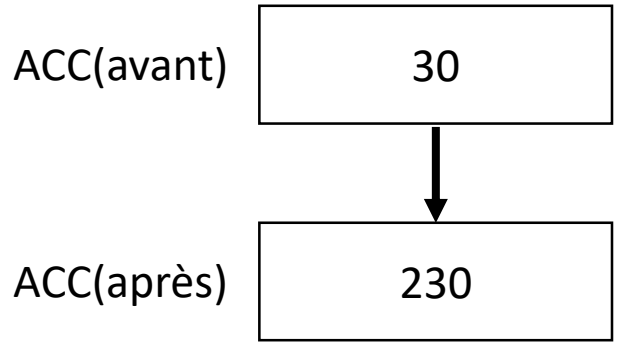


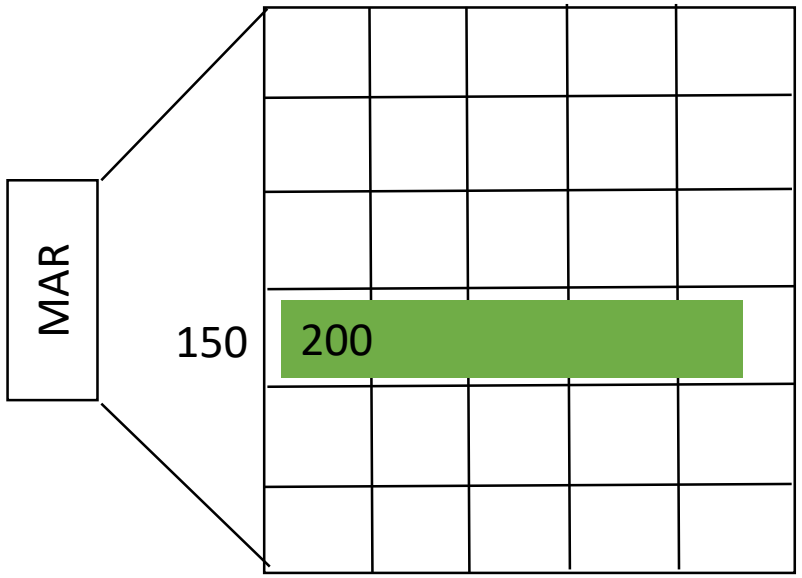
Mode d’adressage

2- Adressage Direct

Code opération	L’opérande1
ADD	150,D

ACC ← (ACC) + (150)





Mode d'adressage

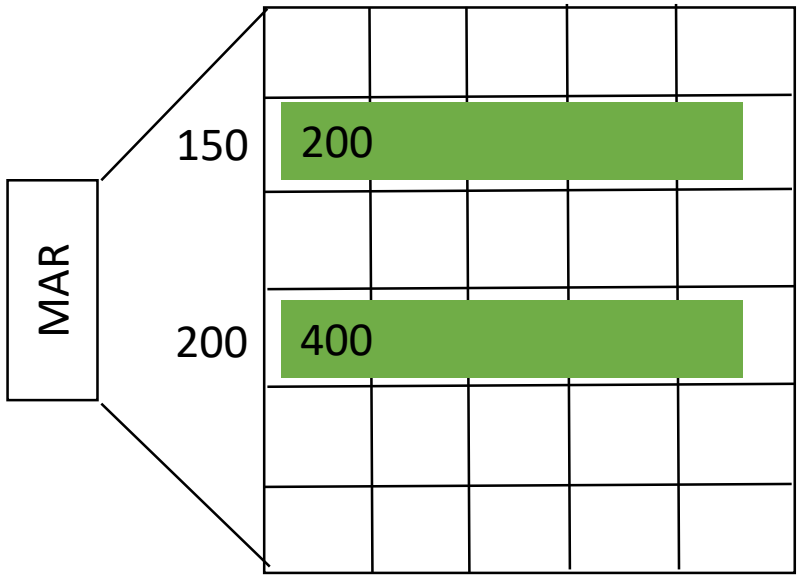
3- Adressage Indirect

Code opération	L'opérande1
ADD	150,IND

ACC ← (ACC) + ((150))

