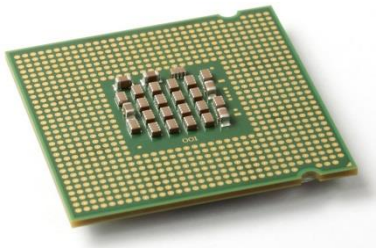
0001 1011

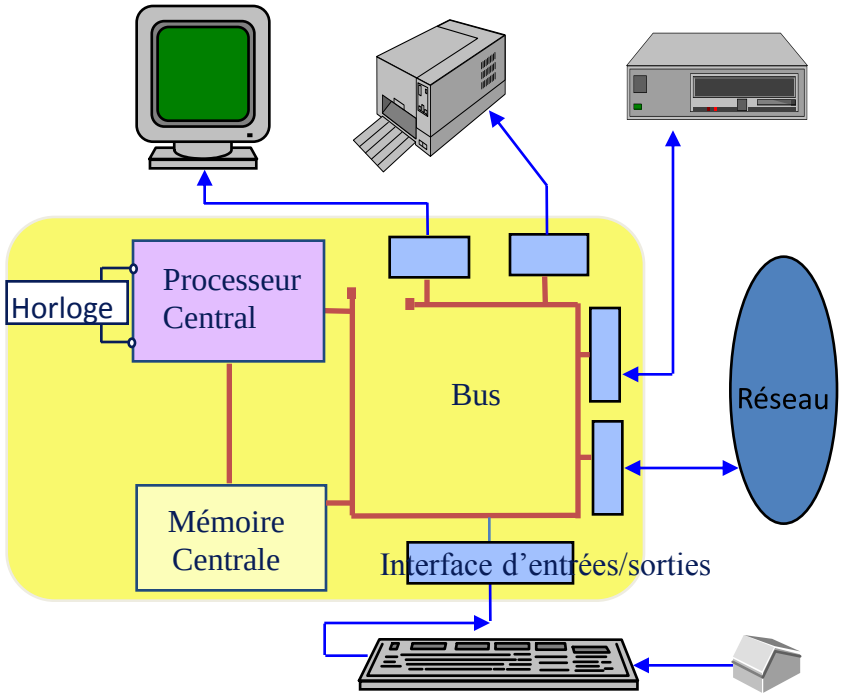
Le langage de l'ordinateur
Le langage du processeur
Les instructions machine

0001 1011

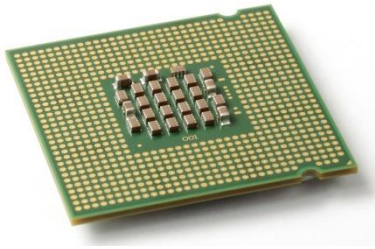


Les fonctions de l'ordinateur
Les composants

- Des éléments permettant la communication entre l'ordinateur et l'être humain : ce sont les **périphériques**.
- Un élément permettant d'exécuter les instructions d'un programme : c'est le **processeur** (CPU).
- Des éléments permettant de stocker les données : ce sont les **mémoires** de l'ordinateur.
- Des éléments permettant aux différents composants (périphériques, processeur, mémoire) de l'ordinateur de communiquer : ce sont les **bus** de l'ordinateur



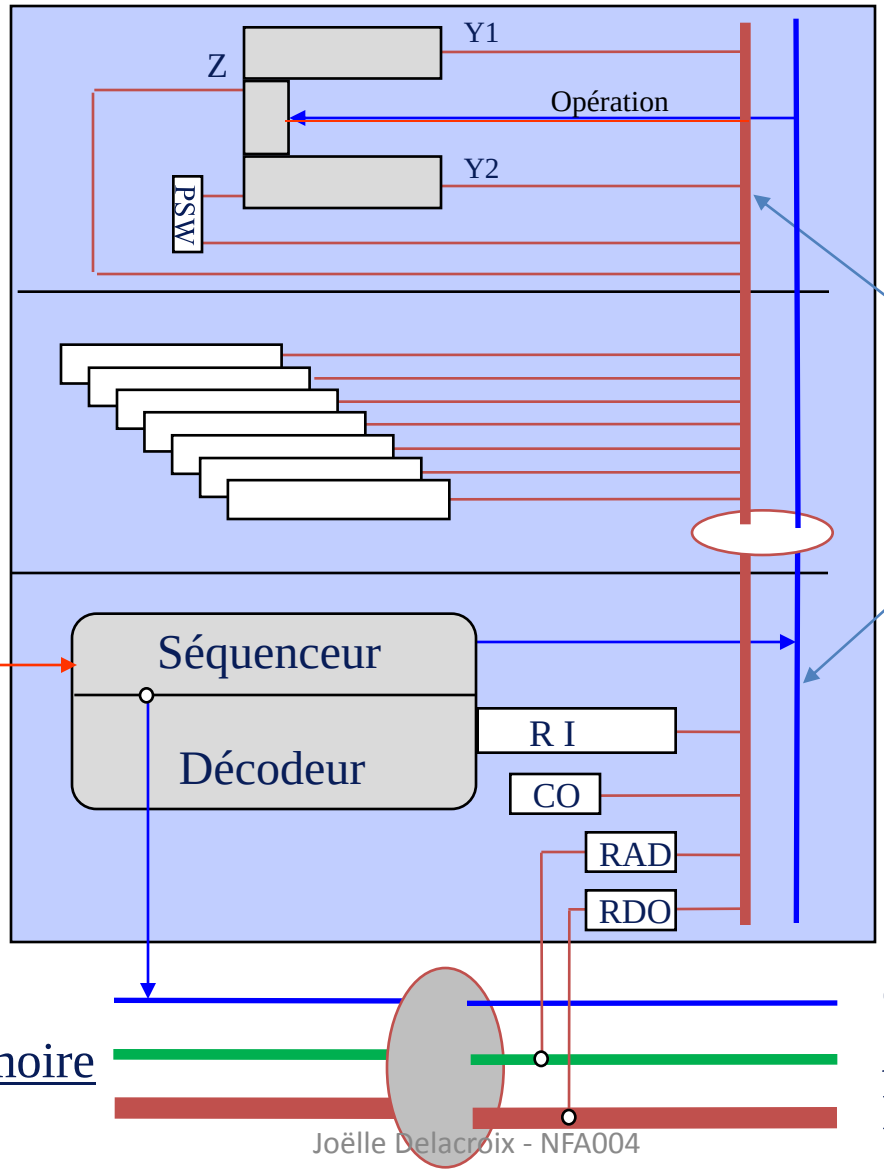
Processeur (Unité Centrale)



