

Laboratório de Microprocessadores
PCS3732 - 2020



Escola Politécnica da USP

Prof. Jorge Kinoshita

Marcos Tan Chi Chen - 9833065

Vitor Dias - 9344786

Samuel Ducca - 9833169

Exercício 5.5.2

Este exercício iremos realizar a construção de um código assembly que calcula o valor de fatorial n. No exemplo abaixo iremos calcular o fatorial de 10.

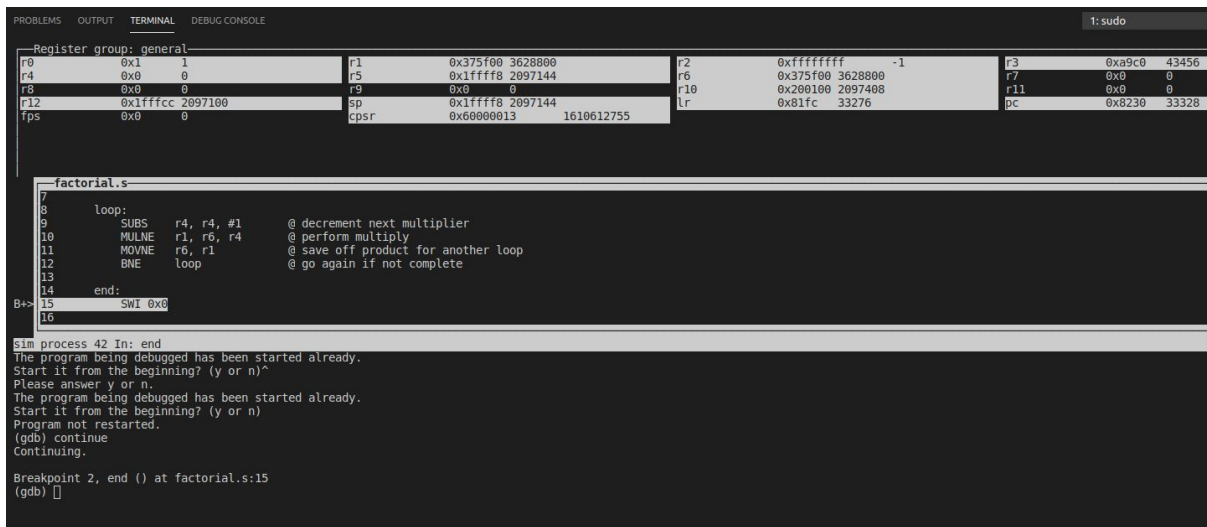
```
.text
    .globl main

main:
    MOV r6,#0xA          @ load 10 into r6
    MOV r4,r6            @ copy n into a temp register

loop:
    SUBS    r4, r4, #1    @ decrement next multiplier
    MULNE   r1, r6, r4    @ perform multiply
    MOVNE   r6, r1        @ save off product for another loop
    BNE     loop          @ go again if not complete

end:
    SWI 0x0
```

O resultado esperado do fatorial de 10 é 3.628.800. Como podemos ver no registrador r1 o valor esperado foi obtido:



The screenshot shows a GDB terminal window with the following components:

- Register group: general**

Register	Value
r0	0x1 1
r4	0x0 0
r8	0x0 0
r12	0x1ffcc 2097100
fps	0x0 0
r1	0x375f00 3628800
r5	0x1ffff8 2097144
r9	0x0 0
sp	0x1ffff8 2097144
cpsr	0x60000013 1610612755
r2	0xffffffff -1
r6	0x375f00 3628800
r10	0x200100 2097408
lr	0x01fc 33276
r3	0xa9c0 43456
r7	0x0 0
r11	0x0 0
pc	0x8230 33328
- factorial.s**

```
7
8     loop:
9         SUBS    r4, r4, #1    @ decrement next multiplier
10        MULNE   r1, r6, r4    @ perform multiply
11        MOVNE   r6, r1        @ save off product for another loop
12        BNE     loop          @ go again if not complete
13
14    end:
15        SWI 0x0
16
```
- Console output**

```
sim process 42 In: end
The program being debugged has been started already.
Start it from the beginning? (y or n)^
Please answer y or n.
The program being debugged has been started already.
Start it from the beginning? (y or n)
Program not restarted.
(gdb) continue
Continuing.
Breakpoint 2, end () at factorial.s:15
(gdb) 
```

