

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DISCIPLINA: LABORATÓRIO DE PROCESSADORES- PCS3732
1º QUADRIMESTRE/2021



Aula 02
13 de Maio de 2021

GRUPO 10

Arthur Pires da Fonseca
Bruno José Móvio
Iago Soriano Roque Monteiro

NUSP: 10773096
NUSP: 10336852
NUSP: 8572921

Sumário

Exercício 2.4.1 - Compiling, making, debugging, and running	3
Exercício 2.4.2 - Stepping and stepping in	3
Exercício 2.4.3 - Data formats	4
Exercício 3.10.1 - Signed and unsigned addition	5
Exercício 3.10.2 - Multiplication / Multiplicação de número	8
Exercício 3.10.3 - Multiplication shortcuts / Multiplicação pelo número 3	9
Apêndice	14
Exercício 3.10.1	14
1ª Soma	14
2ª Soma	14
3ª Soma	15
Exercício 3.10.2	15
Exercício 3.10.3	16
Exercício 3.10.4	16

Exercício 2.4.1 - Compiling, making, debugging, and running

Resultado obtido pelo comando foi na seguinte ordem:

- Arquivo gerado pelo compilador e executado
- Pasta atual
- Arquivo do código com as instruções (extensão .s)

Isso ocorre pois o arquivo de código foi criado, depois chamado para compilar. Nesse momento, é criado o arquivo de saída que será executado, e isso faz uma alteração na pasta. Em seguida, o arquivo recém criado é executado.

A ordem dos processos condiz com o resultado obtido pelo comando **ls -a1t|more**

Exercício 2.4.2 - Stepping and stepping in

Observamos que o programa pára de rodar depois de executar

```
<main+20> swi 0x00123456
```

```
Register group: general
r0      0x0      0      r1      0x20026  131110
r2      0xffffffff -1     r3      0xa9c8   43464
r4      0x1      1      r5      0xffff8   2097144
r6      0x0      0      r7      0x0      0
r8      0x0      0      r9      0x0      0
r10     0x200100 2097408 r11     0x0      0
r12     0x1ffffc 2097100 sp      0x1ffff8 2097144
lr      0x8224   33316  pc      0x822c  33324
fps     0x0      0      cpsr    0x13     19

0x8210 <$d+16> andeq r10, r0, r0, lsr r1
0x8214 <$d+20> streqd r0, [r0], -pc
0x8218 <main>   mov    r0, #15 ; 0xf
0x821c <main+4> mov    r1, #20 ; 0x14
0x8220 <main+8> bl     0x8230 <firstfunc>
0x8224 <main+12> mov    r0, #24 ; 0x18
0x8228 <main+16> ldr    r1, [pc, #8] ; 0x8238 <$d>
0x822c <main+20> swi    0x00123456
0x8230 <firstfunc> adds   r0, r0, r1
0x8234 <firstfunc+4> mov    pc, lr
0x8238 <$d>      andeq  r0, r2, r6, lsr #32
0x823c <atexit>  mov    r12, sp

sim process 42 In: main
(gdb) s
firstfunc () at item-2-2.s:11
(gdb) s
(gdb) s
main () at item-2-2.s:7
(gdb) s
(gdb) s

Breakpoint 2, main () at item-2-2.s:9
(gdb) s

Breakpoint 2, main () at item-2-2.s:9
(gdb) █
```

