

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



Aula 5
10 de junho de 2021

GRUPO 10

Arthur Pires da Fonseca
Bruno José Mório
Iago Soriano Roque Monteiro

NUSP: 10773096
NUSP: 10336852
NUSP: 8572921

Sumário

Exercício 5.5.1 - For Loop	3
Exercício 5.5.2 - Factorial Calculation	4
Exercício 5.5.3 - Find Maximum Value	5
Exercício 5.5.4 - Finite State Machine	5
Exercício 5.5.5 - Sequential Parity Checker	7
Apêndice	9
Exercício 5.5.1	9
Exercício 5.5.2	10
Exercício 5.5.3	11
Exercício 5.5.4	12
1	12
2	12
Exercício 5.5.5	14

Exercício 5.5.1 - For Loop

No print, é possível observar que o vetor b em 0x5000 recebeu os elementos do vetor a em 0x4000 em ordem inversa.

```
Register group: general
r0      0x8      8      r1      0x4000  16384  r2      0x5000  20480
r3      0x0      0      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  r13     0x1ffff8 2097144  lr      0x81fc  33276
pc      0x8240   33344  fps      0x0     0      cpsr    0x20000013 536870931

forLoop.s
15      ADD     r0, r0, #1    @ incrementa r0
16      BAL     loop
17
18      fin:
19      SWI     0x12345
20
21
22
23
24
25

sim process 42 In: fin                                     Line: 19   PC: 0x8240

Breakpoint 1, main () at forLoop.s:5
Current language:  auto; currently asm
(gdb) c
Continuing.

Breakpoint 2, fin () at forLoop.s:19
(gdb) x/8db 0x4000
0x4000: 0      0      0      0      42     -25    -124   -111
(gdb) x/8db 0x5000
0x5000: -111   -124   -25    42     0      0      0      0
(gdb)
```

Exercício 5.5.2 - Factorial Calculation

Observamos o resultado de $(r6)!$ em $r0$. Na 2ª imagem, $r0 = 10! = 3628800$.

```
arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0x0      0          r1      0x1ffff8 2097144
r2      0xffffffff -1        r3      0xa9c4   43460
r4      0xa      10         r5      0x1ffff8 2097144
r6      0xa      10         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      0x8224   33316
fps     0x0      0          cpsr    0x60000013 1610612755

fatorial.s
6      main:
7          ldr r0, =0x0
8
9      factorial:
10         MOV r6, #0xA          @ load 10 into r6
11
12         MOV r4, r6            @ copy n into a temp register
13
14     loop:
15         SUBS r4, r4, #1        @ decrement next multiplier
16
17
18         MULNE r6, r4, r6       @ perform multiply
```

```
arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0x375f00 3628800    r1      0x1ffff8 2097144
r2      0xffffffff -1        r3      0xa9c4   43460
r4      0x0      0          r5      0x1ffff8 2097144
r6      0x375f00 3628800    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      0x8234   33332
fps     0x0      0          cpsr    0x60000013 1610612755

fatorial.s
14     loop:
15         SUBS r4, r4, #1        @ decrement next multiplier
16
17
18         MULNE r6, r4, r6       @ perform multiply
19         MOVNE r0, r6          @ save off product for another loop
20         BNE loop             @ go again if not complete
21
22     swi 0x0
23
24
25
26
```

Exercício 5.5.3 - Find Maximum Value

Observa-se que o valor 35 foi armazenado em *r1* ao final da execução, que é o maior valor entre a lista de 13 inteiros declarada.

```

Register group: general
r0      0xc  12      r1      0x23  35      r2      0x254  33364
r3      0xaa14 43540  r4      0x23  35      r5      0xc  12
r6      0x0  0       r7      0x288  33416  r8      0x0  0
r9      0x0  0       r10     0x200100 2097408  r11     0x0  0
r12     0x1fffcc 2097100 sp      0x1ffff8 2097144  lr      0x81fc 33276
pc      0x8250  33360  fps      0x0  0       cpsr    0x60000013 1610612755

Item-5.5.3.s
16      MOVLT   r1, r4      @ se for menor, armazena r4 em r1
17      ADD     r0, r0, #1   @ incrementa contador
18      B       loop
19      end:
20      STR     r1, [r7]
21      SWI     0x12345
22      dados:
23      .word 1, 30, 2, 5, 19, 21, 25, 29, 4, 20, 32, 6, 35
24
25
26
sim process 42 In: end
Line: 21 PC: 0x8250

