

# NSI - Première

Initiation assembleur

---

qkzk

2021/06/01

# Initiation à l'assembleur

---

- Les instructions machines sont exécutées par l'unité de commande.
- Un programme est une suite d'instructions.

Le CPU est incapable d'exécuter directement des programmes écrits en Python (ou C, java etc.).

Les instructions exécutées au niveau du CPU sont donc codées en binaire.

## **Langage machine :**

Ensemble des instructions exécutables par le microprocesseur

# Les instructions machines sont propres à chaque processeur

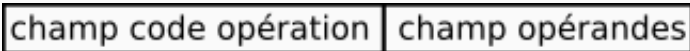
Chaque processeur possède son propre jeu d'instruction !

Mais il existe de grandes familles (x86 PC, ARM téléphones etc.)

# Précisions

Une instruction machine est une chaîne binaire composée principalement de 2 parties :

- le champ “code opération” qui indique au processeur le type de traitement à réaliser. Par exemple le code “00100110” donne l’ordre au CPU d’effectuer une multiplication.
- le champ “opérandes” indique la nature des données sur lesquelles l’opération désignée par le “code opération” doit être effectuée.



**Figure 1:** instruction machine

# Familles Instruction machine

---

# Trois grandes familles

- arithmétiques
- transfert de données
- rupture de séquence



- Les instructions *arithmétiques* (addition, soustraction, multiplication...).

“additionne la valeur contenue dans le registre R1 et le nombre 789 et range le résultat dans le registre R0”.

- Les instructions *de transfert de données*

“prendre la valeur située à l’adresse mémoire 487 et la placer dans la registre R2”

## Séquence d'instruction

- Les instructions de rupture de séquence

les instructions machines sont situées en mémoire vive Au cours de l'exécution, le CPU passe d'une instruction à une autre

- l'instruction n°1 située à l'adresse mémoire 343,
- l'instruction n°2 située à l'adresse mémoire 344 etc.

### Rupture de séquence

Rupture de séquence (ou *saut*, *branchement*): interrompre l'ordre initial sous certaines conditions en passant à une instruction située une adresse mémoire donnée

Si la valeur contenue dans le registre R1 est  $>0$ ,

alors la prochaine instruction à exécuter est l'adresse mémoire 4521

sinon, la prochaine instruction à exécuter est à l'adresse mémoire 355.

# Programme en langage machine

Suite très très longue de “1” et de “0” !

Extrêmement difficile : on remplace les codes binaires par des symboles mnémoniques.

11100010100000100001000001111101 → ADD R1,R2,\#125

## **Assembler :**

Transformer ces codes mnémoniques en langage machine.

ADD R1,R2,\#125 → 11100010100000100001000001111101

Encore aujourd’hui, programmer en assembleur est une activité courante.

# Présentation d'un langage assembleur par l'exemple

---

# Déplacements

---

Mnemonique	Exemple	Description
Charger	LDR R1, 78	Place la valeur stockée à l'adresse mémoire 78 dans le registre R1
Stocker	STR R3, 125	Place la valeur stockée dans le registre R3 en mémoire vive à l'adresse 125
Déplacer	MOV R1, #23	Place 23 dans R0

---

# Opérations arithmétiques

---

Mnemonic	Exemple	Description
Ajouter	ADD R1,R0,#128	Additionne 128 à la valeur du registre R0, place dans R1
Ajouter deux registres	ADD R0,R1,R2	Additionne R1 à R2, place dans R0
Soustraire	SUB R1,R0,#128	Soustrait 128 de R0, place dans R1
Soustraire deux registres	SUB R0,R1,R2	Soustrait R2 de R1, place dans R0

---



## Rupture de séquence

Mnemonic	Exemple	Description
Arrêter la machine	HALT	Arrête l'exécution du programme
Saut incondi- tionnel	B 45	La prochaine instruction se situe en mémoire à l'adresse 45
Comparer	CMP R0, #23	Compare R0 et le nombre 23  CMP doit précéder un branchement conditionnel
Saut "Égal"	BEQ 78	Si le dernier CMP est égal, saute à l'adresse 78
Saut "Différent"	BNE 78	Si le dernier CMP est différent, saute à l'adresse 78
Saut "Plus grand"	BGT 78	Si le dernier CMP est plus grand, saute à l'adresse 78

## Exemple

Expliquez brièvement :

```
ADD R0, R1, #42
```

```
LDR R5, 98
```

```
CMP R4, #18
```

```
BGT 77
```

```
STR R0, 15
```

```
B 100
```

## Exemple

Écrire les instructions en assembleur correspondant à :

- Additionne la valeur stockée dans le registre R0 et la valeur stockée dans le registre R1, le résultat est stocké dans le registre R5
- Place la valeur stockée à l'adresse mémoire 878 dans le registre R0
- Place le contenu du registre R0 en mémoire vive à l'adresse 124
- la prochaine instruction à exécuter se situe en mémoire vive à l'adresse 478
- Si la valeur stockée dans le registre R0 est égale 42 alors la prochaine instruction à exécuter se situe à l'adresse mémoire 85

# Labels

Les instructions assembleur B, BEQ, BNE, BGT et BLT n'utilisent pas directement l'adresse mémoire de la prochaine instruction à exécuter, mais des "labels".

Un label correspond à une adresse en mémoire vive

```
CMP R4, #18
BGT monLabel
MOV R0,#14
HALT
```

```
monLabel:
MOV R0,#18
HALT
```

```
x = 4
y = 8
if x == 10:
    y = 9
else :
    x = x+1
z = 6
```

```
MOV R0, #4      |      else:
STR R0, 30      |          LDR R0, 30
MOV R0, #8      |          ADD R0, R0, #1
STR R0, 75      |          STR R0, 30
LDR R0, 30      |      endif:
CMP R0, #10     |          MOV R0, #6
BNE else        |          STR R0, 23
MOV R0, #9      |          HALT
STR R0, 75      |
B endif         |
```