Exercício 2.4.3 - Data formats

Temos abaixo o output de cada instrução descrita.

```
item-2-2.s
5      MOV     r1, #20
6      BL      firstfunc
7      MOV     r0, #0x18
8      LDR     r1, =0x20026
9      SWI     0x0
10     firstfunc:
> 11     ADDS    r0, r0, r1
12     MOV     pc, lr
13
14
15
0x821c <main+4>      mov     r1, #20 ; 0x14
0x8220 <main+8>      bl      0x8230 <firstfunc>
0x8224 <main+12>     mov     r0, #24 ; 0x18
0x8228 <main+16>     ldr     r1, [pc, #8] ; 0x8238 <$d>
0x822c <main+20>     swi     0x00000000
> 0x8230 <firstfunc> adds    r0, r0, r1
0x8234 <firstfunc+4> mov     pc, lr
0x8238 <$d>          andeq   r0, r2, r6, lsr #32
0x823c <atexit>      mov     r12, sp
0x8240 <atexit+4>    stmdb   sp!, {r4, r5, r11, r12, lr, pc}
0x8244 <atexit+8>    ldr     r5, [pc, #128] ; 0x82cc <$d>
0x8248 <atexit+12>   ldr     r3, [r5]
```

sim process 42 In: firstfunc Line: 11

Current language: auto; currently asm
(gdb) ../../../../newlib-1.12.0/newlib/libc/sys/arm/crt0.S:206: No such file or directory.
(gdb) p/x \$pc
\$1 = 0x822c
(gdb) p/x \$cpsr
\$2 = 0x13
(gdb) x/i \$pc
0x822c <main+20>: swi 0x00000000
(gdb) s
^Xfirstfunc () at item-2-2.s:11
(gdb) x/i \$pc
0x8230 <firstfunc>: adds r0, r0, r1
(gdb)

Exercício 3.10.1 - Signed and unsigned addition

Flags do CPSR, para referência:

When the processor enters an exception, the CPSR is saved to the appropriate SPSR. Figure 1-3 shows the format of the ARM status registers.



Figure 1-3 ARM status register format

Soma 1: 0xffff0000 + 0x87654321

```
File Edit View Search Terminal Help
Register group: general
r0 0xffff0000 -65536
r1 0x87654321 -2023472351
r2 0x1 1
r3 0x0 0
r4 0x0 0
r5 0x0 0
r6 0x0 0
r7 0x0 0
r8 0x0 0
r9 0x0 0
r10 0x200100 2097408
r11 0x1ffffc 2097100
r12 0x8224 33316
lr 0x0 0
fps 0x0 0
cpsr 0xa0000013 -1610612717

4 LDR r0, =0xffff0000 @MOV r0, #15
5 LDR r1, =0x87654321 @MOV r1, #20
6 BL firstfunc
7 MOV r0, #0x18
8 LDR r1, =0x20026
9 SWI 0x0
10 firstfunc:
11 ADDS r2, r0, r1
12 ADD r3, r0, r1
13 MOV pc, lr
14
15
16

Sim process 42 In: firstfunc Line: 12 PC: 0x8234
Current language: auto; currently asm
(gdb) n
(gdb) n
(gdb) b 10
Note: breakpoint 1 also set at pc 0x8230.
Breakpoint 2 at 0x8230: File sona1.s, line 10.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n)
Starting program: /home/student/src/a.out
Breakpoint 1, firstfunc () at sona1.s:11
Current language: auto; currently asm
(gdb) s
(gdb)
```