```

Exercício 5.5.4 - Finite State Machine

No sub exercício 1:

Para o input r1 = 0xAAAA5555 e padrão r8 = 0x0101, o resultado foi como esperado: r2 = 0x0AAA1555.

```
Register group: general
r0 0x1 1 r1 0xaaaa5555 -1431677611 r2 0xaaa1555 178918741
r3 0xa9f8 43512 r4 0x8 8 r5 0x10000000 268435456
r6 0x1c 28 r7 0x21 33 r8 0x5 5
r9 0x4 4 r10 0x200100 2097408 r11 0x0 0
r12 0x1fffc 2097100 r12 0x1ffff8 2097144 lr 0x81fc 33276
pc 0x8264 33380 fps 0x0 0 cpsr 0x20000013 536870931

fIniteState1.s
26 CMP r7, #32 @ Compara r7 e 32
27 BLS loop @ Se r7 maior que 32, itera de novo
28
29 end:
> 30 SWI 0x12345
31
32
33
34
35
36

sim process 42 In: end Line: 30 PC: 0x8264
(gdb) b 29
Breakpoint 2 at 0x8264: file fIniteState1.s, line 29.
(gdb) r
Starting program: /home/student/src/a.out
Breakpoint 1, main () at fIniteState1.s:5
Current language: auto; currently asm
(gdb) c
Continuing.
Breakpoint 2, end () at fIniteState1.s:30
(gdb)
```

No sub exercício 2:

Para o input r1 = 0x6C6D8000 e padrão r8 = 0x0110110, o resultado foi como esperado: r2 = 0x2024000.

```
Register group: general
r0 0x1 1 r1 0x6c6d8000 1819115520 r2 0x2024000 33701888
r3 0xa9f8 43512 r4 0x0 0 r5 0x2000000 33554432
r6 0x19 25 r7 0x21 33 r8 0x36 54
r9 0x7 7 r10 0x200100 2097408 r11 0x0 0
r12 0x1fffc 2097100 r12 0x1ffff8 2097144 lr 0x81fc 33276
pc 0x8264 33380 fps 0x0 0 cpsr 0x20000013 536870931

fIniteState1.s
26 CMP r7, #32 @ Compara r7 e 32
27 BLS loop @ Se r7 maior que 32, itera de novo
28
29 end:
> 30 SWI 0x12345
31
32
33
34
35
36

sim process 42 In: end Line: 30 PC: 0x8264
(gdb) b 29
Breakpoint 2 at 0x8264: file fIniteState1.s, line 29.
(gdb) r
Starting program: /home/student/src/a.out
Breakpoint 1, main () at fIniteState1.s:5
Current language: auto; currently asm
(gdb) c
Continuing.
Breakpoint 2, end () at fIniteState1.s:30
(gdb)
```

Exercício 5.5.5 - Sequential Parity Checker

A sequência a ser analisada fica no registrador *r0* e o resultado, no *r1*.

Primeiramente, uma sequência de paridade par, 0x00e7 (11100111):

```
arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0xe7      231
r2      0x20      32
r4      0x1       1
r6      0x0       0
r8      0x0       0
r10     0x200100 2097408
r12     0x1fffcc 2097100
lr      0x81fc   33276
fps     0x0       0
r1      0x0       0
r3      0x1       1
r5      0x1ffff8 2097144
r7      0x0       0
r9      0x0       0
r11     0x0       0
sp      0x1ffff8 2097144
pc      0x8228   33320
cpsr    0x60000013 1610612755

parity.s
8      mov r1, #0
9
10     mov r2, #32      @ count = 32
11     mov r3, #1       @ mask
12
13     loop:
> 14     and r4, r0, r3 @ count-esimo bit
15     mov r0, r0, lsr #1 @ tira o ultimo bit da sequencia
16
17     subs r4, r4, r3   @ tira 1 do bit
18     addeq r1, r1, #1  @ muda a paridade
19
20     subs r2, r2, #1   @ count--
```

```
arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0x0       0
r2      0x0       0
r4      0xffffffff -1
r6      0x0       0
r8      0x0       0
r10     0x200100 2097408
r12     0x1fffcc 2097100
lr      0x81fc   33276
fps     0x0       0
r1      0x0       0
r3      0x1       1
r5      0x1ffff8 2097144
r7      0x0       0
r9      0x0       0
r11     0x0       0
sp      0x1ffff8 2097144
pc      0x8244   33348
cpsr    0x60000013 1610612755

parity.s
17     subs r4, r4, r3   @ tira 1 do bit
18     addeq r1, r1, #1  @ muda a paridade
19
20     subs r2, r2, #1   @ count--
21     bne loop          @ continua se count != 0
22
23     and r1, r1, r3
24     swi 0x0
25
26
27
28
29
```

