5.1

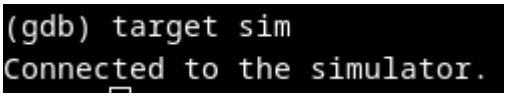
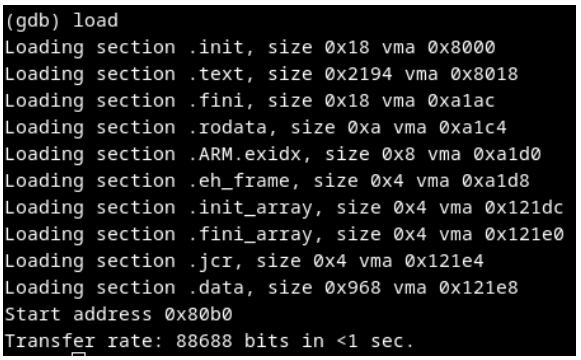
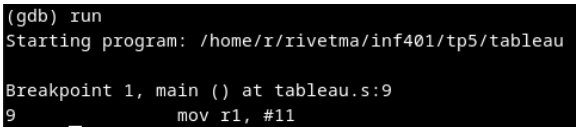
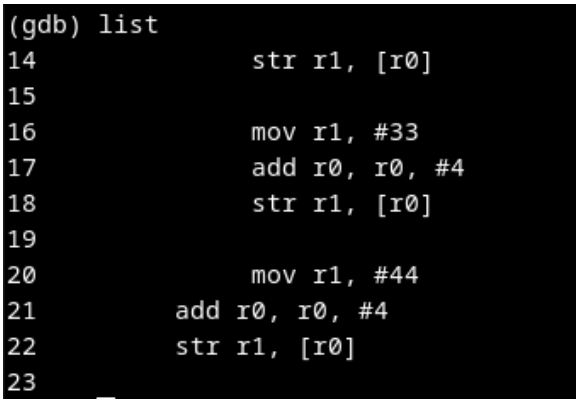
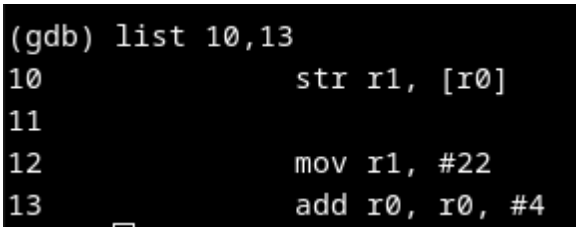
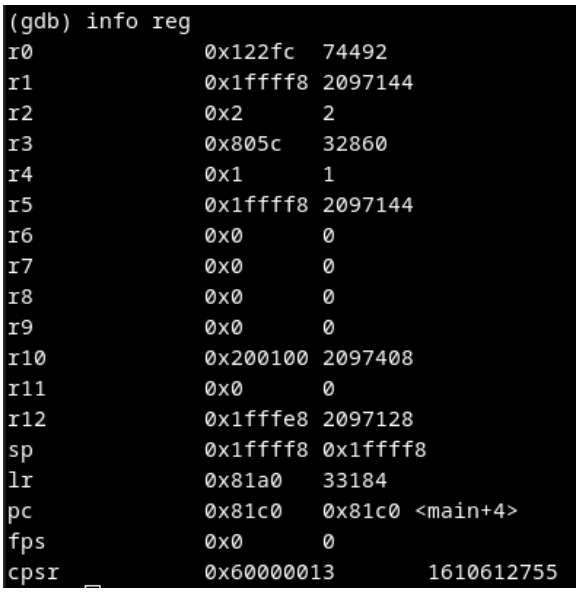
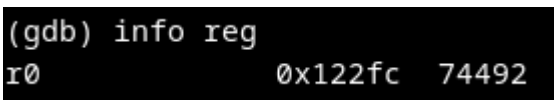
image avec la commande 1	
image avec la commande 2	
image avec la commande 3	
image avec la commande 5	
image avec la commande 6	
image avec la commande 7	

image avec la commande 8	
image avec la commande 9	
image avec la commande 10	
image avec la commande 11	

(gdb) x/5w &debutTAB 0x122fc <debutTAB>: 11 22 0 0 0x1230c <debutTAB+16>: 0 (gdb) x/5w &debutTAB 0x122fc <debutTAB>: 11 0 0 0 0x1230c <debutTAB+16>: 0	(gdb) info reg r0 0x12300 74496 (gdb) info reg r0 0x12304 74500
(gdb) x/5w &debutTAB 0x122fc <debutTAB>: 11 22 33 0 0x1230c <debutTAB+16>: 0	(gdb) info reg r0 0x12308 74504
(gdb) x/5w &debutTAB 0x122fc <debutTAB>: 11 22 33 44 0x1230c <debutTAB+16>: 0	(gdb) info reg r0 0x1230c 74508
(gdb) x/5w &debutTAB 0x122fc <debutTAB>: 11 22 33 44 0x1230c <debutTAB+16>: 55	La valeur stockée dans R0 est l'adresse du début du tableau. A chaque fois, on lui rajoute 4 et comme les entiers sont sur 32 bits, on rajoute 4 octets pour le prochain nombre;