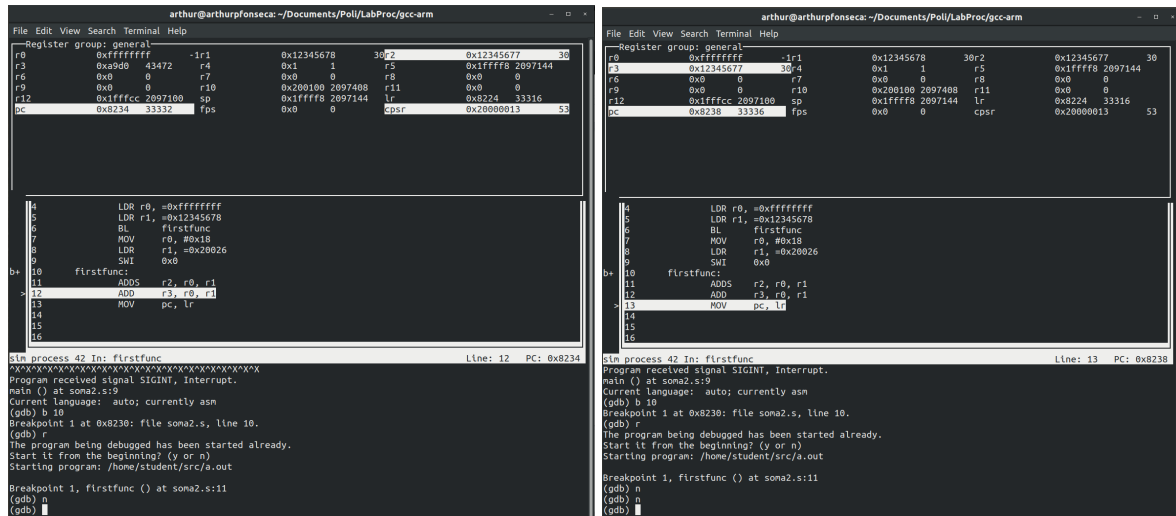
```
File Edit View Search Terminal Help
Register group: general
r0 0xffff0000 -65536
r1 0x87654321 -2023472351
r2 0x1 1
r3 0x0 0
r4 0x0 0
r5 0x1ffff8 2097144
r6 0x0 0
r7 0x0 0
r8 0x0 0
r9 0x0 0
r10 0x200100 2097408
r11 0x1ffffc 2097100
r12 0x8224 33316
lr 0x0 0
fps 0x0 0
cpsr 0xa0000013 -1610612717

4 LDR r0, =0xffff0000 @MOV r0, #15
5 LDR r1, =0x87654321 @MOV r1, #20
6 BL firstfunc
7 MOV r0, #0x18
8 LDR r1, =0x20026
9 SWI 0x0
10 firstfunc:
11 ADDS r2, r0, r1
12 ADD r3, r0, r1
13 MOV pc, lr
14
15
16

Sim process 42 In: firstfunc Line: 13 PC: 0x8238
Loading section .ctors, size 0x8 vma 0xa9d8
Loading section .jcr, size 0x4 vma 0xa9e0
Start address 0x8110
Transfer rate: 83744 bits/sec.
(gdb) b 11
Breakpoint 1 at 0x8230: File sona1.s, line 11.
(gdb) run
Starting program: /home/student/src/a.out
Breakpoint 1, firstfunc () at sona1.s:11
Current language: auto; currently asm
(gdb) n
(gdb) n
(gdb)
```

Vê-se que as flags do CPSR se tornam A (1010) depois de ADDS. Isso significa que o resultado foi negativo e houve carry. Como esperado, depois da operação de ADD, não há alteração nas flags.

