

Exercices 3.1

1)

Step	R0	R1	R2	R3	N	Z	C	V	PC
0	0x60	0x67	0x124	0x0	F	F	F	F	0x168
1	0x80	0x67	0x19	0x0	F	F	T	F	0x16a
2	0x80	0x67	0x19	0x0	F	F	T	F	0x170
3	0x19	0x67	0x19	0x0	F	F	T	F	0x172
4	0x19	0x67	0x19	0x0	F	F	T	F	0x1ee
5	0x8	0x18	0x19	0x0	F	F	T	F	0x168
6	0x8	0x18	0x11111110	0x0	T	F	F	F	0x16a
7	0x8	0x18	0x11111110	0x0	T	F	F	F	0x16c
8	0x8	0x18	0x10	0x0	T	F	F	F	0x170
9	0x10	0x18	0x10	0x0	T	F	F	F	0x172

2) En ce qui concerne le fanion **C**, il est levé suite à la soustraction binaire entre **R₀** et **R₁**, qui contiennent respectivement les valeurs **128** et **103**. Cela se produit car la soustraction binaire est en réalité une addition entre la partie gauche de l'opération et le complément à 1 (inversion des bits) de la partie de droite. Puis on rajoute 1 à cette somme et on obtient le résultat.

$$(128)_{10} = (1000\ 0000)_2$$

$$(103)_{10} = (0110\ 0111)_2$$

Complément à 1 de 103:

$$(0110\ 0111)_2$$

$$(1001\ 1000)_2 \Rightarrow (152)_{10}$$

$$128 + 152 + 1$$

$$= (281)_{10}$$

$$\hookrightarrow (0001\ 0001\ 1001)_2$$

↳ Finalement, on tronque la partie en dehors des 8 premiers bits car les registres sont sur 8 bits

$$\text{Resultat: } (0001\ 1001)_2$$

$$= (25)_{10}$$

Après cette opération, on assignera la valeur de R_2 (résultat de la soustraction) à R_0 qui correspondra au retour de la fonction.

En ce qui concerne le fonction **N**, il est levé dès que le résultat de la soustraction est négatif. Cela se cause que la seconde soustraction sera exécutée où les opérandes seront intervertis à fin d'obtenir un résultat positif, puis satisfaire le branchement conditionnel **bhi** (**C** → levé; **Z** → pas levé) qui causera **↑ branch higher** l'assignation à R_0 , le résultat de la seconde soustraction contenu dans R_2 .

3) La valeur absolue d'une soustraction

Exercice 3.3: (Suite de Fibonacci)

```
.thumb_func
funcex03:
    // insert your code here
    // r0 => n
    // r1 => a
    // r2 => b
    // r3 => i

    mov r1, #2
    mov r2, #1
    mov r3, #0

    cmp r3, r0
    // branch -> lower, unsigned
    bcc funcex03_for_loop
funcex03_for_loop:
    add r1, r2
    sub r2, r1, r2
    add r3, #1
    cmp r3, r0
    bcc funcex03_for_loop

    mov r0, r1
    mov pc, r14    /* end of subroutine */
```