Exercício 5.5.3

Esta atividade consiste em encontrar o maior número em uma sequência de números de 32 bits, usando execução condicional sempre que possível. Para isso, foi feito o seguinte código:

```

.text
.globl main

main:
    MOV r5, #11      @ 11 numeros para comparar
    MOV r0, #1       @ j = 1
    MOV r3, #0       @ reg para guardar temporariamente
    LDR r1, =dados+4 @ ponteiro para o começo dos dados, primeiro end eh pro
maior valor
    LDR r2, [r1], #4  @ carrega primeiro valor para o registrador de 'maior
numero'

loop:
    CMP r0, r5       @ j < r5?
    BGE done         @ j >= r5, termina

    LDR r3, [r1], #4  @ Carrega próximo numero em r3
    CMP r2, r3       @ Compara numero com maior valor
    MOVLT r2, r3     @ Se o numero de r2 eh menor, colocar no novo numero no
lugar
    ADD r0, r0, #1    @ j++
    B loop

done:
    STR r2, dados     @Armazena maior valor no começo de dados
end:
    SWI 0x0

dados:
    .word 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xa, 0xb

```

A seguir estão as imagens da memória no final da execução do programa, bem como do gdb:

```

0x8250 <dados>: 0x0000000b  0x00000001  0x00000002  0x00000003
0x8260 <dados+16>: 0x00000004  0x00000005  0x00000006  0x00000007
0x8270 <dados+32>: 0x00000008  0x00000009  0x0000000a  0x0000000b

```

Maior valor armazenado corretamente no início de “dados”

```

samuelducca@samuelducca-N501VW: ~/gcc-arm
Register group: general
r0 0xb 11 r1 0x8280 33408r2 0xb 11 r3 0xb 11 r4 0xb 11
r5 0xb 11 r6 0x0 0 r7 0x0 0 r8 0x0 0 r9 0x0 0
r10 0x200100 20974r11 0x0 0 r12 0xffffcc 20971sp 0xfffff8 20971lr 0x81fc 33276
pc 0x8248 33352fps 0x0 0 cpsr 0x60000013

item-5-5-3.s
15 LDR r3, [r1], #4 @ Carrega próximo numero em r3
16 CMP r2, r3 @ Compara numero com maior valor
17 MOVLIT r2, r3 @ Se o numero de r2 eh menor, colocar no novo numero no lugar
18 ADD r0, r0, #1 @ j++
19 B loop
20
21 done:
22 STR r2, dados @Armazena maior valor no começo de dados
23 end:
24 SWI 0x0
25
26 dados:
27 .word 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xa, 0xb
28
29
30

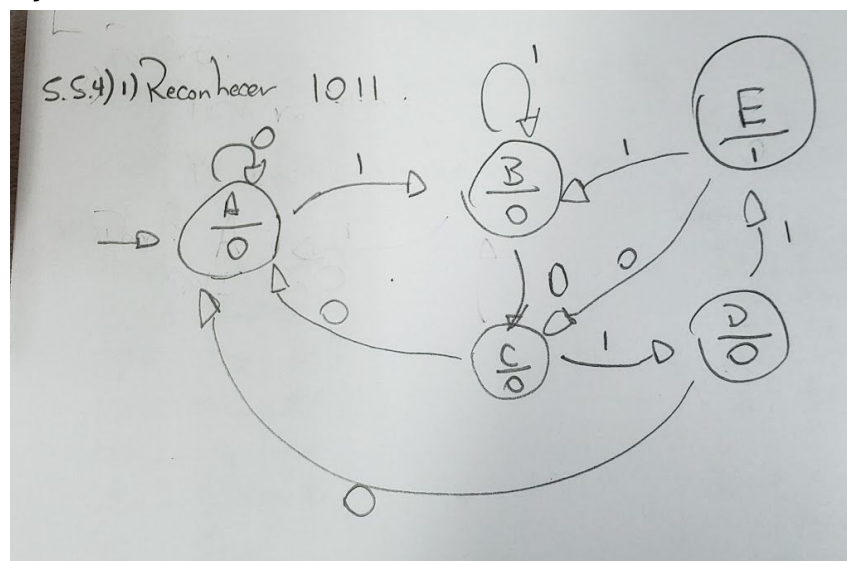
sim process 42 In: done
(gdb) run
Starting program: /home/student/src/item-5-5-3
Breakpoint 1, done () at item-5-5-3.s:22
Current language: auto; currently asm
(gdb)