ACC(avant) 30

ACC(après) 430



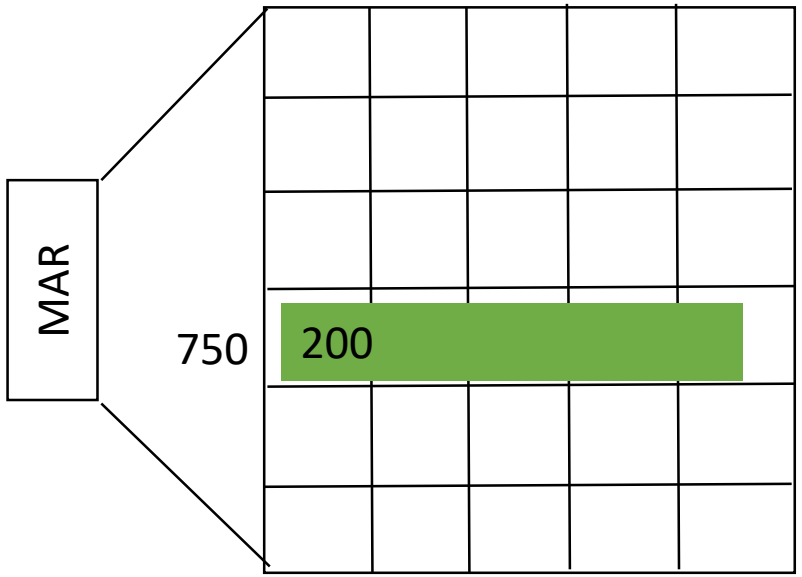
Mode d’adressage

4- Adressage Indexé

Code opération	L’opérande1
ADD	150,XRI

ACC ← (ACC) + (150+(XRI))

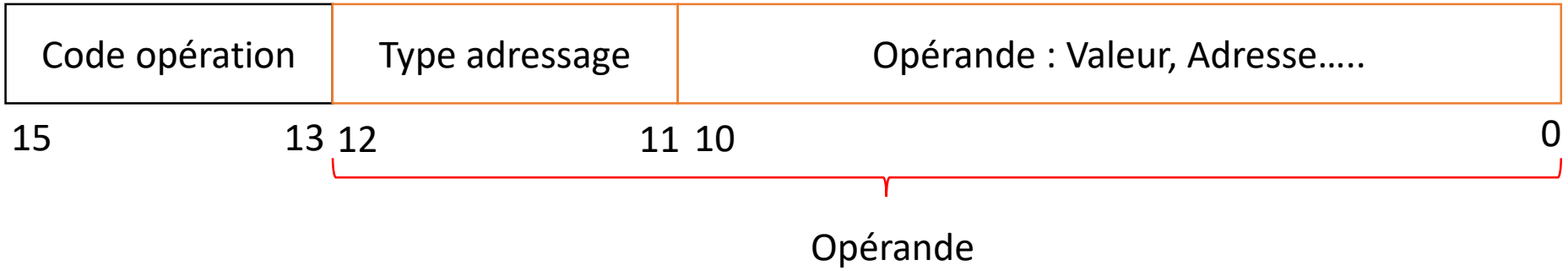
XRI(avant)	600
ACC(avant)	30
ACC(après)	230



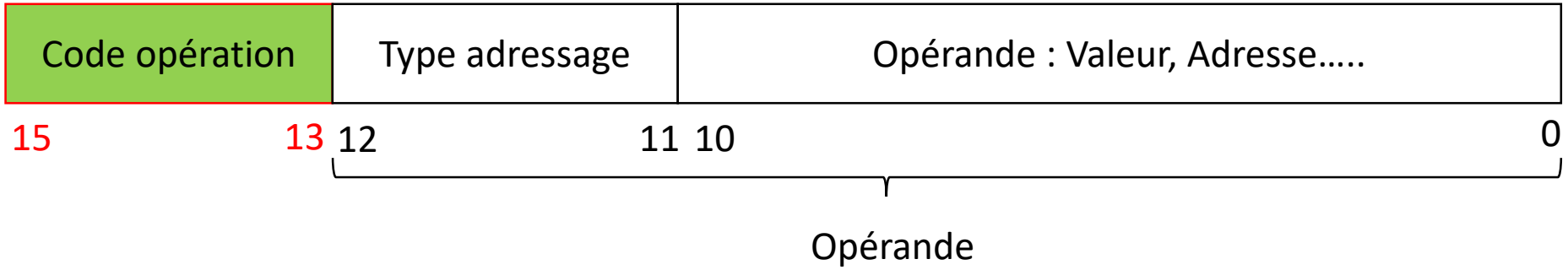
Format d'une instruction

La mémoire de la machine étudiée dans ce cours est représentée par une RAM 2K x 16. (RAM de 2^{11} mots de 16 bits)

Chaque instruction est codée en binaire sur 16 bits selon le format suivant :