Soma 2: 0xffffffff + 0x12345677



```
File Edit View Search Terminal Help
--Register group: general--
r0 0xffffffff -lr1 0x12345678 3br2 0xffffffff 30
r3 0x0000 43472 r4 0x1 1 r5 0xfffff8 2097144
r6 0 0 r7 0x0 0 r8 0x0 0
r9 0x0 0 r10 0x200100 2097408 r11 0x0 0
r12 0xffffcc 2097100 sp 0xfffff8 2097144 lr 0x224 33316
pc 0x8234 33332 fps 0x0 0 cpsr 0x20000013 53

4 LDR r0, =0xffffffff
5 LDR r1, =0x12345678
6 BL firstfunc
7 MOV r0, #0x18
8 LDR r1, =0x20026
9 SWI 0x0
10 firstfunc:
11 ADDS r2, r0, r1
12 MOV pc, lr
13
14
15
16

Sim process d2 In: firstfunc Line: 12 PC: 0x8234
Program received signal SIGINT, Interrupt.
main () at soma2.s:9
Current language: auto; currently asm
(gdb) b 10
Breakpoint 1 at 0x8230: file soma2.s, line 10.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n)
Starting program: /home/student/src/a.out
Breakpoint 1, firstfunc () at soma2.s:11
(gdb) n
(gdb)
```

```
File Edit View Search Terminal Help
--Register group: general--
r0 0xffffffff -lr1 0x12345678 3br2 0xffffffff 30
r3 0x12345677 20 4 0x1 1 r5 0xfffff8 2097144
r6 0x0 0 r7 0x0 0 r8 0x0 0
r9 0x0 0 r10 0x200100 2097408 r11 0x0 0
r12 0xffffcc 2097100 sp 0xfffff8 2097144 lr 0x224 33316
pc 0x8230 33336 fps 0x0 0 cpsr 0x20000013 53

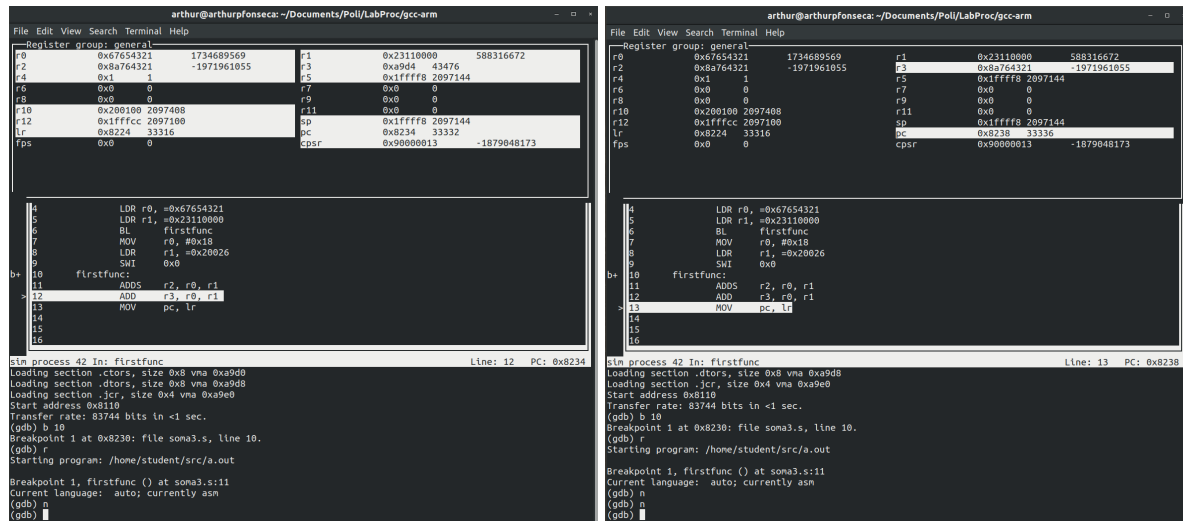
4 LDR r0, =0xffffffff
5 LDR r1, =0x12345678
6 BL firstfunc
7 MOV r0, #0x18
8 LDR r1, =0x20026
9 SWI 0x0
10 firstfunc:
11 ADDS r2, r0, r1
12 MOV r3, r0, r1
13 MOV pc, lr
14
15
16

Sim process d2 In: firstfunc Line: 13 PC: 0x8230
Program received signal SIGINT, Interrupt.
main () at soma2.s:9
Current language: auto; currently asm
(gdb) b 10
Breakpoint 1 at 0x8230: file soma2.s, line 10.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n)
Starting program: /home/student/src/a.out
Breakpoint 1, firstfunc () at soma2.s:11
(gdb) n
(gdb)
```

Vê-se que as flags do CPSR se tornam 2 (0010) depois de ADDS.

Isso significa que o resultado foi positivo e houve carry. Como esperado, depois da operação de ADD, não há alteração nas flags.