```

Final da execução do programa no gdb

Exercício 5.5.4

- 1) Esta atividade usaremos como base a seguinte máquina de estado construída na preparação do exercício:



Com isso, temos o seguinte código assembly para reconhecimento da sequência solicitada.

```

.text
.globl main

main:
    LDR r1, =0xB6 @r1 Initial input X

```

```
LDR r2, =0 @r2 valor de saida
MOV r3, r1 @r3 auxiliar com valor de r1
LDR r8, =0 @r8 valor de Y
LDR r5, =32 @Auxiliar de contagem
B EstadoA
```

EstadoA:

```
CMP r5, #0
BEQ end
MOVS r4, r3, LSR #31 @r4 vai ser o bit que vou analisar
MOV r3, r3, LSL #1 @atualizo valor de r3
MOV r2, r2, LSL #1 @shift left the value of Y
SUB r5, r5, #1
BEQ EstadoA @se r4 for zero
BNE EstadoB @se r4 nao for um
```

EstadoB:

```
CMP r5, #0
BEQ end
MOVS r4, r3, LSR #31 @ pego o segundo mais significativo
MOV r3, r3, LSL #1 @atualizo valor de r3
MOV r2, r2, LSL #1 @shift left the value of Y
SUB r5, r5, #1
BEQ EstadoC @se r4 for zero
BNE EstadoB @se r4 nao for um
```

EstadoC:

```
CMP r5, #0
BEQ end
MOVS r4, r3, LSR #31 @ pego o segundo mais significativo
MOV r3, r3, LSL #1 @atualizo valor de r3
MOV r2, r2, LSL #1 @shift left the value of Y
SUB r5, r5, #1
BEQ EstadoA @se r4 for zero
BNE EstadoD @se r4 nao for um
```

EstadoD:

```
CMP r5, #0
BEQ end
MOVS r4, r3, LSR #31 @ pego o segundo mais significativo
MOV r3, r3, LSL #1 @atualizo valor de r3
MOV r2, r2, LSL #1 @shift left the value of Y
SUB r5, r5, #1
BEQ EstadoA @se r4 for zero
BNE EstadoE @se r4 for um
```

```

EstadoE:
    CMP r5, #0
    BEQ end
    MOVS r4, r3, LSR #31 @ pego o segundo mais significativo
    MOV r3, r3, LSL #1 @atualizo valor de r3
    ADD r2, r2, #1 @add 1
    MOV r2, r2, LSL #1 @shift left the value of Y
    SUB r5, r5, #1
    BEQ EstadoC @se r4 for zero
    BNE EstadoB @se r4 nao for um

end:
    SWI 0x0

```

No código acima realizamos para o valor de exemplo 0xB6(0010110110) cuja a saída tem que ser 0x12(00010010) como podemos observar na saída do registrador da imagem abaixo:

