

Assembleur ARM – TD

Semestre 4

Table des matières

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 multipliatiion

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

2.2.2.1 Nombres non signés

```
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:
```

2.2.2.2 Nombres signés

```
1 | si r1 > 0 alors
2 |   multiplication non signé;
3 | sinon
4 |   r1 <- 0 - r1;
5 |   r3 <- 0;
6 | fin si;
7 |
8 | si r3 = 0 alors
9 |   r1 <- 0 - r1;
10 | sinon
11 |   on sort;
```

```
1 | MOV r3, #1
2 | CMP r1, #0
3 | BGT boucle
4 | RSB r1, r1, #0
5 | MOV r3, #0
6 | boucle: MOV R2, #0
7 | tq: CMP R1, #0
8 | BEQ sortie1
9 | ADD r2, r0, r0
10 | SUB r1, r1, #1
11 | B tq
12 | sortie1: CMP R3, #0
13 | BNE sortie2
```

```

14 | RSB r1, r1, #0
15 | sortie2:

```

2.2.2.3 Autres solutions pour nombres signés

```

1 | si r1 < 0 alors
2 |   r1 <- -r1;
3 |   r0 <- -r0;
4 | fin si;
5 | i <- 0;
6 | tantque i < r1 faire
7 |   r2 <- r2 + r0;
8 |   i <- i + 1;
9 | fin tantque

```

```

1 | CMP r1, #0
2 | RSBLT r1, r1, #0
3 | MOV r2, #0
4 | MOV r3, #0
5 | tq: CMP r3, r1
6 |   BGE ftq
7 |   ADD r2, r2, r0
8 |   ADD r3, r3, #1
9 |   B tq
10 | ftq:

```

2.2.3 Réaliser la division de r0 par r1 qui donne dans r2 le quotient et dans r3 le reste

2.2.3.1 Non signé

$$r0 = r1 \times r2 + r3, 0 \leq r3 < r1$$

```

r2 <- 0;
tantque r3 >= r1 faire
  r1 <- r2 + 1;
  r3 <- r3 - r1;
fin tantque;

```

```

MOV r2,#0
MOV r3,r0
tq: CMP r3,r1
  BLT ftq
  ADD r2,r2,#1
  SUB r3,r3,r1
  B tq
ftq:

```

2.2.3.2 Signé

```

1 | si r0 > 0 alors
2 |   r4 <- 0;
3 | sinon
4 |   r4 <- 1;
5 |   r0 <- -r0;

```

```
6  fin si;
7
8  si r1 > 0 alors
9      r5 <- 0;
10 sinon
11     r5 <- 1;
12     r1 <- -r1;
13     r2 <- 0;
14     r3 <- 0;
15 fin si;
16 tantque r3 > r1 faire
17     r3 <- r3 - r1;
18     r2 <- r2 + 1;
19 fin tantque;
20
21 si r4 = -r3 alors
22     r3 = -r3;
23 fin si;
24 si r4 != r5 alors
25     r2 = -r2;
26 fin si;
```

```
1  CMP r0, #0
2  BLT sinon1
3  MOV r4,#0
4  RSB r0,#0
5  B fsi1
6  sinon1: MOV r4,#1
7  fsi1:
8      CMP r1, #0
9      BLT sinon2
10     MOV r5,#0
11     RSB r2,#0
12  sinon2:
13  @(boucle)
14     CMP r4,#1
15     BNE fsi
16     RSB r3,#0
17  fsi3:
18     CMP r4,r5
19     BEQ fsi4
20     RSB r2,#0
21  fsi4:
```


TD2 – Manipulation de la mémoire

3.1

3.1.0.3 Avec un itérateur

```

1 | define N 10
2 | t[10];
3 | t1 <- @t;
4 | r2 <- 0; --i
5 | r0 <- 0;
6 |
7 | tantque t2 < N faire
8 |     r3 <- MEM[t1+r2 << 2];
9 |     r2 <- r2+1;
10 |    r0 <- r0 + r3
11 | fin tantque;

