

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



Aula 03
20 de Maio de 2021

GRUPO 10

Arthur Pires da Fonseca
Bruno José Mório
Iago Soriano Roque Monteiro

NUSP: 10773096
NUSP: 10336852
NUSP: 8572921

Sumário

Exercício 3.10.5 - Signed multiplication	3
Exercício 3.10.6 - Absolute Value	5
Exercício 3.10.7 - Division	6
Exercício 3.10.8 - Gray codes	7
Apêndice	8
Exercício 3.10.5 - Signed Multiplication	8
Exercício 3.10.6 - Absolute Value	8
Exercício 3.10.7 - Division	9
Exercício 3.10.8 - Gray codes	10

Exercício 3.10.5 - Signed multiplication

```
Register group: general
r0      0x2      2
r1      0x4      4
r2      0xffffffff -3
r3      0x8      8
r4      0x1      1
r5      0x1      1
r6      0x0      0
r7      0x0      0

mult_signed.s
4      .text
5      .globl main
6      main:
7          ldr r0, #-2
8          ldr r1, #-4
9          ldr r5, =1      @ fator para o resultado final
10         ldr r6, =0
11

Sim process 42 In: main                               Line: 27   PC: 0x8254
Transfer rate: 41968 bits/sec.
(gdb) b 27
Breakpoint 1 at 0x8254: file mult_signed.s, line 27.
(gdb) r
Starting program: /home/student/src/lab3-20-05/a.out

Breakpoint 1, main () at mult_signed.s:27
Current language: auto; currently asm
(gdb)
```

Multiplicando dois números negativos.

```
Register group: general
r0      0x2      2
r1      0x4      4
r2      0xffffffff -3
r3      0xffffffff -8
r4      0x1      1
r5      0xffffffff -1
r6      0x0      0
r7      0x0      0

mult_signed.s
4      .text
5      .globl main
6      main:
7          ldr r0, #-2
8          ldr r1, =4
9          ldr r5, =1      @ fator para o resultado final
10         ldr r6, =0
11

Sim process 42 In: main                               Line: 27   PC: 0x8254
Transfer rate: 41968 bits/sec.
(gdb) b 27
Breakpoint 1 at 0x8254: file mult_signed.s, line 27.
(gdb) r
Starting program: /home/student/src/lab3-20-05/a.out

Breakpoint 1, main () at mult_signed.s:27
Current language: auto; currently asm
(gdb)
```

Multiplicando um número negativo e um positivo

```
Register group: general
r0      0x2      2
r1      0x4      4
r2      0x2      2
r3      0x8      8
r4      0x1      1
r5      0x1      1
r6      0x0      0
r7      0x0      0

mult_signed.s
4      .text
5      .globl  main
6      main:
7          ldr r0, =2
8          ldr r1, =4
9          ldr r5, =1      @ fator para o resultado final
10         ldr r6, =0
11

sim process 42 In: main                                Line: 27
Transfer rate: 41936 bits/sec.
(gdb) b 27
Breakpoint 1 at 0x8254: file mult_signed.s, line 27.
(gdb) r
Starting program: /home/student/src/lab3-20-05/a.out

Breakpoint 1, main () at mult_signed.s:27
Current language: auto; currently asm
```

Multiplicando dois números positivos

Exercício 3.10.6 - Absolute Value

```
Activities Terminal mai 20 15:14
bruno@bruno-340XAA-350XAA-550XAA: ~/Documents/Poli/gcc-arm

Register group: general
r0 0x2 2 r1 0xffffffff -3 r2 0xffffffff -1
r3 0xa9b8 43448 r4 0x1 1 r5 0xffffffff 2097144
r6 0x0 0 r7 0x0 0 r8 0x0 0
r9 0x0 0 r10 0x200100 2097408 r11 0x0 0
r12 0x1fffc 2097100 sp 0x1ffff8 2097144 lr 0x81fc 33276
pc 0x8228 33320 fps 0x0 0 cpsr 0x60000013 1610612755

B+ 4 MOV r0, #-2
5 MOV r1, #0
6 ADD r1, r0, r0, ASR #31
7 EOR r0, r1, r0, ASR #31
> 8 SWI 0x123456
9
10
11
12
13
14

sim process 42 In: main Line: 8 PC: 0x8228
(gdb) b 9
No line 9 in file "item3-10-5.s".
(gdb) r
Starting program: /home/student/src/a.out
Breakpoint 1, main () at item3-10-5.s:4
Current language: auto; currently asm
(gdb) n
(gdb) n
(gdb) n
(gdb) n
(gdb) n
(gdb) 
```