Unité Arithmétique et Logique

Registres

Unité de Commande



Bus Interne

Données

Commandes

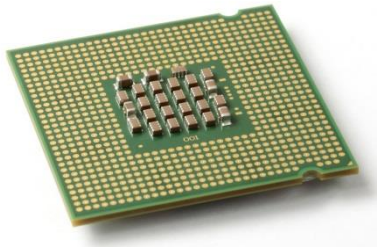
Bus Processeur /Mémoire

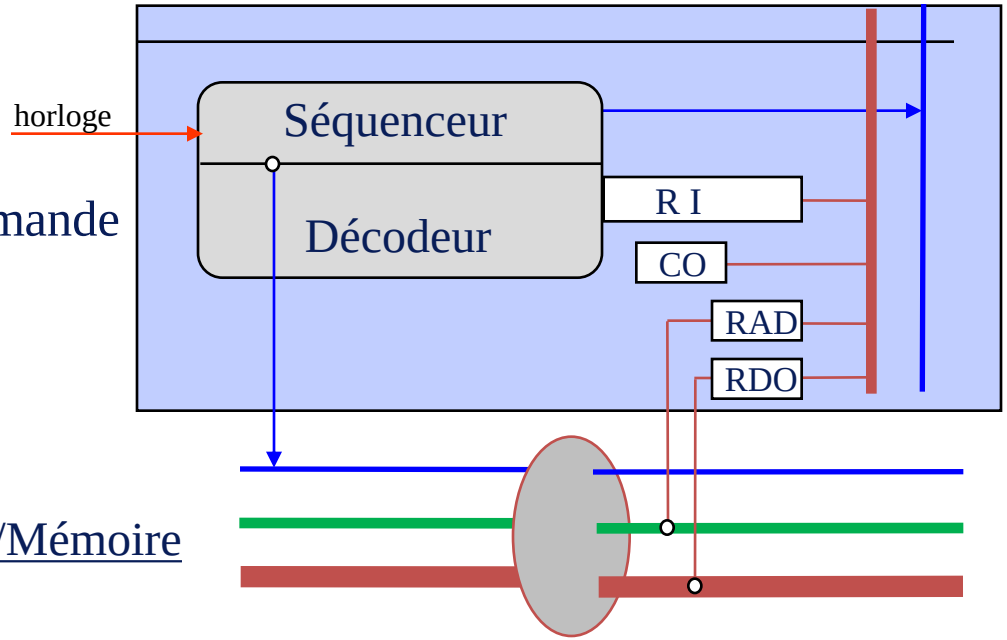
Commandes Lecture/Ecriture

Adresses

Données

Processeur (Unité Centrale)





Commandes Lecture/Ecriture
Adresses
Données

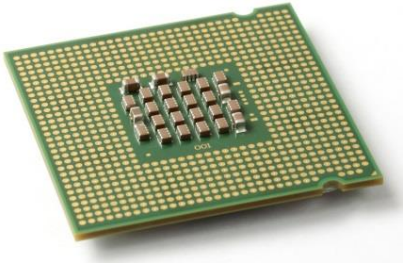
Bus Processeur /Mémoire

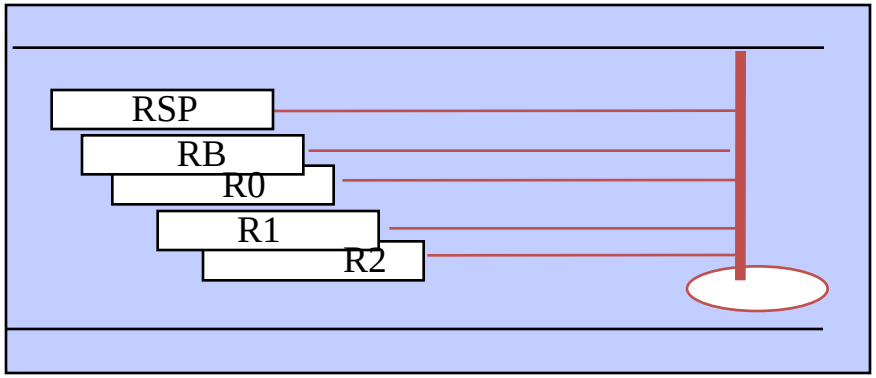
L'unité de commande est chargée de la reconnaissance des instructions et de leur exécution par l'unité de traitement au rythme de l'horloge

Les registres :

- RI (registre instruction) : contient l'instruction en cours d'exécution
- CO (compteur ordinal) : contient l'adresse en MC de la prochaine instruction
- RAD (registre adresse) et RDO (registre de données) : registres d'interfaçage avec la mémoire centrale

Processeur (Unité Centrale)



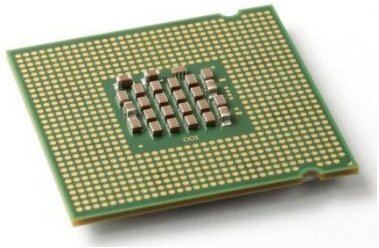


- Le registre est l'entité de base manipulée par le processeur.
- Aucune opération n'est directement réalisée sur les cellules mémoires.

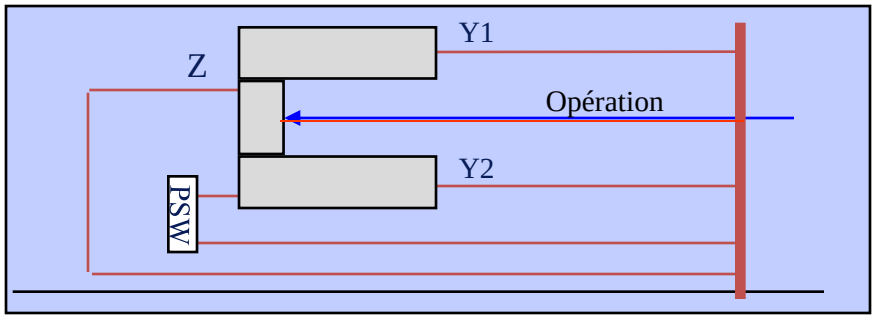
Registres :

- les registres généraux R0, R1, R2
- le registre de pile RSP (Register Stack Pointer)
- les registres d'adressage : RB (registre de base)

Processeur (Unité Centrale)



Unité Arithmétique et Logique



L'unité Arithmétique et Logique (UAL) constitue l'unité d'exécution du processeur. Elle est composée :

- de l'ensemble des circuits permettant de réaliser les opérations arithmétiques (addition, multiplication, division,…) et les opérations logiques (complément à 2, inverse, OU, ET, …) sur les opérandes Y1 et Y2
- d'un registre d'état PSW qui contient des indicateurs positionnés par le résultat Z des opérations effectuées :

S	O	C	Z	

- O : positionné à 1 si Overflow, 0 sinon
- Z : positionné à 1 si résultat opération nul, 0 sinon
- C : positionné à 1 si carry, 0 sinon
- S : positionné à 0 si résultat opération positif, 1 sinon

La mémoire centrale

00000	00001	00010	00011
$(0A)_{16}$	$(FE)_{16}$	$(10)_{16}$	$(BA)_{16}$
00100	00101	00110	00111
$(FA)_{16}$	$(11)_{16}$	$(34)_{16}$	$(57)_{16}$
01000	01001	01010	01011
01100	01101	01110	01111
10000	10001	10010	10011
10100	10101	10110	10111
$(FA)_{16}$	$(11)_{16}$	$(34)_{16}$	$(57)_{16}$
11000	11001	11011	11100
11101	11110	11110	11111

adresse

adresse

adresse

adresse

adresse

adresse

adresse

8 mots de 4 octets à adresser : 32 octets = 2^5
Les adresses doivent être sur 5 bits au moins

- La mémoire centrale est constituée par un ensemble de mots mémoire. Un mot est constitué par un ensemble d'octets.
- Chaque octet est repéré de manière unique par une adresse; la mémoire est dite « adressable par octet ».
- Elle s'interface avec le processeur via un bus

Adresse
Commande
données

La mémoire centrale

Gros Boutiste et Petit Boutiste

Un mot composé de plusieurs octets peut être écrit selon deux ordonnancements différents en mémoire centrale :

- Big Endian ou Gros Boutiste : l'octet de poids fort d'un mot est écrit dans l'octet du mot mémoire de plus grande adresse.
- Little Endian ou Petit Boutiste : l'octet de poids fort d'un mot est écrit dans l'octet du mot mémoire de plus petite adresse.

