Eléments de correction

1. Rupture de séquence
* Rupture inconditionnelle (b etiquette, pour la répétition d'une boucle)
* Rupture conditionnelle (beq etiquette, bgt etiquette, [etc.] pour la sortie d'une boucle)
* Appel de fonction (bl fonction, pour le passage des paramètres, choisir/appliquer une convention commune à l'appelant/appelé)
* Retour de fonction (bx lr)

2. Course à 100
d) Affichage

```
  LDR R1, LD_MsgTab
  BL EcrChaine

  LDR R1, LD_N
  LDR R1, [R1]
  BL EcrZdecimal32
```

...

e) Saisie

```
  LDR R0, LD_Tab
  MOV R1, #100
  BL LireTableauNombres
  LDR R1, LD_N
  STR R0, [R1]
```

f) Appel

```
  LDR R0, LD_Tab
  PUSH {R0}
  MOV R0, #0
  PUSH {R0}
  LDR R0, LD_N
  LDR R0, [R0]
  PUSH {R0}
  MOV R0, #100
  PUSH {R0}
  SUB SP, SP, #4
  BL course
  POP {R0}
  ADD SP, SP, #12
  CMP R0, #0
  BNE finsi
  LDR R1, LD_MsgEch
  BL EcrChaine
finsi:
```

...

h) Pile

```
 R1
 RO
 ko
 ok
 res
 suivant
 valeur
*FP* <<== FP
 LR
 Resultat
```

```
 somme/100
 taille/N
 i/0
 TE/Tab
```

i) Solution

```
  LDR R0, [FP, #12]
  LDR R0, [R0]
  CMP R0, #0
  BNE sinon
  LDR R1, LD_MsgSol
  BL EcrChaine
  LDR R0, [FP, #-16]
  STR R0, [FP, #-12]
  b finsisinon
sinon:
...
finsisinon:
...
```

j) Acces tableau

```
  LDR R0, [FP, #20]
  LDR R1, [FP, #24]
  LDR R0, [R1, R0, lsl #2]
  STR R0, [FP, #-4]
```

k) Appel recursif

```
  LDR R0, [FP, #24]
  PUSH {R0}
  LDR R0, [FP, #-8]
  PUSH {R0}
  LDR R0, [FP, #16]
  PUSH {R0}
  LDR R0, [FP, #12]
  LDR R1, [FP, #-4]
  SUB R0, R0, R1
  PUSH {R0}
  SUB SP, SP, #4
  BL course
  POP {R0}
  ADD SP, SP, #12
  CMP R0, #0
  BNE sisinon
  LDR R1, [FP,#-4]
  BL EcrZdecimal32
  LDR R0, [FP, #-16]
  STR R0, [FP, #-12]
  b finsisisinon
sisinon:
...
finsisisinon:
...
```

l) Prologue, L.18, épilogue

```
Proloque:
PUSH {LR}
PUSH {FP}
MOV FP, SP
SUB SP, SP, #20
```

```
PUSH {R0}
PUSH {R1}
MOV R0, #0
STR R0, [FP, #-20]
MOV R0, #1
STR R0, [FP, #-16]

... corps de la fonction

L.18 :
LDR R0, [FP, #-12]
STR R0, [FP, #8]

Epilogue:
PUSH {R1}
PUSH {R0}
ADD SP, SP, #20
PUSH {FP}
PUSH {LR}
BX LR
```

3. Microprocesseur

m) Mémoire

@	Valeur
0	2
1	6
2	5
3	E
4	5
5	F
6	3
7	E
8	1
9	4
A	2
B	-
C	-
D	-
E	4
F	E

n) Exécution, ASM

ACC : 6, A, 8, 8, 0, 8, 6, 6, 0, 6, 4, 4, 0, 4, 2

o) Exécution automate

```
ma <= pc (0)
md <= MEM[ma] (2)
ri <= md (2)
pc <= pc+1 (1)
ma <= pc (1)
pc <= pc+1 (2)
md <= MEM[ma] (6)
acc <= md (6)
---
ma <= pc (2)
md <= MEM[ma] (5)
ri <= md (5)
pc <= pc+1 (3)
ma <= pc (3)
pc <= pc+1 (4)
```

```
md <= MEM[ma] (E)
ma <= md (E)
md <= MEM[ma] (4)
acc<=ac+md (A)
---
...
```

p) dup

```
en alternative à acc<=acc+md :
ma<=ma+1
MEM[ma]<=md
```

q) swp

```
en alternative à acc<=acc+md :
tmp <= md
ma<=ma+1
md <= MEM[ma]
ma<=ma-1
MEM[ma]<=md
md<=tmp
et on continue avec l'alternative de dup (c. prec)

fin.
```

Moyenne des copies : 9.11
Min : 1 ; Max : 19
Courbe à 2 bosses (7 et 14).

A noter :
* a, b, c) ~1/3 des copies ne répond pas à la question de cours
* 2) les traductions doivent être systématiques et complètes :
** ne pas "supposer que R0=..., R1=..."
** ne pas oublier de traduire l'ensemble des lignes demandées
* e) le passage de paramètre prévu est "par registre", ne pas utiliser la pile
* f) attention à inverser correctement l'avant/après appel : push {R0}, push {R1}, push {R2}, push {R3}, SUB SP, SP, #4 ne s'inverse pas en LDR R4, [SP], pop {R3}, pop {R2}, pop {R1}, pop {R0}, ADD SP, SP, #4 et il faut éviter de récupérer le résultat dans R0 si récupérer la valeur initiale de R0 dans la suite immédiate
* i) sur la pile, ce sont les valeurs, il n'y a pas d'indirection à faire
* i, k) en fin de traduction du "alors" ne pas oublier "b finsi"
* j) l'adresse de l'élément de tableau est TE+4*i (ne pas oublier le *4)
* l) l'initialisation de ok/ko est à faire au bon moment (trop tôt, cela entre en conflit avec la sauvegarde des temporaires), push {#0} n'est pas autorisé
* m) l'état initial de la mémoire (MEM/RAM), ce n'est pas l'état initial de pc/acc/ri/...
* n) ne pas s’arrêter à la fin de la première boucle