Langage d'assemblage

Arthur Garnier

February 4, 2015

1 Syntaxe

	mnémonique	0 à 3 opérandes	Facultatif
label	OP	opérande1, opérande2, opérande3	//commentaire

Exemple :

```
toto ADD R1,R2,R3 // R1+R2 -> R3; R3=a+b
```

1.1 Jump à l'aide des labels

Adresse de l'instruction en cours : $

Donc

```
JEQ #toto-$-2
```

saute vers l'instruction toto (Dans un assembleur réel on utiliserait JEQ toto)

1.2 Instructions du groupe III

```
LDW Ra,Rb
```

Va de Rb vers Ra

Alors que

```
STW Ra, Rb
```

Va de Ra vers Rb

2 Directives d'assemblage

C'est un ordre à l'assembleur, c'est à dire au traducteur

2.1 EQU

```
seuil equu 3 // Remplace seuil par 3 partout
ADQ-seuil, R0
```

Après une passe :

```
ADQ -3, R0
```

Après deux passes :

```
0011 0000 11111101
```

2.2 ORG (ORiGine)

C'est l'adresse de chargement, c'est là où sera placé le programme dans la mémoire.

```
org 0xFFD0
```

ou

```
LOAD_ADRESS equ 0xFFD0
            org LOAD_ADDRESS
```

2.3 START

Indique l'adresse de démarrage

Exemple :

```
start 0xFFD0
```

ou

```
LOAD_ADDRESS equ 0xFFD0
             start START_ADDRESS
```

Différent de org car le début du programme peut contenir des définitions de tableau, sous-programme, ... au début de celui ci.

2.4 RSW (ReServe Words)

C'est une directive qui permet de réserver des mots

```
RSW 5 // Réserve 5 mots
```

2.5 RSB (ReServe Bytes)

Permet de reserver n Bytes

2.6 STRING

Place une chaine de caractères en ASCII

```
string "Hello World !"
```

3 Expressions

Elles doivent être calculables *statiquement* (i.e. avant execution)

Exemples :

```
4*2-3
4*2*(6-4)
4*toto+titi //toto et titi définis par equ ou label ou expression
4*$+titi // $ = adresse de l'instruction dans laquelle il se trouve
x equ 4*3+2
y equ 3*x-5
```