Format d'une instruction

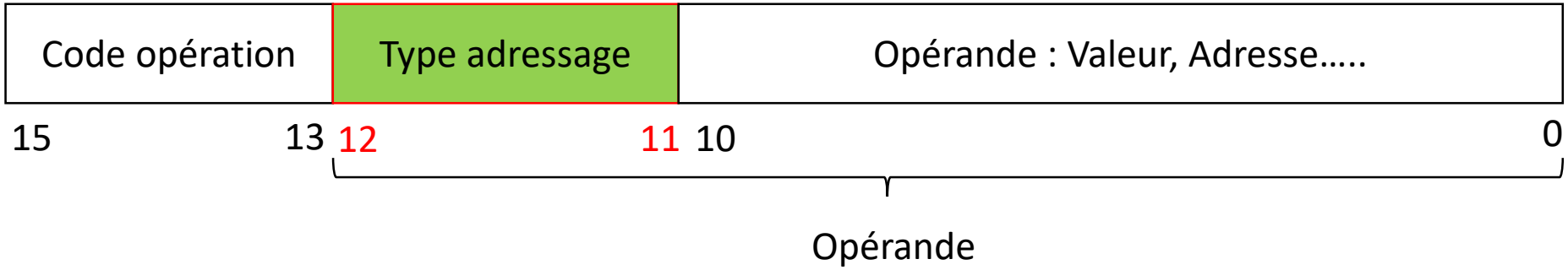


Le champ code opération est sur 3 bits (de 13 à 15).

On peut donc utiliser au maximum 8 opérations sur cette machine:

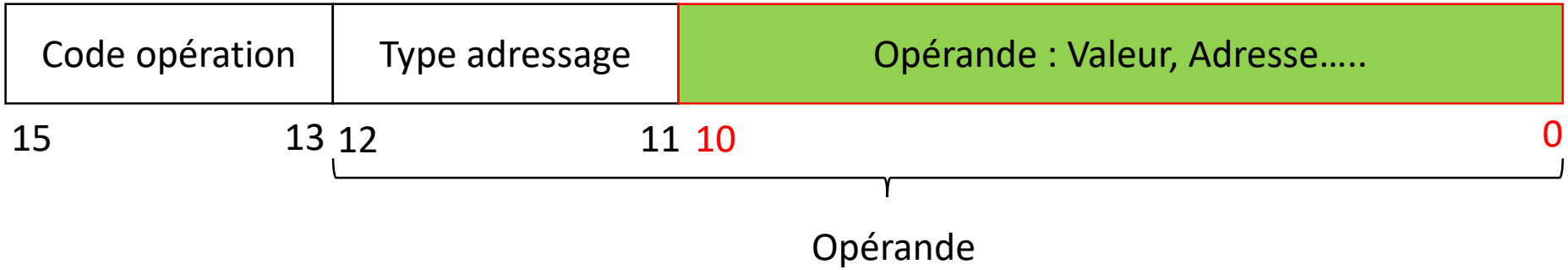
000 : LOAD	100 : MUL
001 : STORE	101 : DIV
010 : ADD	110 : READ
011 : SUB	111 : WRITE

Format d'une instruction



Le champ adressage est sur 2 bits (10, 11).
On utilisera :
- 00 pour l'adressage immédiat,
- 01 pout l'adressage direct
- 10 pour l'adressage indirect
- 11 pour l'adressage indexé

Format d'une instruction



Le champ opérande est sur 11 bits de (0 à 10).

Il contient généralement une adresse (binaire) ou parfois une donnée en adressage immédiat.

Le MAR de cette RAM sera constitué de 11 bits pour accéder à 2^{11} adresses.
Le MBR de cette RAM sera constitué de 16 bits.

Un programme est une suite d'opérations

Format d'une instruction

Instructions	Signification	Code Opération
LOAD X, D	Acc ← (X)	0 0 0
STORE X, D	Mémoire ← (Acc)	0 0 1
ADD X, D	Acc ← (Acc) + (X)	0 1 0
SUB X, D	Acc ← (Acc) - (X)	0 1 1
MUL X, D	Acc ← (Acc) x (X)	1 0 0
DIV X, D	Acc ← (Acc) / (X)	1 0 1
READ	Acc ← val du périphérique d'entrée	1 1 0
WRITE	périphérique de sortie ← (Acc)	1 1 1

Format d'une instruction

ADD val, IMM	Additionner val avec le contenu de l'Acc	**Acc←(Acc)+val**
ADD xxx, D	Additionner le contenu de l'@ xxx au contenu de l'Acc	**Acc←(Acc)+(xxx)**
ADD xxx, IND	Additionner le contenu du contenu de xxx au contenu de l'Acc	**Acc←(Acc)+((xxx))**
ADD xxx, XRI	Additionner le contenu de (xxx+le contenu deXRI) au contenu de l'Acc	**Acc←(Acc)+(xxx+(XRI))**