Exemple : Mot $(87ADFCE5)_{16}$

Adresse	00	01	02	03
petit boutiste	87	AD	FC	E5
gros boutiste	E5	FC	AD	87

La mémoire centrale

Alignement en mémoire centrale

Un mot de p octets est dit aligné si son adresse est un multiple de p. L'allocation alignée entraîne une perte de place en mémoire centrale, mais l'accès à la mémoire est plus rapide si les mots sont alignés.

$(87AD)_{16}$ est un mot de 4 octets ; $(BC)_{16}$ est un mot de deux octets

Cas a mots non alignés			Cas b : mots alignés	
Adresse	Valeur		Adresse	Valeur
8	B		8	B
9	C		9	C
10	8		10	
11	7		11	
12	A		12	8
13	D		13	7
14			14	A
15			15	D

Le bus processeur / mémoire

Il existe aussi au sein du processeur un bus dit bu interne qui relie les différents composants du processeur

Le bus processeur/mémoire permet la communication entre le processeur et la mémoire centrale. Au niveau du processeur cet interfaçage est mis en œuvre via les registre RAD et RDO.

Le bus est composé :

- de lignes d'adresses qui transportent l'adresse à laquelle le processeur s'intéresse
- de lignes de données qui transportent les mots mémoires
- de lignes de commandes qui spécifient le type d'opérations en cours sur le bus

On appelle largeur du bus le nombre de bits que le bus peut transporter en parallèle.

Le programme machine

- Le rôle du processeur consiste à exécuter un programme.

- Les instructions et les données sur lesquelles elles travaillent sont placées dans les mots de la mémoire centrale.

- Au niveau matériel, les instructions exécutées par le processeur constitue le jeu d'instructions du processeur. Ce sont des chaînes binaires.

Les niveaux de langage de programmation

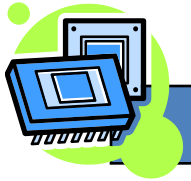
Programme en langage haut niveau
(indépendant machine physique)

```
While (x > 0)
do
        y := y + 1;
        x := x - 1;
done;
```

COMPILATEUR

Programme en langage d'assemblage
(très proche du langage machine)

```
loop : add  R1, 1
       sub R2, 1
       jmpP loop
```



ASSEMBLEUR

Programme en binaire
(langage machine)

```
1000 : 00000110000100000000001
       00000110001000000000001
       01100000000000000001000
```

Le programme machine

Programme en langage de haut niveau
instructions de haut niveau

Compilateur

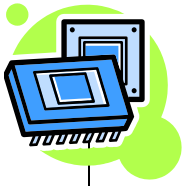
Niveau utilisateur

Machine physique "matérielle"

Programme en langage d'assemblage

Assembleur

processeur



Bus

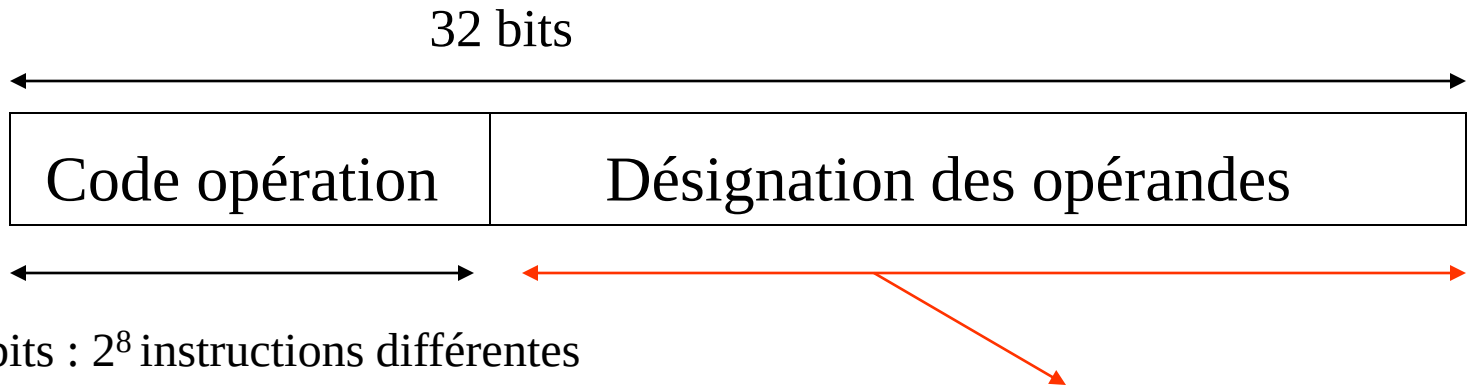
01101110111110011
01111010001011100
10111101110111111
00111011110111011
00111111000111101

Mémoire centrale

Programme à
exécuter : instructions machine
et valeurs en binaire

traduction

Le programme machine



8 bits : 2^8 instructions différentes

00000000 : chargement de registre

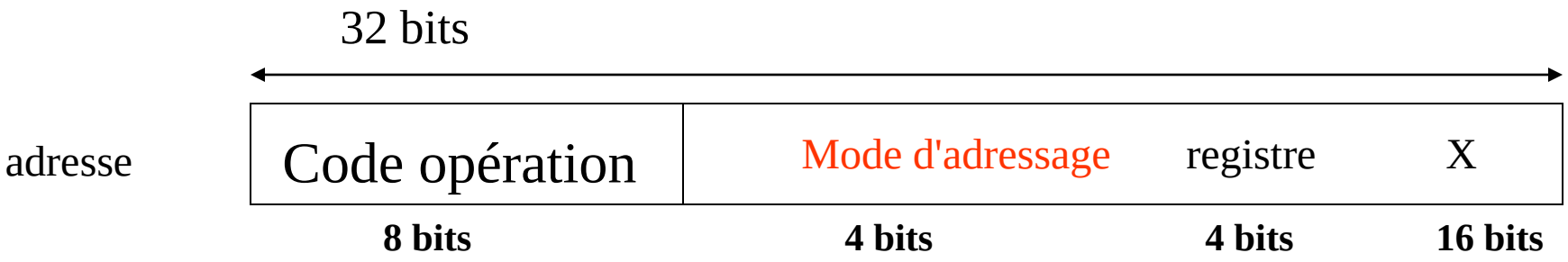
00000101 : addition, etc...

