

# Assembleur ARM – TD

---

Semestre 4



---

# Table des matières

---

<b>1</b>	<b>Opérations</b>	<b>4</b>
<b>2</b>	<b>TD1</b>	<b>5</b>
2.1	.....	5
2.1.1	.....	5
2.1.2	.....	5
2.2	.....	5
2.2.1	.....	5
2.2.2	.....	6

# Opérations

```

1 | @R3 <- R0 + (R1-R2)
2 | SUB R3,R1,R2
3 | ADD R3,R0,R3

```

Listing 1.1 –  $R3 \leftarrow R0 + (R1 - R2)$

```

1 | @R0 <- R1 + R2 + R3
2 | ADD R3,R1,R2
3 | ADD R3,R1,R3

```

Listing 1.2 –  $R0 \leftarrow R1 + R2 + R3$

```

1 | @R0 <- R1+(R2-4)
2 | SUB R0,R2,#4
3 | ADD R0,R1,R0

```


Listing 1.3 –  $R0 \leftarrow R1 + (R2 - 4)$

```

1 | @R0 <- R2 + R2*4
2 | MOV R0,#4
3 | MUL R0,R2,R0
4 | ADD R0,R2,R0

```

Listing 1.4 –  $R0 \leftarrow R2 + R2 * 4$

 MUL R3,R2,#4 n'est pas possible

```

1 | @R0 <- -R1 sous 3 formes différentes
2 | RSB R0,R1,#0
3 |
4 | MOV R2,#0
5 | SUB R0,R2,R1
6 |
7 | MVN R0,R1
8 | ADD R0,R0,#1

```

Listing 1.5 –  $R0 \leftarrow -R1$  sous 3 formes différentes

```

1 | @En un minimm d'instructions calculez r0 <- 10 * r1
2 | @(sans utiliser la multiplication)
3 | MOV r0,r1,LSL #3 @r0 = r1 * 2^3
4 | @r0 <- r0 + r1 * 2^1
5 | @r0 <- r1 * 2^3 + r1 * 2^1
6 | @r1 * (2^3 + 2^1)
7 | ADD r0,r0,r1,LSL #1

```

Listing 1.6 –  $R0 \leftarrow 10 * R1$  sans utiliser la multiplication

---

# TD1

---

## 2.1

### 2.1.1

ON considère l'algorithme suivant :

```
1 | si r0 > 0 alors
2 |   s1;
3 | sinon
4 |   s2;
5 | fin si;
```

Traduire cette forme algorithmique en assembleur.

```
1 | CMP r0,#0 @si r0 > 0 alors
2 | BLS sinon
3 |   <s1>
4 |   B finsi
5 | sinon:  @ sinon
6 |   <s2>
7 | finsi:
```

### 2.1.2

```
1 | tantque 0 > 0 faire
2 |   s;
3 | fin tantque;

1 | boucle: CMP r0,#0
2 |   BLE finboucle
3 |   <S>
4 |   B boucle
5 | finboucle:
```

## 2.2

### 2.2.1

```
1 | r0 <- 0;
2 | tantque R0 < N faire
3 |   r1 <- r1 + r0;
```

```
4 | r0 <- r0 + 1;
5 | fin tantque;

1 | .equ N,10
2 | MOV R0,#0 @s <- 0
3 | MOV R1,#0 @i <- 0
4 | tq: CMP R1,#N @tantque(i <= N)
5 | BHI ftq
6 | ADD R1,R0,R1 @s += i
7 | ADD R1,R1,#1 @ i++
8 | B tq
9 | ftq:
```

## 2.2.2

Écrire un programme qui calcule la multiplication de r0 par r1 et range le résultat dans r2 sans l'opération de multiplication

```
1 | MOV r3, #0 @ i <- 0
2 | MOV r2, #0 @ r2 <- 0
3 | tq: CMP r3,r1 @ tantque(i < r1)
4 | BHS ftq
5 | ADD r2,r0,r2 @r2 <- r0 + r2
6 | ADD r3, r3, #1 @i++
7 | B tq
8 | ftq:
```