Soma 3: 0x67654321 + 0x23110000



```
File Edit View Search Terminal Help
Register group: general
r0 0x67654321 1734689569 r1 0x23110000 588316672
r2 0x8a764321 -1971961055 r3 0xa9d4 43476
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 0x1ffffc 2097100 sp 0x1ffff8 2097144
lr 0x8224 33316 pc 0x8234 33332
fps 0x0 0 cpsr 0x90000013 -1879048173

4 LDR r0, =0x67654321
5 LDR r1, =0x23110000
6 BL firstfunc
7 MOV r0, #0x18
8 LDR r1, =0x20026
9 SWI 0x0
10 firstfunc:
11 ADDS r2, r0, r1
12 ADD r3, r0, r1
13 MOV pc, lr
14
15
16
$M process 42 In: firstfunc Line: 12 PC: 0x8234
Loading section .ctors, size 0x8 vma 0xa9d0
Loading section .dtors, size 0x8 vma 0xa9d8
Loading section .jcr, size 0x4 vma 0xa9e0
Start address 0x8110
Transfer rate: 83744 bits in <1 sec.
(gdb) b 10
Breakpoint 1 at 0x8230: file soma3.s, line 10.
(gdb) r
Starting program: /home/student/src/a.out
Breakpoint 1, firstfunc () at soma3.s:11
Current language: auto; currently asm
(gdb) n
(gdb)

File Edit View Search Terminal Help
Register group: general
r0 0x67654321 1734689569 r1 0x23110000 588316672
r2 0x8a764321 -1971961055 r3 0xa9d4 43476
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 0x1ffffc 2097100 sp 0x1ffff8 2097144
lr 0x8224 33316 pc 0x8238 33336
fps 0x0 0 cpsr 0x90000013 -1879048173

4 LDR r0, =0x67654321
5 LDR r1, =0x23110000
6 BL firstfunc
7 MOV r0, #0x18
8 LDR r1, =0x20026
9 SWI 0x0
10 firstfunc:
11 ADDS r2, r0, r1
12 ADD r3, r0, r1
13 MOV pc, lr
14
15
16
$M process 42 In: firstfunc Line: 13 PC: 0x8238
Loading section .ctors, size 0x8 vma 0xa9d0
Loading section .dtors, size 0x8 vma 0xa9d8
Loading section .jcr, size 0x4 vma 0xa9e0
Start address 0x8110
Transfer rate: 83744 bits in <1 sec.
Breakpoint 1 at 0x8230: file soma3.s, line 10.
(gdb) b 10
Starting program: /home/student/src/a.out
Breakpoint 1, firstfunc () at soma3.s:11
Current language: auto; currently asm
(gdb) n
(gdb)
(gdb)
```

Vê-se que as flags do CPSR se tornam 9 (1001) depois de ADDS. Isso significa que o resultado foi negativo, não houve carry e houve overflow. Como esperado, depois da operação de ADD, não há alteração nas flags.

Portanto, após os testes, notamos que as diferenças entre os comandos é apenas dada na mudança das flags CPSR. O *signed* identifica o overflow na soma e ativa a flag V. Em contrapartida, a soma sem sinal não identifica o overflow.

Exercício 3.10.2 - Multiplication / Multiplicação de número

```
arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0xffffffff -1
r2      0x80000000 -2147483648
r4      0x1        1
r6      0x0        0
r8      0x0        0
r10     0x200100   2097408
r12     0x1ffffc   2097100
lr      0x8224     33316
fps     0x0        0
r1      0x80000000 -2147483648
r3      0xa9c8     43464
r5      0x1ffff8   2097144
r7      0x0        0
r9      0x0        0
r11     0x0        0
sp      0x1ffff8   2097144
pc      0x8234     33332
cpsr    0xa0000013 -1610612717

4      LDR r0, =0xffffffff
5      LDR r1, =0x80000000
6      BL firstfunc
7      MOV r0, #0x18
8      LDR r1, =0x20026
9      SWI 0x0
b+ 10 firstfunc:
11     MULL r2, r0, r1
> 12     MOV pc, lr
13
14
15
16

sim process 42 In: firstfunc Line: 12 PC: 0x8234
Loading section .ctors, size 0x8 vma 0xa9c4
Loading section .dtors, size 0x8 vma 0xa9cc
Loading section .jcr, size 0x4 vma 0xa9d4
Start address 0x8110
Transfer rate: 41824 bits/sec.
(gdb) b 10
Breakpoint 1 at 0x8230: file mult.s, line 10.
(gdb) r
Starting program: /home/student/src/a.out

Breakpoint 1, firstfunc () at mult.s:11
Current language: auto; currently asm
(gdb) n
(gdb) █
```