5.2

```
(gdb) info reg
r0             0x122ec  74476
r1             0x1ffff8 2097144
r2             0x0      0
```

2,3,4°) Les valeurs de R0 à chaque itération sont les suivantes: 74476 jusqu'à 74492 de 4 en 4.
Les valeurs augmentent de 4*i pour i allant de 0 à 4 puisque le code n'est pas exécuté pour i=5.
R2 est le compteur et permet à la boucle de s'arrêter quand il vaut 5

5°)

```
    .data
debutTAB: .skip 5*4

    .text
    .global main
main:
    ldr r0, ptr_debutTAB

    mov r3, #11               @ val ← 11
    mov r2, #0                @ i ← 0
tq: cmp r2, #4                @ i-4 ??
        bgt fintq
            @ i-4 ⩽ 0
            ldr r0, ptr_debutTAB    @ r0 ← debutTAB
            add r0, r0, r2, LSL #2  @ r0 ← r0 + r2*4 = debutTAB + i*4
            str r3, [r0]            @ MEM[debutTAB+i*4] ← val
            add r2, r2, #1          @ i ← i + 1
            add r3, r3, #11         @ val ← val + 11
        b tq
fintq:  @ i-4 > 0

fin:    BX LR

ptr_debutTAB: .word debutTAB
```

Nous avons les mêmes valeurs donc les conditions sont équivalentes.*

6°)

```
    .data
debutTAB: .skip 5*2

    .text
    .global main
main:
    ldr r0, ptr_debutTAB

    mov r3, #11               @ val ← 11
    mov r2, #0                @ i ← 0
tq: cmp r2, #4                @ i-4 ??
        bgt fintq
            @ i-4 ⩽ 0
            ldr r0, ptr_debutTAB    @ r0 ← debutTAB
            add r0, r0, r2, LSL #1  @ r0 ← r0 + r2*2 = debutTAB + i*2
            strh r3, [r0]            @ MEM[debutTAB+i*2] ← val
            add r2, r2, #1          @ i ← i + 1
            add r3, r3, #11         @ val ← val + 11
        b tq
fintq:  @ i-4 > 0

fin:    BX LR

ptr_debutTAB: .word debutTAB
```

```
(gdb) info reg
r0             0x122f4  74484
r1             0x1ffff8 2097144
r2             0x5      5
```

Les valeurs de R0 à chaque itération sont les suivantes: 74476 jusqu'à 74484 de 2 en 2.

7°)

```
    .data
debutTAB: .skip 5

    .text
    .global main
main:
    ldr r0, ptr_debutTAB

    mov r3, #11                 @ val ← 11
    mov r2, #0                  @ i ← 0
tq: cmp r2, #4                  @ i-4 ??
        bgt fintq
            @ i-4 ⩽ 0
            ldr r0, ptr_debutTAB    @ r0 ← debutTAB
            add r0, r0, r2          @ r0 ← r0 + r2 = debutTAB + i
            strb r3, [r0]           @ MEM[debutTAB+i] ← val
            add r2, r2, #1          @ i ← i + 1
            add r3, r3, #11         @ val ← val + 11
        b tq
fintq:  @ i-4 > 0

fin:    BX LR

ptr_debutTAB: .word debutTAB
```

```
(gdb) info reg
r0             0x122f0  74480
r1             0x1ffff8 2097144
r2             0x5      5
```

Les valeurs de R0 à chaque itération sont les suivantes: 74476 jusqu'à 74480 de 1 en 1.

5.3

15 - 46 - 23 - 70 - 35 - 106 - 53 - 160 - 80 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1

```
    .text
    .global main
main:
    mov r1, #15             @ x ← 15
tq: cmp r1, #1              @ x - 1 ??
    beq fintq
    @ x - 1 <> 1

si: tst r1, #1              @ x pair ??
    bne sinon
alors:
    mov r1, r1, LSR #1      @ x ← x/2
    b finsi
sinon:
    add r1, r1, r1, LSL #1  @ x ← x + x*2
    add r1, #1              @ x ← x + 1
finsi:

    b tq
fintq:                      @ x - 1 == 0

fin:    BX LR
```