

5.1

image avec la commande 1	<pre>rivetma@im2ag-turing-01:~/inf401/tp5: arm-eabi-gdb tableau GNU gdb (GDB) 7.2 Copyright (C) 2010 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html> This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Type "show copying" and "show warranty" for details. This GDB was configured as "--host=x86_64-unknown-linux-gnu --target=arm-eabi" . For bug reporting instructions, please see: <http://www.gnu.org/software/gdb/bugs/>... Reading symbols from /home/r/rivetma/inf401/tp5/tableau...done.</pre>
image avec la commande 2	<pre>(gdb) target sim Connected to the simulator.</pre>
image avec la commande 3	<pre>(gdb) load Loading section .init, size 0x18 vma 0x8000 Loading section .text, size 0x2194 vma 0x8018 Loading section .fini, size 0x18 vma 0xa1ac Loading section .rodata, size 0xa vma 0xa1c4 Loading section .ARM.exidx, size 0x8 vma 0xa1d0 Loading section .eh_frame, size 0x4 vma 0xa1d8 Loading section .init_array, size 0x4 vma 0x121dc Loading section .fini_array, size 0x4 vma 0x121e0 Loading section .jcr, size 0x4 vma 0x121e4 Loading section .data, size 0x968 vma 0x121e8 Start address 0x80b0 Transfer rate: 88688 bits in <1 sec.</pre>
image avec la commande 5	<pre>(gdb) break main Breakpoint 1 at 0x81c0: file tableau.s, line 9.</pre>
image avec la commande 6	<pre>(gdb) run Starting program: /home/r/rivetma/inf401/tp5/tableau Breakpoint 1, main () at tableau.s:9 9 mov r1, #11</pre>
image avec la commande 7	<pre>(gdb) list 4 5 .text 6 .global main 7 main: 8 ldr r0, ptr_debutTAB 9 mov r1, #11 10 str r1, [r0] 11 12 mov r1, #22 13 add r0, r0, #4</pre>

image avec la commande 8	<pre>(gdb) list 14 str r1, [r0] 15 16 mov r1, #33 17 add r0, r0, #4 18 str r1, [r0] 19 20 mov r1, #44 21 add r0, r0, #4 22 str r1, [r0] 23</pre>
image avec la commande 9	<pre>(gdb) list 10,13 10 str r1, [r0] 11 12 mov r1, #22 13 add r0, r0, #4</pre>
image avec la commande 10	<pre>(gdb) info reg r0 0x122fc 74492 r1 0x1ffff8 2097144 r2 0x2 2 r3 0x805c 32860 r4 0x1 1 r5 0x1ffff8 2097144 r6 0x0 0 r7 0x0 0 r8 0x0 0 r9 0x0 0 r10 0x200100 2097408 r11 0x0 0 r12 0x1ffff8 2097128 sp 0x1ffff8 0x1ffff8 lr 0x81a0 33184 pc 0x81c0 0x81c0 <main+4> fps 0x0 0 cpsr 0x60000013 1610612755</pre>
image avec la commande 11	<pre>(gdb) s 10 str r1, [r0]</pre>

<pre>(gdb) x/5w &debutTAB 0x122fc <debutTAB>: 0 0 0 0 0x1230c <debutTAB+16>: 0</pre>	<pre>(gdb) info reg r0 0x122fc 74492</pre>
--	--

<pre>(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</pre>	<pre>(gdb) info reg r0 0x12300 74496 (gdb) info reg r0 0x12304 74500</pre>
<pre>(gdb) x/5w &debutTAB 0x122fc <debutTAB>: 11 22 33 0 0x1230c <debutTAB+16>: 0</pre>	<pre>(gdb) info reg r0 0x12308 74504</pre>
<pre>(gdb) x/5w &debutTAB 0x122fc <debutTAB>: 11 22 33 44 0x1230c <debutTAB+16>: 0</pre>	<pre>(gdb) info reg r0 0x1230c 74508</pre>
<pre>(gdb) x/5w &debutTAB 0x122fc <debutTAB>: 11 22 33 44 0x1230c <debutTAB+16>: 55</pre>	<p>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;</p>

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
```