Ier opérande	2ème opérande
Registre	registre valeur immédiate opérande en mémoire

Le mode d'adressage et le champ X permettent de code le type du deuxième opérande

Mode d'adressage	registre	X

Le programme machine



Une instruction machine est une chaine binaire repérée par une adresse.
Pour faciliter la manipulation du langage machine par l'humain, on substitue à la chaine binaire une représentation "alphanumérique" équivalente pour chaque instruction : c'est le langage d'assemblage qui remplace les chaines binaires par des mnémoniques équivalents

00001000 00000101 0000 0001 00000000000000000000
addition : ADD Im R1 0

etiquette : codeop mode adressage registre X

Le programme machine

- Les modes d'adressage

mode	Mémmonique	Signification de X
Immédiat	Im	X est la valeur du deuxième opérande
Direct	D	X est une adresse; X est l'adresse d'un mot qui contient le second opérande
Indirect	I	X est une adresse; X est l'adresse d'un mot qui contient *l'adresse* du second opérande
Basé	B	X est un déplacement; l'adresse du second opérande est calculée comme étant : (RB) + X
Registre/Registre	Reg2	X est un numéro de registre
Registre	Reg 1	X est sans signification; il n'y a pas de second opérande

Le programme machine

- Mode d'adressage : adressage Immédiat (Im)

Load Im R5 2

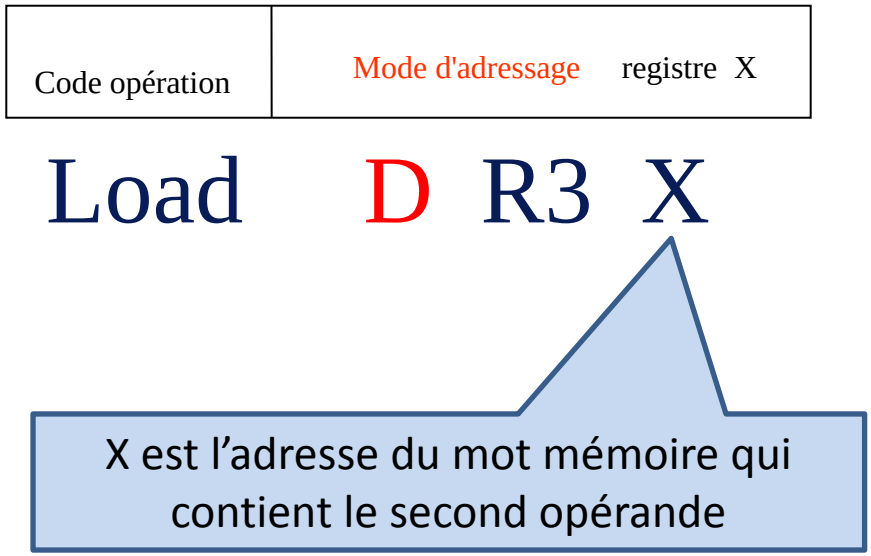
Chargement de registre.
Le premier opérande registre (R5) est chargé avec le second opérande

2 est la valeur de l 'opérande.
L'opérande est dit « immédiat »

R5 ← 2; le registre R5 est chargé avec la valeur 2

Le programme machine

- Mode d'adressage : adressage Direct (D)

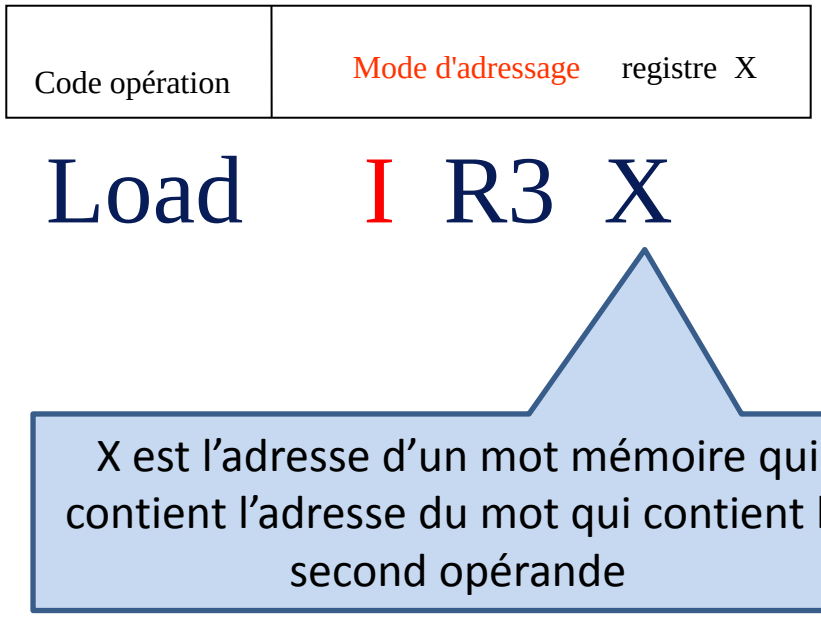


adresse	contenu
X	123

R3 ← (X); R3 ← 123; R3 est chargé avec le contenu du mot mémoire d'adresse X

Le programme machine

- Mode d'adressage : adressage Indirect (I)



adresse	contenu
X	Y
Y	50

R3 ← ((X)); R3 < (Y); R3 ← 50; R3 est chargé avec le contenu du mot mémoire dont l'adresse X est contenue dans le mot d'adresse Y

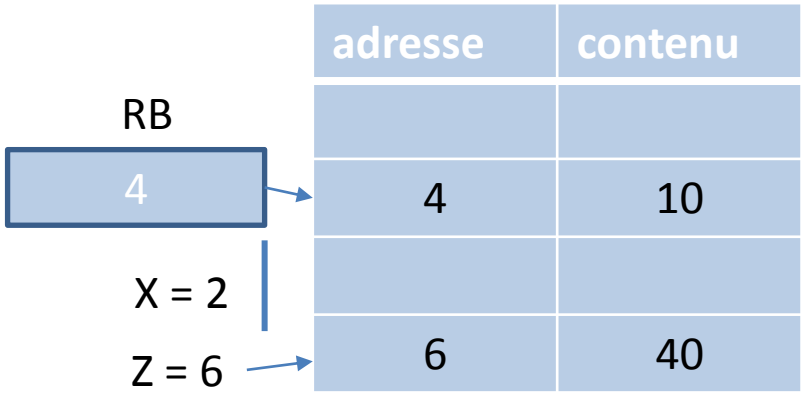
Le programme machine

- Mode d'adressage : adressage Basé (B)

Code opération	Mode d'adressage	registre X

Load B R3 X

X est un déplacement. L'adresse du mot mémoire qui contient le second opérande est égale à (RB) + X



(R3) ← ((RB) + X) ← (Z); R3 ← 40; R3 est chargé avec le contenu du mot mémoire dont l'adresse X est Z (6) ; Z = (RB) + X

Le programme machine

- Synthèse

mode	mnémonique	opérande	
immédiat	Im	Opérande = X	constante
direct	D	Opérande = (X)	variable
indirect	I	Opérande = ((X))	pointeur
Basé	B	Opérande = (RB + X)	tableau

```
main() {
int j;
int tab[3];


tab[2] = *j + 4;
}
```

104	**J**
108	tab[1]
112	tab[2]
116	tab[3]
120	104

```
LOAD Im RB 108
LOAD I R2 120
ADD Im R2 4
STORE B R2 4
```

Le programme machine

- Synthèse

```
main() {
int j;
int tab[3];

tab[2] = *j + 4;
}
```


```
LOAD Im RB 108
LOAD I R2 120
ADD Im R2 4
STORE B R2 4
```

RB ← 108
R2 ← ((120)) ← (104) ← (*j)
R2 ← R2 + 4 = *j + 4
R2 est écrit à l'adresse (RB) + 4
R2 est écrit à l'adresse 112; tab[2] = *j + 4