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1: sudo
Register group: general
r0 0x1 1 r1 0xb6 182 r2 0x12 18 r3 0xaa64 43620
r4 0x1 1 r5 0x1ffff8 2697144 r6 0x0 0 r7 0x0 0
r8 0x0 0 r9 0x0 0 r10 0x200100 2697400 r11 0x0 0
r12 0x1ffffc 2697100 sp 0x1ffff8 2697144 lr 0x81fc 33276 r15 0x82d4 33492
fps 0x0 0 cpsr 0x60000013 1610612755

item-5-5-4.s
60 BEQ EstadoC @se r4 for zero
61 BNE EstadoB @se r4 nao for um
62
63 end:
64 SWI 0x0
B+>
65
66
67
68
69

sim process 42 In: end
Breakpoint 2 at 0x82d4: file item-5-5-4.s, line 64.
(gdb) run
Starting program: /home/student/src/LabMicro/Lab5/fsm

Breakpoint 1, main () at item-5-5-4.s:5
Current language: auto; currently asm
(gdb) continue
Continuing.

Breakpoint 2, end () at item-5-5-4.s:64
(gdb)

```

- 2) Este exercício não conseguimos terminar a tempo. A idéia consiste em fazer uma busca por vetores. Primeiro definimos o valor de Y que será parametro e com este valor em binário definido, iremos analisar o primeiro valor bit do vetor input X e iremos analisar se os valores seguintes desse valor em análise fazem a combinação. Em seguida, iremos analisar para o segundo bit de X e assim por diante.

```

.text
.globl main

main:
    LDR r1, =0x5555AAAA @valor do input de X
    LDR r8, =0x5 @valor da sequencia Y
    LDR r9, =0x3 @r9 valor do inteiro n comprimento de Y

```

```

SUB r11, r9, #1
LDR r2, =0 @r2 valor de saida
LDR r7, =0 @auxiliar do loopIgual
MOV r3, r1 @r3 auxiliar com valor do input
MOV r4, r8 @r4 auxiliar com valor de Y
LDR r10, =31
LDR r12, =32
B loopPrincipal

loopPrincipal:
    CMP r10, #-1 @ve se X acabou
    BEQ end

    MOV r5, r4, LSR r11 @vou analisar o bit i significativo de Y
    MOV r6, r1, LSR #31 @vou analisar o bit mais significativo de X

    @Atualiza valores dos aux
    SUB r10, r10, #1
    MOV r3, r1 @r3 auxiliar com valor do input

    @Loops de comparacao
    CMP r5, r6
    BEQ loopIguais
    BNE naoIguais

naoIguais:
    LDR r7, =0
    MOV r1, r1, LSL #1
    MOV r11, r9 @ restart no indice de y
    SUB r11, r11, #1 @ restart no indice de y
    B loopPrincipal

loopIguais:
    SUB r11, r11, #1 @ passa para proximo valor de Y
    ADD r7, r7, #1 @ adiciona 1 passo do loop
    CMP r7, r9 @compara com r9
    BEQ achouSequencia
    LDR r12, =31
    SUB r12, r12, r11

```

```

    @ Atualizar valor de Y que quero analisar
    MOV r4, r9, LSL r11
    MOV r4, r4, LSR #31

    @ Atualiza valor de X que quero analisar
    MOV r3, r3, LSL r11
    MOV r6, r3, LSR #31 @vou analisar o bit mais significativo de X

    CMP r6, r4
    BEQ loopIguais
    BNE loopPrincipal

achouSequencia:
    SUBS r7, r7, #1 @diminui o passo
    MOV r2, r2, LSL #1
    BNE achouSequencia
    ADD r2, r2, #1 @adiciona 1 na saida
    MOV r2, r2, LSL #1
    MOV r11, r9 @ restart no indice de y
    SUB r11, r11, #1 @ restart no indice de y
    B loopPrincipal

end:
    SWI 0x0

```