```

```

1 | .eq N,10
2 | t: .fill N,4,0
3 | LDR r1,=t
4 | MOV R2,#0
5 | MOV R0,#0
6 |
7 | tq: CMP r1,N
8 |     BHS ftq
9 |     LDR r3,[r1,r2,LSL#2]
10 |    ADD r0,r0,r3
11 |    ADD r2,r2,#1
12 |    B tq
13 | ftq:

```

3.1.1 Avec un pointeur

```

1 | r1 <- @t;
2 | r2 <- @tfin;
3 | r0 <- 0;
4 |
5 | tantque r1 != r2 faire
6 |     r3 <- MEM32[r1];
7 |     r1 <- r1+4;
8 |     r0 <- r0 + r3;
9 | fin tantque

```

```

1 | LDR r1,=t
2 | LDR r2,=tfin @ <=> ADD r2,r1,#N*4
3 | MOV r0,#0
4 | tq: CMP r1,r2
5 |     BEQ ftq
6 |     LDR r3,[r1] @on pourrait faire
7 |     ADD r1,r1,#4 @LDR r3,[r1],#4
8 |     ADD r0,r0,r3
9 |     B tq
10 | ftq:

```

3.2

```
1 |
2 | t[N]
3 | r1 <- @t;
4 | r2 <- @tfin;
5 | r0 <- 1;
6 | tantque r1 != r2 faire
7 |     MEM32[r1] <- r0;
8 |     r1 <- r1 + 1;
9 |     r0 <- r0 + ;
10 | fin tantque;
```

```
1 | .eq N,10
2 | t: .fill N,1,0
3 | LDR r0,=t
4 | LDR r2,=tfin
5 | MOV r0,#1
6 | tq: CMP r1,r2
7 | BEQ ftq
8 | STRB r3,[r1],#1
9 | ADD r0,r0,#1
10 | B tq
11 | ftq:
```

3.3 Que fais le programme suivant ?

```
1 | LDR r0,=t
2 | MOV f1,#111
3 | STR r1,[r0]
4 | ADD r1,r0,#4
5 | MOV r2,#0
6 | tq: CMP r2,#4
7 | BEQ ftq
8 | LDR r3,[r0,r2,LSL#2]
9 | STR r3,[r1],#4
10 | ADD r2,r2,#1
11 | B tq
12 | ftq:
```

```

1
2
3
4 r0 <- @t;
5 r1 <- @fins;
6 i <- 0;
7 tantque i < N faire
8     t[N-1-i] = s[i];
9     i <- i + 1;
10 fin tantque;

```

```

1 .eq N, 100
2 s: .byte 1,23,255 @,...
3 t: .fill N, 1, 0
4     LDR r0,=t
5     LDRB r1,=s
6     MOV r2, #N
7 tq: CMP r2, #N
8     BHS ftq:
9     RSB r4, R0, #N-1
10    LDRB r3, [r1,r2]
11    STRB r3,[r0,r4]
12    ADD r2, r2, #1
13    B tq:
14 ftq:
15
16 @ Avec pointeurs
17 ADR r1, s
18 ADR r2, t
19 ADD r3, r1, #N-1
20 tq: CMP r3, r1
21     BHS ftq
22     LDR r4, [r2], #1
23     STR r4, [r3], #-1
24
25     B tq
26 ftq:

```

[rj]
[rj,rk]
[rj,rk,LSL#]
[rj],#k
[rj],#k]

3.4 Chevauchement de tableaux

```

1 r0 <- @t1
2 r1 <- @t2
3
4 si r0 < r1 alors
5     r2 <- N;
6     tantque r2 != 0 faire
7         r2 = r2 - 1;
8         r0[r2] <- r1[r2];
9     fin tantque;
10 sinon
11     tantque r2 != N faire
12         r2 <- 0;
13         r0[r2] <- r1[r2];
14         r2 = r2 + 1;
15     fin tantque;
16 fin si;