Le programme machine

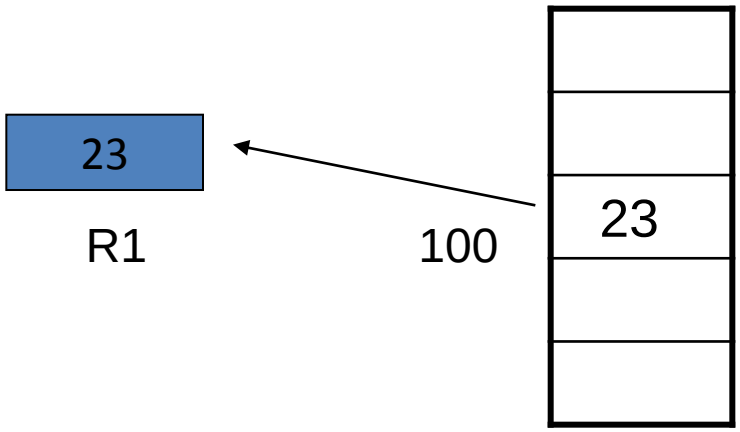
- **Les différents types d'instructions**

Chargement / Stockage de registre LOAD , STORE, MOV
Opérations arithmétiques et logiques mettant en jeu l'UAL ADD, MUL, OR, XOR, AND, ...
Les opérations de rupture de séquence Les sauts : JMP et JMPO, JMPS, JMPZ, …. Les appels de sous programme : CALL et RET

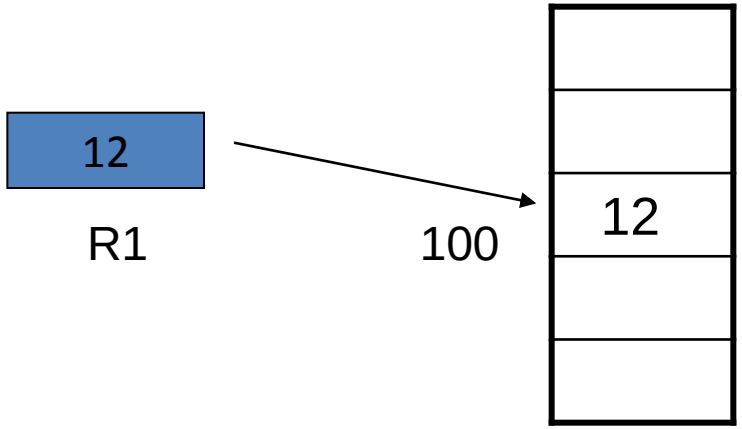
Les différents types d'instructions
Chargement/ Ecriture d'un registre

- **Chargement de registre :**
- LOAD D R1 100 : chargement du registre R1 avec la case mémoire d'adresse 100

- LOAD Im R1 100 : chargement du registre R1 avec la valeur immédiate 100



- **Ecriture de registre :**
- STORE D R1 100 : Ecriture du contenu du registre R1 dans la case mémoire d'adresse 100

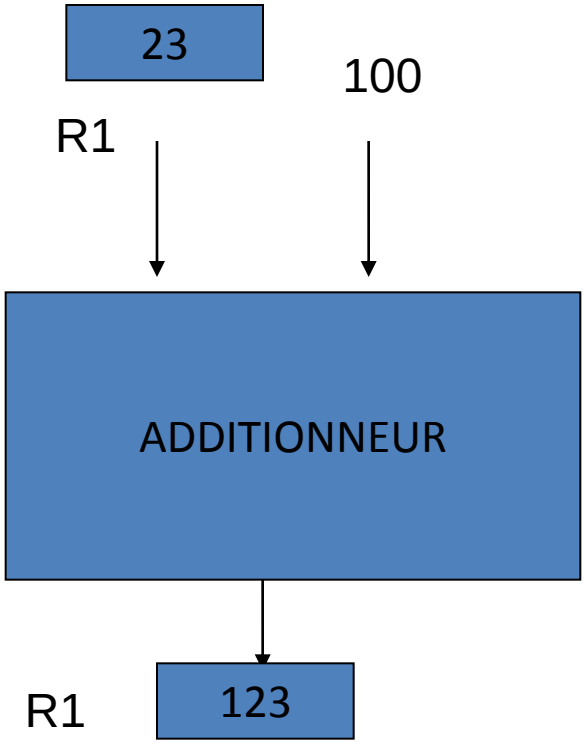


Les différents types d'instructions

Opérations arithmétiques et logiques mettant en jeu l'UAL

ADD, MUL, OR, XOR, AND, ...

ADD Im R1 100 : addition de la valeur immédiate 100 avec le contenu du registre R1 et stockage du résultat dans R1.

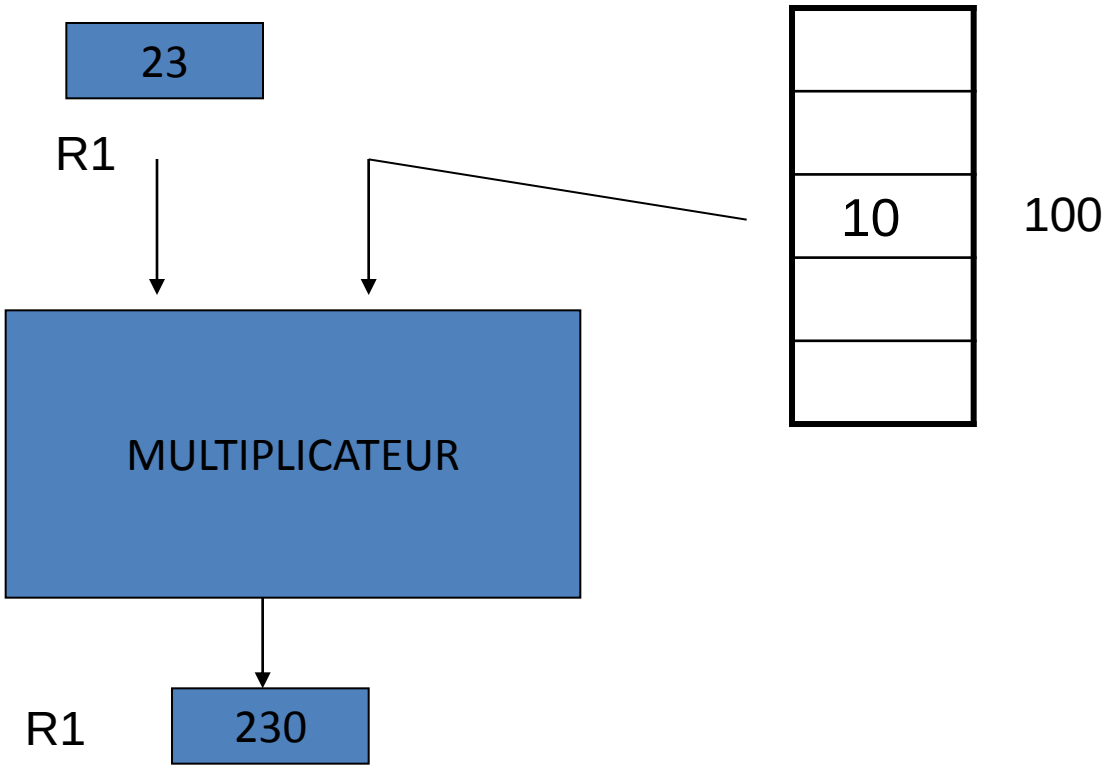


Les différents types d'instructions

Opérations arithmétiques et logiques mettant en jeu l'UAL

ADD, MUL, OR, XOR, AND, ...