Exercício 3.10.7 - Division

```
Register group: general
r0      0x3e8    1000
r1      0x399    921
r2      0x1f4    500
r3      0x217c   8572
r4      0x0      0
r5      0x399    921
r6      0x0      0

division.s
3      main:
4          ldr r1, =8572921 @ Dividendo (=8572921)
5          ldr r2, =1000    @ Divisor (=1000)
6          mov r3, #0      @ Quociente
7          mov r5, #0      @ Resto
8
9          mov r4, #0      @ Tamanho do shift

sim process 42 In: end2                                Line: 50   PC:
(gdb) b 50
Breakpoint 1 at 0x828c: file division.s, line 50.
(gdb) r
Starting program: /home/student/src/lab3-20-05/a.out

Breakpoint 1, end2 () at division.s:50
Current language:  auto; currently asm
(gdb)
```

Exercício 3.10.8 - Gray codes

Antes de rodar o código:

```
arthur@arthurfonseca: ~/Documents/Pol/LabProc/gcc-arm
File Edit View Search Terminal Help
--Register group: general--
r0      0x1      1          r1      0xb4     180
r2      0x0      0          r3      0x3      3
r4      0x0      0          r5      0x2      2
r6      0x0      0          r7      0x0      0
r8      0x3      3          r9      0x0      0
r10     0x200100 2097408    r11     0x0      0
r12     0x1fffc4 2097092    sp      0x1ffff0 2097136
lr      0x81fc   33276     pc      0x8240   33344
fps      0x0      0          cpsr    0x6000013 1610612755

grey.s
24      ldr r9, =0          @ Guarda r5 temporariamente
25
26      ldr r6, =0          @ Contem o valor dos bits que importan
27
28      loop1:
29      and r6, r1, r3, lsl r4 @ AND entre mascara e codigo, pega so os bits que impo
30      mov r6, r6, lsr r4    @ Shift de (bits do grey) posicoes
31      add r2, r2, r6, lsl r7 @ Soma ao grey do futuro
32
33      add r4, r4, r5        @ Soma a qtde de bits originais de grey ao contador
34      add r7, r7, r8        @ Soma os bits futuros
35
36      mov r9, r5           @ Guarda o valor de r5

sim process 42 In: loop1 Line: 29 PC: 0x8240
Loading section .dtors, size 0x8 vma 0xaa1c
Loading section .jcr, size 0x4 vma 0xaa24
Start address 0x8110
Transfer rate: 84288 bits/sec.
(gdb) b 28
Breakpoint 1 at 0x8240: file grey.s, line 28.
(gdb) b 59
Breakpoint 2 at 0x8288: file grey.s, line 59.
(gdb) r
Starting program: /home/student/src/Lab3/Lab de fato/a.out

Breakpoint 1, loop1 () at grey.s:29
Current language: auto; currently asm
(gdb) █
```

Depois de rodar o código:

```
arthur@arthurfonseca: ~/Documents/Pol/LabProc/gcc-arm
File Edit View Search Terminal Help
--Register group: general--
r0      0x1      1          r1      0xb4     180
r2      0x97e4c8 9954504    r3      0x3      3
r4      0x0      0          r5      0x2      2
r6      0x4      4          r7      0x18     24
r8      0x3      3          r9      0x2      2
r10     0x200100 2097408    r11     0x0      0
r12     0x1fffc4 2097092    sp      0x1ffff0 2097136
lr      0x81fc   33276     pc      0x8288   33416
fps      0x0      0          cpsr    0x6000013 1610612755

grey.s
54
55      cmp r4, #0 @ Acho que e zero
56
57      bgt loop2 @ Se for maior do que zero ainda
58
59      swi 0x0
60
61
62
63
64
65
66

sim process 42 In: loop2 Line: 59 PC: 0x8288

Breakpoint 1, loop1 () at grey.s:29
(gdb) c
Continuing.

Breakpoint 1, loop1 () at grey.s:29
(gdb) c
Continuing.

Breakpoint 1, loop1 () at grey.s:29
Continuing.

Breakpoint 2, loop2 () at grey.s:59
(gdb) █
```

Apêndice

1. Exercício 3.10.5 - Signed Multiplication

```
@ Exercicio 3.10.5 da apostila
@ Para debugar este codigo:
@ gcc mult_signed.s && gdb a.out

.text
.globl main
main:
    ldr r0, =2
    ldr r1, =4
    ldr r5, =1    @ fator para o resultado final
    ldr r6, =0

    CMP r0, #0
    SUBLT r5, r6, r5, lsl #0
    CMP r1, #0
    SUBLT r5, r6, r5, lsl #0

    MOV r2, #0
    ADD r2, r1, r1, ASR #31
    EOR r1, r2, r1, ASR #31

    ADD r2, r0, r0, ASR #31
    EOR r0, r2, r0, ASR #31

    MUL r3, r0, r1
    MUL r3, r5, r3

    swi 0x0
```