```

```

1 .equ N, 10
2 t1: .fill N,
3 t2: .fill N,
4
5     ADR r0, t1
6     ADR r1, t2
7 si:  CMP r0,r1
8     BHS else
9     MOV r2, #N
10 tq1: CMP r0,#0
11     BEQ ftq1
12     SUB r2,r2,#1
13     LDR r3, [r0,r2,LSL#2]
14     STR r3, [r1,r2,LSL#2]
15     B tq1
16     B fsi
17 else: MOV R2, #0
18 tq2:  CMP r2,#N
19     BEQ ftq2
20     ADD r2,r2,#1
21     LDR r3[r0,r2,LSL#2]
22     STR r3[r1,r2,LSL#2]
23     B tq2
24 fsi:

```

-- Autre solution

```

27 si t1 < t2 alors

```

```

28     i <- N-1;

```

```

29     f <- -1;

```

```

30     p <- -1;
31     Assembleur ARM - TD — archi4

```

@@ Autre solution

```

27     MOV r3,#4

```

```

28     ADR r0,r1

```

```

29     ADR r1,r2

```

```

30     ADR r2,r2+N*4

```

```

31     CMP r1,r0

```

```

32     BHS NO_CHANCE

```

3.5 Tris à bulles

```

1 N <- 50;
2 t[N];
3 i <- 0;
4 size <- N;
5 tmp;
6
7 tantque size > 1 faire
8   i <- 0;
9   tantque i < (size-1) faire
10    tmp <- t[i];
11    t[i] <- t[i+1];
12    t[i+1] <- tmp;
13    i <- i + 1;
14   faire
15   size <- size - 1;
16 fin tantque;
17
18
19
20
21
22
23
24
25
26
27
28
29
30 -- Autre algorithme
31 r0 <- t;
32 r1 <- t + 4N - 4;
33 tantque r0 = r1 faire
34   r2 <- r0;
35   tantque r2 < r1 faire
36     si Mem32[2) < MEM32[r3+r4] alors
37       t <- Mem32[r2];
38       Mem32[r2] <- Mem32[r2+4];
39       Mem32[r2+4] <- t;
40     fin si;
41     r2 <- r2 + 4;
42   fin tantque;
43   r1 <- r1 + 4;
44 fin tantque

```

```

1 .equ N,50
2 t: .word 1,5,...
3 MOV r1,#0
4 MOV r2,#N
5 ADR r0,t
6
7 tq1: CMP r2,#N
8 BLS ftq1
9 MOV r1,#0
10 tq2: SUB r3,r2,#1
11 CMP r1,r3
12 BHS ftq2
13 LDR r3,[r4,r1,LSL #2]
14 ADD r5,r1,#1
15 LDR r4,[r0,r5,LSL #2]
16 si1: CMP r3,r4
17 BHS fin1
18 STR r4,[r0,r1,LSL #2]
19 STR r3,[r0,r5,LSL #2]
20 fin1: ADD r1,r1,#1
21 B tq2
22 ftq2: SUB r2,r2,#1
23 ftq1: B tq1
24
25
26
27
28
29
30 @ Autre algorithme
31 si1: CMP r3,r4
32 BHS fsi
33 STR r3,[r2]
34 STR r4,[r2,#-4]
35 fsi1: B tq2
36 ftq2: SUB r1,r1,#4
37 B tq1
38 ftq1:

```

Sous programmes

4.1 rechmax

```

1  r0 <- @ft - 4;
2  r1 = @t;
3
4  tantque r0 < r1 faire
5      rechMax
6      Mem32[r2] = Mem32[r1]
7      Mem32[r1] = r2
8      r2 += 4
9  fin tantque;
10
11 -- Rechmax
12 r2 = MEM32[r0]
13 r3 = r0;
14
15 tantque r0 < r1 faire
16     r4 = MEM32[r0];
17     si r2 < r4 alors
18         r2 = r4;
19         r3 = r0;
20     fin si;
21     r0 += 4;
22 fin tantque;