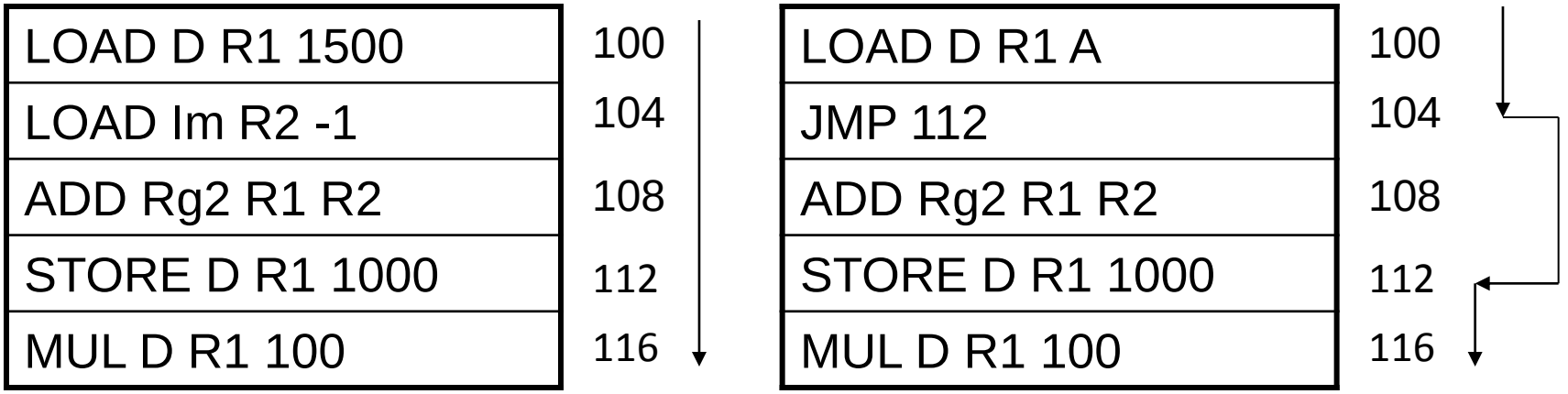
MUL D R1 100 : Multiplication du contenu du mot mémoire 100 avec le contenu du registre R1 et stockage du résultat dans R1.



Les différents types d'instructions

Les opérations de sauts : JMP et JMPO, JMPS, JMPZ, ….

Ces instructions permettent de rompre l'exécution en séquence des instructions pour « sauter » en avant ou en arrière.

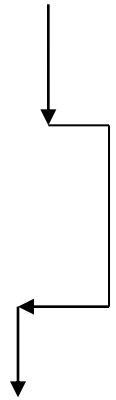


Les différents types d'instructions

Les opérations de sauts : JMP et JMPO, JMPS, JMPZ, ….

- Il existe deux types de saut :
  - Le saut inconditionnel (JMP)
  - Le saut conditionnel (JMPO, JMPS…)

LOAD D R1 1500	100
JMP 112	104
ADD Rg2 R1 R2	108
STORE D R1 1000	112
MUL D R1 100	116



- Saut inconditionnel

➢ Le saut à l'adresse spécifiée dans l'instruction est toujours réalisé.

Les différents types d'instructions

Les opérations de sauts : JMP et JMPO, JMPS, JMPZ, ….

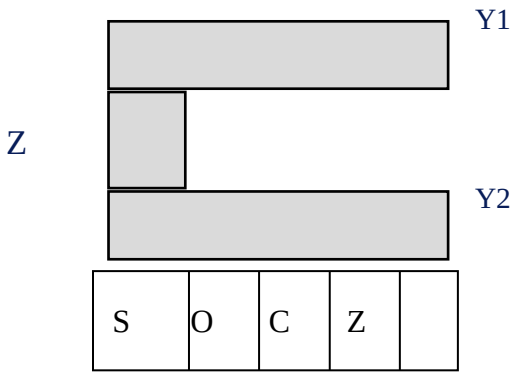
- Il existe deux types de saut :
  - Le saut inconditionnel (JMP)
  - Le saut conditionnel (JMPO, JMPP…)

LOAD D R1 1500	100
LOAD Im R2 10	104
ADD Rg2 R1 R2	108
JMPP 124	112
STORE D R1 1000	116
STOP	120
STORE D R1 2000	124

- Saut conditionnel

➢ Le saut à l'adresse spécifiée dans l'instruction est réalisé si une condition relative aux flags du registre PSW de l'UAL est vraie.

Sinon l'exécution continue en séquence



Les différents types d'instructions

Les opérations de sauts : JMP et JMPO, JMPS, JMPZ, ….

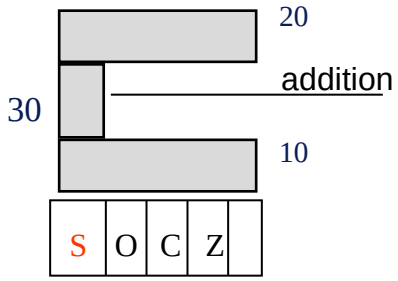
LOAD D R1 1500	100	20	R1
LOAD Im R2 10	104	10	R2
ADD Rg2 R1 R2	108	30	R1
JMPP 124	112		
STORE D R1 1000	116		
STOP	120		
STORE D R1 2000	124		

MC programme

Le résultat est positif ; le saut est réalisé

	1000
20	1500
30	2000

MC données



- Saut conditionnel

➢ Le saut à l'adresse spécifiée dans l'instruction est réalisé si une condition relative aux flags du registre PSW de l'UAL est vraie. Sinon l'exécution continue en séquence

➢ JMPP : saut si le bit S de l'UAL est positionné sur positif

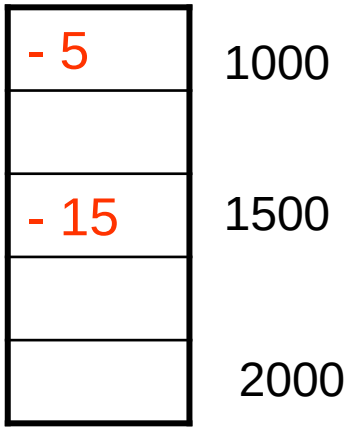
Les différents types d'instructions

Les opérations de sauts : JMP et JMPO, JMPS, JMPZ, ….

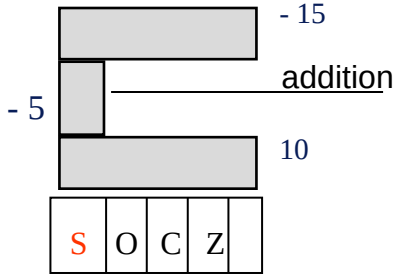
LOAD D R1 1500	100	- 15	R1
LOAD Im R2 10	104	10	R2
ADD Rg2 R1 R2	108	- 5	R1
JMPP 124	112		
STORE D R1 1000	116		
STOP	120		
STORE D R1 2000	124		