1. Does your result make sense? Why or why not?

Não faz sentido, pois é uma multiplicação entre 2 números bem grandes e o resultado está idêntico a um dos operandos. Houve overflow e também é possível notar que os resultados tanto das flags quanto dos registradores não são confiáveis neste caso..

2. Assuming that these two numbers are signed integers, is it possible to overflow in this case?

Sim, é possível dado que é uma multiplicação entre dois elementos de 32 bits sendo armazenada em um registrador de 32 bits

3. Why is there a need for two separate long multiply instructions, UMULL and SMULL? Give an example to support your answer.

São necessários para que se possa armazenar o resultado da multiplicação de dois elementos de 32 bits em um elemento de resposta de 64 bits, evitando overflow e possibilitando o cálculo correto.

Exercício 3.10.3 - Multiplication shortcuts / Multiplicação pelo número 3

```

arthur@arthurpfonseca: ~/Documents/Polí/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0x1      1
r2      0xffffffff -1
r4      0x1      1
r6      0x0      0
r8      0x0      0
r10     0x200100 2097408
r12     0x1ffffc 2097100
lr      0x81fc   33276
fps     0x0      0
r1      0x20     32
r3      0xa8b0   43184
r5      0x1ffff8 2097144
r7      0x0      0
r9      0x0      0
r11     0x0      0
sp      0x1ffff8 2097144
pc      0x8220   33312
cpsr    0x60000013 1610612755

4      LDR r0, =0x00000001
5      MOV r1, r0, lsl #5
> 6      SWI 0x00
7
8
9
10
11
12
13
14
15
16

sim process 42 In: main                               Line: 6    PC: 0x8220
Loading section .ctors, size 0x8 vma 0xa8ac
Loading section .dtors, size 0x8 vma 0xa8b4
Loading section .jcr, size 0x4 vma 0xa8bc
Start address 0x8110
Transfer rate: 83456 bits/sec.
(gdb) b 5
Breakpoint 1 at 0x821c: file mult_um_ciclo.s, line 5.
(gdb) run
Starting program: /home/student/src/a.out

Breakpoint 1, main () at mult_um_ciclo.s:5
Current language: auto; currently asm
(gdb) n
(gdb)

```

Como esperado, observamos que o resultado de uma multiplicação de 0x1 (*r0*) por 32 (através de um left shift de 5 posições) resulta em 32 (*r1*).

Exercício 3.10.4 - Register-swap algorithm

```
arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0xf631024c      -164560308    r1      0x17539abd      391355069
r2      0xffffffff      -1            r3      0xa9c4      43460
r4      0x1      1            r5      0x1fffff8      2097144
r6      0x0      0            r7      0x0      0
r8      0x0      0            r9      0x0      0
r10     0x200100      2097408      r11     0x0      0
r12     0x1ffffcc      2097100      sp      0x1fffff8      2097144
lr      0x81fc      33276        pc      0x8220      33312
fps      0x0      0            cpsr     0x60000013      1610612755

4          LDR r0, =0xF631024C
5          LDR r1, =0x17539ABD
b+> 6      EOR r0, r0, r1
b+ 7      EOR r1, r0, r1
b+ 8      EOR r0, r0, r1
9          SWI 0x0
10
11
12
13
14
15
16

sim process 42 In: main                               Line: 6    PC: 0x8220
Starting program: /home/student/src/a.out

Breakpoint 1, main () at swap.s:6
Current language: auto; currently asm
(gdb) b 5
Breakpoint 4 at 0x821c: file swap.s, line 5.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n)
Starting program: /home/student/src/a.out

Breakpoint 4, main () at swap.s:5
(gdb) n
(gdb) █
```