```

```

1  _start:
2      ADR r0, t
3      ADR r1, ft
4      SUB r1, r1, #4
5  tq1:   CMP r0, r1
6      BHS ftq
7      BL  rechMax
8      LDR r3, [r1]
9      STR r3, [r3]
10     STR r2, [r1]
11     SUB r1, r1, #4
12     B   tq1
13 ftq1:
14
15
16 rechMax:
17     STMFD sp!, {r0, r4}
18     LDR  r2, [r0]
19     MOV  r3, r0
20
21 tq:   CMP r0, r1
22     BHS ftq
23     LDR  r4, [r0]
24
25 si1:  CMP r2, r4
26     BHS fsi
27     MOV  r2, r4
28     MOV  r3, r0
29 fsi:
30     ADD  r0, r0, #4
31     B    tq
32 ftq:
33     LDMFD sp!, {r0, r4}
34     MOV  pc, r14
35
36 t:    .int 3, 4, 9, ...
37 ft:

```

4.2 atoi

```
1  -- r0 adresse tableau
2  -- r1 valeur
3  fonction atoi(entree r0,
4               sortie r1)
5  debut
6      s <- 0;
7      tantque *r0 != '\0' faire
8          s <- s * 10;
9          s = s + (*r0 - '0');
10         r0 <- r0 + 1;
11     fin tantque;
12 fin
```

```
1  _atoi:    STMFD sp!{r0,r2,r3,r4}
2             MOV     r1, #0
3             MOV     r4, #10
4
5  tq:        LDRB     r3,[r0]
6             CMP     r3, #'\'0\'
7             BEQ     ftq
8             MOV     r2,r3
9             MUL     r1,r4,r1
10            ADD     r0,r0,#1
11            SUB     r2,r2,#0
12            B       tq
13  ftq:       LDMFD   sp!{r0,r2,r3,r4}
14
15
16  @ Optimisation
17  _atoi:    STMFD   sp!{r0,r2}
18             MOV     r1, #0
19
20  tq:        LDRB     r2,[r0]
21            CMP     r2, #'\'0\'
22            BEQ     ftq
23            ADD     r1,r1,r1
24            ADD     r1,r2,r2,LSL#2
25            SUB     r2,r2,#0
26            B       tq
27  ftq:       LDMFD   sp!{r0,r2}
```

4.3 itoa

```

1  -- Utiliser le sosu programme de      1  @ transformation d'un entier en
   division du TD6                      2  @ chaine de caratères
2  -- r0 : valeur entière                3  @ r0 (entree) entier
3  -- r1 : adresse du taleau ou stocker 4  @ r1 (entree) chaine
   la cha ne de caractère              5
4  fonction itoa(entree r0, entree r1) 6  itoa:  STMFD sp!{ LR}
5  debut                                7      MOV    r4,r1
6                                          8      MOV    r5,r1
7  fin                                  9      MOV    r1, #10
                                       10
                                       11  @ boucle de décomposition
                                       12  tq1:    CMP    r0, #0
                                       13          BEQ    ftq1
                                       14          BL     division
                                       15          MOV    r0, r2
                                       16          ADD    r3, r3, #'0'
                                       17          STRB   r3, r5, #1
                                       18          B      tq1
                                       19  ftq1:
                                       20          MOV    r3, #'\\0'
                                       21          STRB   r3, [r5]
                                       22
                                       23  @ Boucle inersion
                                       24  tq2:    CMP    r4,r5
                                       25          BHS    fq2
                                       26          LDRB   r2, [r4]
                                       27          LDRB   r3, [r5, #-1]!
                                       28          STRB   r2,[r5]
                                       29          STRB   r3, [r4], #1
                                       30          B      ftq2
                                       31  ftq2:

```