MC programme

Le résultat est négatif ;
le saut n'est pas réalisé



MC données



- Saut conditionnel

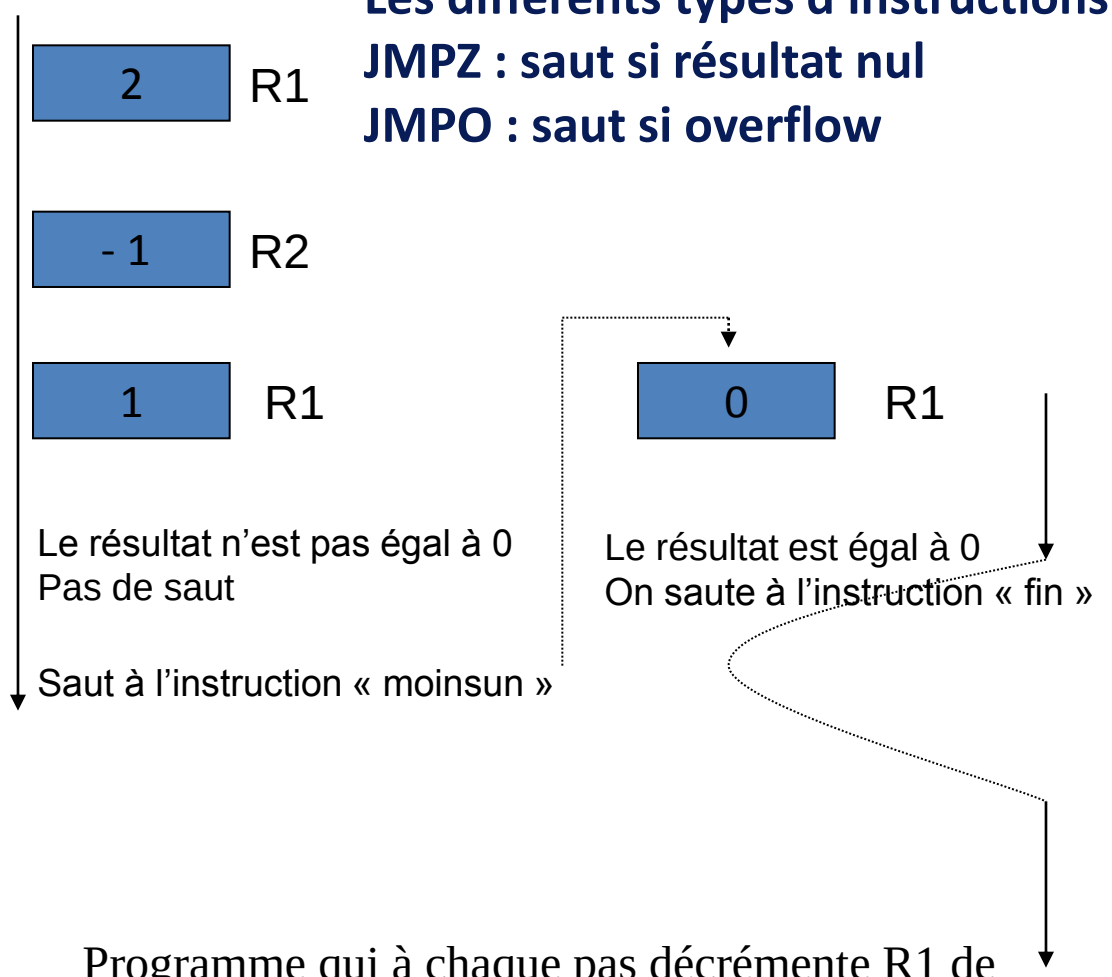
➢ Le saut à l'adresse spécifiée dans l'instruction est réalisé si une condition relative aux flags du registre PSW de l'UAL est vraie. Sinon l'exécution continue en séquence

➢ JMPP : saut si le bit S de l'UAL est positionné sur positif

Les différents types d'instructions

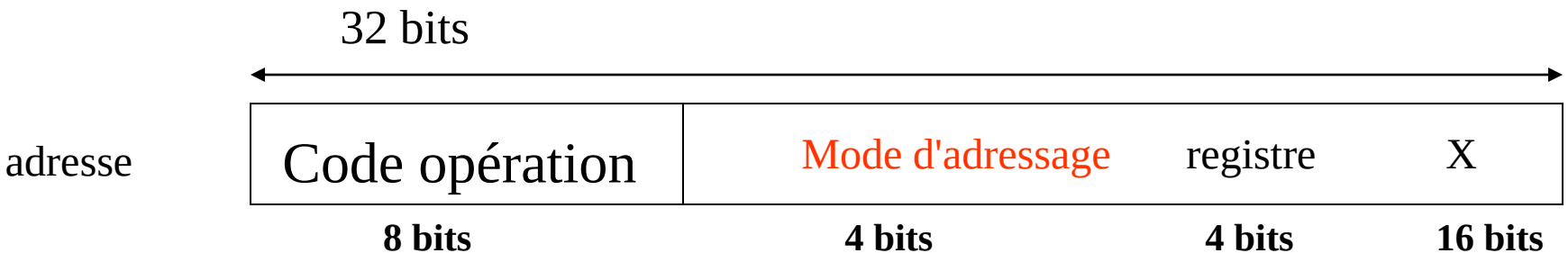
Les opérations de sauts : JMP et JMPO, JMPS, JMPZ, ….

LOAD Im R1 2
LOAD Im R2 -1
moinsun : ADD Rg2 R1 R2
JMPZ fin
JMP moinsun
fin : STOP



Programme qui à chaque pas décrémente R1 de 1 unité et s'arrête dès que R1 = 0

Le Langage d'assemblage



• Une instruction machine est une chaine binaire repérée par une adresse.
Pour faciliter la manipulation du langage machine par l'humain, on substitue à la chaine binaire une représentation "alphanumérique" équivalente pour chaque instruction : c'est le langage d'assemblage qui remplace les chaines binaires par des mnémoniques équivalents

00001000 00000101 0000 0001 00000000000000000000
addition : ADD Im R1 0

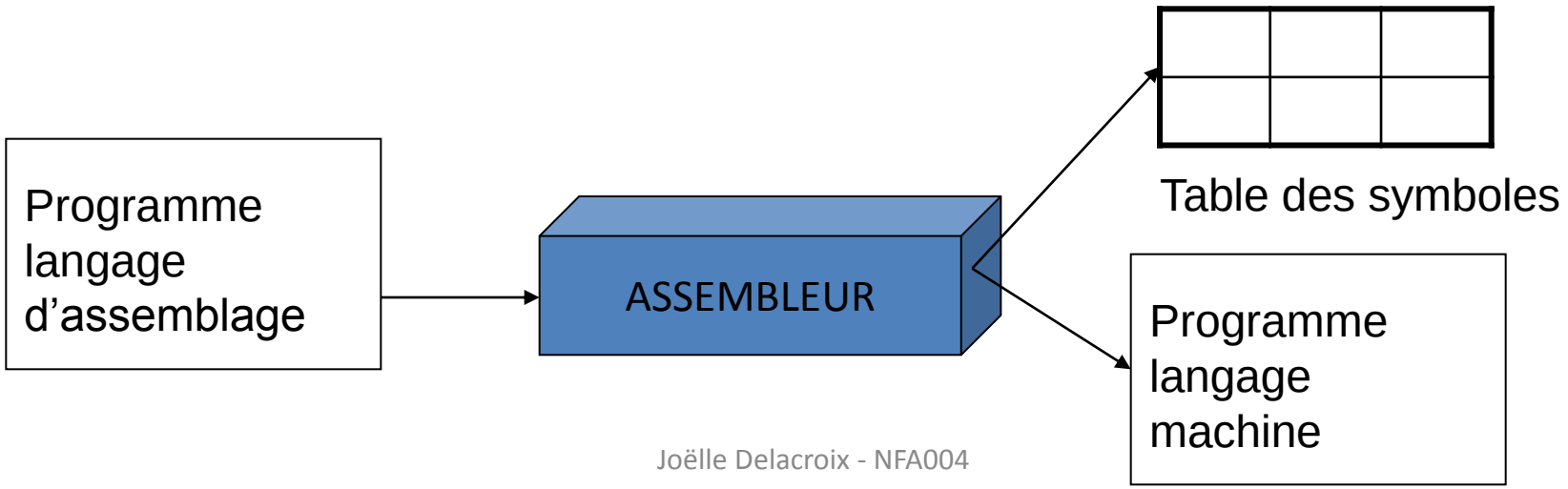
etiquette : codeop Opérandes

Le Langage d'assemblage

var : DS 1 Var : DS 1 = 12	définition de la variable var et réservation d'un mot Initialisation à 12	directive
LOAD Im R1 127	R1 ← 127	
boucle : ADD Im R1 1	R1 ← R1 + 1	
STORE D R1 var	R1 est écrit à l'adresse var	
JMP boucle	Retour à l'instruction ADD	
STOP	fin	directive

Le Langage d'assemblage

- L'assembleur est un programme qui traduit le langage d'assemblage en langage machine.
- L'assembleur travaille en deux passes.
  - Lors de la première passe, l'assembleur rassemble l'ensemble des symboles et étiquettes dans une table et leur associe une adresse dans le code.
  - Lors de la deuxième passe, l'assembleur génère le langage machine en utilisant la table construite lors de la première passe.