Estado inicial do programa, antes da execução do algoritmo de swap.

```

    arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0x1b      27
r2      0xffffffff -1
r4      0x1       1
r6      0x0       0
r8      0x0       0
r10     0x200100 2097408
r12     0x1ffffc 2097100
lr      0x8224   33316
fps     0x0       0
r1      0x14      20
r3      0xa9c4   43460
r5      0x1ffff8 2097144
r7      0x0       0
r9      0x0       0
r11     0x0       0
sp      0x1ffff8 2097144
pc      0x822c   33324
cpsr    0x60000013 1610612755

3      main:
4          MOV     r0, #15
5          MOV     r1, #20
6          BL      firstfunc
7          SWI     0x0
b+ 8      firstfunc:
b+ 9          EOR     r0, r0, r1
b+> 10         EOR     r1, r0, r1
b+ 11         EOR     r0, r0, r1
12         MOV     pc, lr
13
14
15

sim process 42 In: firstfunc                               Line: 10   PC: 0x822c
(gdb) b 9
Note: breakpoint 1 also set at pc 0x8228.
Breakpoint 2 at 0x8228: file swap.s, line 9.
(gdb) b 10
Breakpoint 3 at 0x822c: file swap.s, line 10.
(gdb) b 11
Breakpoint 4 at 0x8230: file swap.s, line 11.
(gdb) run
Starting program: /home/student/src/a.out

Breakpoint 1, firstfunc () at swap.s:9
Current language: auto; currently asm
(gdb) n
(gdb) █

```

Após a execução da primeira instrução de EOR.

```

arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0xe16298f1      -513632015
r2      0xffffffff      -1
r4      0x1            1
r6      0x0            0
r8      0x0            0
r10     0x200100      2097408
r12     0x1ffffc      2097100
lr      0x81fc        33276
fps     0x0            0
r1      0xf631024c     -164560308
r3      0xa9c4        43460
r5      0x1ffff8      2097144
r7      0x0            0
r9      0x0            0
r11     0x0            0
sp      0x1ffff8      2097144
pc      0x8228        33320
cpsr    0x60000013     1610612755

4      LDR r0, =0xF631024C
B+ 5      LDR r1, =0x17539ABD
b+ 6      EOR r0, r0, r1
b+ 7      EOR r1, r0, r1
b+> 8      EOR r0, r0, r1
9      SWI 0x0
10
11
12
13
14
15
16

sim process 42 In: main                               Line: 8    PC: 0x8228
Breakpoint 1, main () at swap.s:6
Current language: auto; currently asm
(gdb) b 5
Breakpoint 4 at 0x821c: file swap.s, line 5.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n)
Starting program: /home/student/src/a.out

Breakpoint 4, main () at swap.s:5
(gdb) n
(gdb) n
(gdb) n
(gdb) █
```