2. Exercício 3.10.6 - Absolute Value

```
@ Exercicio 3.10.6 da apostila
@ Para debugar este codigo:
@ gcc absolute_val.s && gdb a.out

.text
.globl main
main:
    MOV r0, #-2
    MOV r1, #0
    ADD r1, r0, r0, ASR #31
    EOR r0, r1, r0, ASR #31
    SWI 0x123456
```


3. Exercício 3.10.7 - Division

```
@ Exercicio 3.10.7 da apostila
@ Para debugar este codigo:
@ gcc division.s && gdb a.out

.text
.globl main
main:
    ldr r1, =0x000003FD @ Dividendo (=1021)
    mov r2, #25 @ Divisor (=25)
    mov r3, #0 @ Quociente
    mov r5, #0 @ Resto

    mov r4, #0 @ Tamanho do shift
    mov r0, r2 @ Guarda o valor inicial do divisor
    b align

    @ Divisao
    @ Alinhar bits mais à esquerda

align:
    mov r2, r2, lsl #1
    cmp r2, r1
    ble align @ r2 (divisor) < r1 (dividendo) => continua

div_loop:
    cmp r1, r2
    bge quociente_1 @ r1 (dividendo) >= r2 (divisor) ?
    b quociente_0

loop_end:
    mov r2, r2, lsr #1 @ Desloca para a direita o divisor
    cmp r1, r0
    blt end @ r1 (dividendo) < r0 (divisor original) => acabou divisao
    b div_loop @ Senao, continua o loop

quociente_1:
    sub r1, r1, r2 @ Tira o divisor do dividendo
    mov r3, r3, lsl #1 @ Desloca o quociente
    add r3, r3, #1 @ Adiciona um 1 ao fim do quociente
    b loop_end

quociente_0:
    mov r3, r3, lsl #1
    b loop_end

end:
    cmp r2, r0
    blt end2 @ Se o divisor voltou ao valor original, nao tem mais o
que fazer
```

```
mov r2, r2, lsr #1 @ Divide divisor por 2
```

4. Exercício 3.10.8 - Gray codes

```
@ Exercicio 3.10.8 da apostila
@ Para debugar este codigo:
@ gcc grey.s && gdb a.out
@Codigo grey exemplo: 000 001 011 010 110 111 101 100
@ Grey 2: 10 11 01 00
@ Algoritmo: 0 10, 0 11, 0 01, 0 00, 1 00, 1 01, 1 11, 1 10
@ 10 11 01 00 = 0xb4
@ VIRA
@ 100 101 111 110 010 011 001 000 = 1001 0111 1110 0100 1100 1000 =
0x97e4c8

        .text
        .globl main
main:
    ldr r0, =1          @ Constante 1

    ldr r1, =0x000000b4 @ Codigo grey de 2 bits
    ldr r2, =0          @ Futuro codigo grey de 3 bits

    ldr r3, =0x00000003 @ Mascara para pegar 2 bits
    ldr r4, =0          @ Contador de bits para shift no codigo original
    ldr r7, =0          @ Contador de bits para shift no codigo futuro
    ldr r5, =2          @ Qtde de bits do codigo grey original
    ldr r8, =3          @ Qtde de bits do codigo grey futuro

    ldr r9, =0          @ Guarda r5 temporariamente

    ldr r6, =0          @ Contem o valor dos bits que importam

loop1:
    and r6, r1, r3, lsl r4 @ AND entre mascara e codigo, pega so os bits
que importam
    mov r6, r6, lsr r4    @ Shift de (bits do grey) posicoes
    add r2, r2, r6, lsl r7 @ Soma ao grey do futuro

    add r4, r4, r5        @ Soma a qtde de bits originais de grey ao contador
    add r7, r7, r8        @ Soma os bits futuros

    mov r9, r5            @ Guarda o valor de r5
    mov r5, r0, lsl r5    @ Sobrescreve com o valor do shift (1 * 2^(bits do
grey))
    cmp r4, #8 @r0, lsl r5 @ Ve se r4 e maior que 2^(bits do grey)
    mov r5, r9            @ Recupera valor de r5

    blt loop1            @ Continua o loop se contador < 2^(bits do grey)
loop2:
```

```
sub r4, r4, r5  
@sub r7, r7, r8
```

```
and r6, r1, r3, lsl r4 @ Mesmo codigo de cima  
mov r6, r6, lsr r4  
add r6, r6, r0, lsl r5 @ Soma 100 (qtde de zeros necessarias) pra fazer  
um complemento do loop1  
add r2, r2, r6, lsl r7
```

```
@sub r4, r4, r5  
add r7, r7, r8
```

```
cmp r4, #0 @ Acho que e zero
```

```
bgt loop2 @ Se for maior do que zero ainda
```

```
swi 0x0
```