Le Langage d'assemblage

- Lors de la première passe, l'assembleur rassemble l'ensemble des symboles et étiquettes dans une table et leur associe une adresse dans le code.
- Pour ce faire, l'assembleur manipule un compteur appelé **compteur d'emplacement**, mis à 0 et incrémenté de la longueur de l'instruction ou de la longueur de la variable à chaque instruction ou déclaration traitée.

	Compteur
var : DS 1	0
LOAD Im R1 127	4
boucle : ADD Im R1 1	8
STORE D R1 var	12
JMP boucle	16
STOP	

Adresse	Nom du symbole
0	var
8	boucle

Table des symboles

Le Langage d'assemblage

- Lors de la seconde passe, l'assembleur génère les instructions en langage machine
- Pour cela, l'assembleur remplace chaque mnémonique par son code binaire, chaque étiquette ou variable par son adresse et chaque constante par sa valeur. Pour effectuer ce travail, l'assembleur utilise la table des symboles construite lors de la première passe.

		Compteur
var :	DS 1	0
	LOAD Im R1 127	4
boucle:	ADD Im R1 1	8
	STORE D R1 var	12
	JMP boucle	16
	STOP	

Adresse	Nom du symbole
0	var
8	boucle

Table des symboles

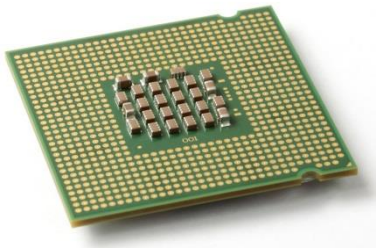
0 :
4 : 00000000 0000 0001 0000000001111111
8 : 00001001 0000 0001 0000000000000001
12 : 10001110 0001 0001 0000000000000000
16 : 11110000 xxxx xxxx 0000000000001000

0001 1011

PARTIE 1
Pour s'entrainer…

-

0001 1011



Question 1

Si on considère une représentation des nombres signés en complément à 2 sur 8 bits, l'intervalle des nombres représentables est

1. [-127, +127]
2. [0, 256]
3. [-128, +127]

Question 2

Si on considère une mode d'adressage direct, le champ X de l'instruction LOAD D R1 X code

1. une valeur immédiate
2. l'adresse en mémoire centrale d'un mot contenant l'opérande
3. l'adresse en mémoire centrale d'un mot contenant lui-même l'adresse du mot contenant l'opérande
4. un numéro de registre

Question 3

L'instruction JMPO X réalise un saut à l'adresse X

1. si la dernière opération arithmétique a mis à vrai l'indicateur d'overflow de l'UAL
2. si la dernière opération arithmétique a produit un résultat en dehors de l'intervalle des nombres représentables
3. si le résultat de la dernière opération arithmétique est nul
4. toujours

Question 4

Une chaine de 8 bits

1. permet de coder 128 états différents, cette chaine s'appelle un octet
2. permet de coder 256 états différents, cette chaine s'appelle un octet
3. permet de coder 256 états différents, cette chaine s'appelle un quartet

Question 5

La chaine 1000 1001 code la valeur

1. + 137 en binaire
2. (8A) en base 16
3. - 9 selon la convention de la valeur signée sur 8 bits
4. + 137 selon la convention du complément à 2 sur 8 bits
5. -119 selon la convention du complément à 2 sur 8 bits
6. (211) en base 8

Question 6

L'instruction ADD Im R1 X

1. Effectue une addition entre R1 et un opérande contenu dans le mot d'adresse X
2. Effectue une addition entre R1 et la valeur immédiate X
3. Effectue une addition entre R1 et un opérande dont l'adresse est contenue dans le mot d'adresse X

On considère à un instant t, l'image initiale suivante de la mémoire centrale et des registres du processeur d'un ordinateur (les valeurs sont exprimées en base 10).

On sait que les données manipulées par la machine sont codées sur 8 bits selon la norme du complément à 2. Les adresses mémoires sont elles sur 16 bits.

Adresse	Contenu
100	10
101	100
200	
201	127
302	– 44
303	

Registre	Contenu
RB	100
PSW	00000xxxxxxx
R1	20

Soient les opérations suivantes; pour chacune d'elles, représentez l'évolution des registres du processeur ainsi que celle de la mémoire. Chaque opération reprend l'état de la mémoire centrale et des registres du processeur résultant de l'opération précédente.

```
LOAD Im R0 100
ADD Rg2 R0 R1
LOAD I R2 101
LOAD Im R1 -44
ADD D R1 100
JMPN NEG
STORE B R2 203
STOP
NEG : STORE B R1 100
STOP
```

Question 1 : La taille d'un fichier se mesure en octets. Les listes ci-dessous donnent chaque fois 4 tailles de fichiers. Laquelle de ces listes est classée par ordre croissant ?

1 : 627 octets ; 235 Ko ; 1,64 Mo ; 4,73 Go
2 : 1,64 Mo ; 4,73 Go ; 235 Ko ; 627 octets
3 : 627 octets ; 1,64 Mo ; 4, 73 Go ; 235 Ko
4 : 1,64 Mo ; 235 Ko ; 4,73 Go, 627 octets
5 : 2240 octets ; 1,52 Ko ; 1,25 Mo ; 3 Go

Question 2 : Quelle est la taille, en bits, d'un mot de deux octets ?

1 : 256 bits
2 : 8 bits
3 : 16 bits
4 : 1 bit

Question 3 : Quel(s) est(sont) le(les) périphérique(s) de sortie seulement ?

1 : Souris
2 : Imprimante
3 : Modem
4 : Scanner
5 : Disque

Question 4 : Qu'est-ce que la mémoire vive ou RAM de l'ordinateur ?

1 : La mémoire du disque dur
2 : La mémoire de la carte mère
3 : La mémoire d'une clé USB
4 : La mémoire d'une carte mémoire flash
5 : La mémoire de la carte vidéo

Question 5 : Combien de combinaisons différentes en binaire peut-on coder sur un octet ?

1 : 8
2 : 16
3 : 64
4 : 128
5 : 256

Question 7 : Combien de fichiers d'une taille de 10 Mo peut-on stocker dans 1 Go ?

1 : On ne peut pas stocker de fichier de cette taille dans un espace de 1 Go.
2 : de l'ordre de 1000
3 : de l'ordre de 10
4 : de l'ordre de 100