Após a execução da segunda instrução de EOR.

```

arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0x17539abd      391355069      r1      0xf631024c      -164560308
r2      0xffffffff      -1            r3      0xa9c4      43460
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     0x1ffffc      2097100      sp      0x1ffff8      2097144
lr      0x81fc      33276        pc      0x822c      33324
fps     0x0      0            cpsr    0x60000013      1610612755

4          LDR r0, =0xF631024C
B+ 5          LDR r1, =0x17539ABD
b+ 6          EOR r0, r0, r1
b+ 7          EOR r1, r0, r1
b+ 8          EOR r0, r0, r1
> 9          SWI 0x0
10
11
12
13
14
15
16

sim process 42 In: main                                Line: 9    PC: 0x822c
Current language: auto; currently asm
(gdb) b 5
Breakpoint 4 at 0x821c: file swap.s, line 5.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n)
Starting program: /home/student/src/a.out

Breakpoint 4, main () at swap.s:5
(gdb) n
(gdb) n
(gdb) n
(gdb) n
(gdb) █
```

Após a execução da terceira instrução de EOR. Os valores dos registradores terminaram trocados.

Apêndice

1. Exercício 3.10.1

1ª Soma

```
@ Exercicio 3.10.1 : 1a soma do exercicio
@ Para debugar este codigo:
@ gcc mult.s && gdb a.out
    .text
    .globl main
main:
    LDR r0, =0xffff0000 @MOV r0, #15
    LDR r1, =0x87654321 @MOV r1, #20
    BL firstfunc
    MOV r0, #0x18
    LDR r1, =0x20026
    SWI 0x0
firstfunc:
    ADDS r2, r0, r1 @ breakpoint aqui
    ADD r3, r0, r1 @ outro breakpoint aqui (dá step para ir de um pro
outro)
    MOV pc, lr
```

2ª Soma

```
@ Exercicio 3.10.1 : 2a soma do exercicio
@ Para debugar este codigo:
@ gcc mult.s && gdb a.out
    .text
    .globl main
main:
    LDR r0, =0xffffffff
    LDR r1, =0x12345678
    BL firstfunc
    MOV r0, #0x18
    LDR r1, =0x20026
    SWI 0x0
firstfunc:
    ADDS r2, r0, r1 @ breakpoint aqui
    ADD r3, r0, r1 @ breakpoint aqui
    MOV pc, lr
```

3ª Soma

```
@ Exercicio 3.10.1 : 3a soma do exercicio
@ Para debugar este codigo:
@ gcc mult.s && gdb a.out
    .text
    .globl main
main:
    LDR r0, =0x67654321
    LDR r1, =0x23110000
    BL   firstfunc
    MOV  r0, #0x18
    LDR  r1, =0x20026
    SWI  0x0
firstfunc:
    ADDS r2, r0, r1 @ breakpoint aqui
    ADD  r3, r0, r1 @ breakpoint aqui
    MOV  pc, lr
```

2. Exercício 3.10.2

```
@ Exercicio 3.10.2 do livro
@ Para debugar este codigo:
@ gcc mult.s && gdb a.out
    .text
    .globl main
main:
    LDR r0, =0xffffffff
    LDR r1, =0x80000000
    BL   firstfunc
    MOV  r0, #0x18
    LDR  r1, =0x20026
    SWI  0x0
firstfunc:
    MULS r2, r0, r1      @ break nesta linha
    MOV  pc, lr
```

3. Exercício 3.10.3

```
@ Exercicio 3.10.3 : shift de 5 : multiplicar por 32
@ Para debugar este codigo:
@ gcc mult.s && gdb a.out
.text
.globl main
main:
    LDR r0, =0X00000001
    MOV r1, r0, lsl #5 @ breakpoint aqui
    SWI 0x00
```

4. Exercício 3.10.4

```
@ Exercicio 3.10.4 : algoritmo de SWAP usando XOR
@
@ Algoritmo
@ A = A XOR B
@ B = A XOR B
@ A = A XOR B
@
@ Para debugar este codigo:
@ gcc mult.s && gdb a.out
.text
.globl main
main:
    LDR r0, =0xF631024C
    LDR r1, =0x17539ABD
    EOR r0, r0, r1 @ breakpoint aqui
    EOR r1, r0, r1 @ breakpoint aqui
    EOR r0, r0, r1 @ breakpoint aqui
    SWI 0x0
```