Também, o resultado com paridade ímpar 0xd (1101):

```
arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0xd      13      r1      0x0      0
r2      0x20     32      r3      0x1      1
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      0x8228   33320
fps     0x0      0       cpsr    0x60000013 1610612755

6      main:
7          ldr r0, =0b1101      @ sequencia de paridade
8          mov r1, #0
9
10         mov r2, #32          @ count = 32
11         mov r3, #1          @ mask
12
13     loop:
> 14         and r4, r0, r3      @ count-esimo bit
15         mov r0, r0, lsr #1    @ tira o ultimo bit da sequencia
16
17         subs r4, r4, r3      @ tira 1 do bit
18         addeq r1, r1, #1      @ muda a paridade

sim process 42 In: load                                     Line: 14   PC: 0x8228

arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
r0      0x0      0      r1      0x1      1
r2      0x0      0      r3      0x1      1
r4      0xffffffff -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      0x8244   33348
fps     0x0      0       cpsr    0x60000013 1610612755

parity.s
19
20         subs r2, r2, #1      @ count--
21         bne loop            @ continua se count != 0
22
23         and r1, r1, r3
B+> 24         swi 0x0
25
26
27
28
29
30
31
```


Apêndice

Exercício 5.5.1

```
@ Exercicio 5.5.1 do livro
@ Para debugar este codigo:
@ gcc forloop.s && gdb a.out
.text
.globl main

main:
    LDR    r0, =0        @ contador inicia em 0
    LDR    r1, =0x4000    @ r1 - array a
    LDR    r2, =0x5000    @ r2 - array b

loop:
    CMP    r0, #7        @ compara r0 com 7
    BHI    fim           @ se for maior, acaba o loop
    RSB    r3, r0, #7     @ r3 = 7 - i
    LDRB   r4, [r1, r3]   @ carrega b[r3] ou seja b[7-1]
    STRB   r4, [r2, r0]   @ armazena em a[r0] ou seja a[i]
    ADD    r0, r0, #1     @ incrementa r0
    BAL    loop

fim:
    SWI    0x12345
```

Exercício 5.5.2

```
@ Exercicio 5.5.2 do livro
@ Para debugar este codigo:
@ gcc fatorial.s && gdb a.out
    .text
    .globl main
main:
    ldr r0, =0x0

factorial:
    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 r6, r4, r6    @ perform multiply
    MOVNE r0, r6        @ save off product for another loop
    BNE loop            @ go again if not complete

    swi 0x0
```

Exercício 5.5.3

```
@ Exercicio 5.5.3 do livro
@ Para debugar este codigo:
@ gcc maxval.s && gdb a.out
.text
.globl main

main:
    ADR    r2, dados    @ carrega array de inteiros em r2
    MOV    r5, #13      @ salva tamanho em r5
    SUB    r5, r5, #1    @ pega valor do ultimo indice com tamanho -1
    MOV    r7, r2        @ guarga endereço em r7
    LDR    r1, [r7], #4   @ r1 salva o maior elemento, e inicia com o primeiro
    MOV    r0, #0        @ contador começa em 0
loop:
    CMP    r0, r5        @ compara contador com valor do ultimo indice
    BGE    end           @ se maior, sai
    LDR    r4, [r7], #4   @ pega o próximo elemento de r7 e guarga em r4
    CMP    r1, r4        @ compara r1 com r4
    MOVLT  r1, r4        @ se for menor, armazena r4 em r1
    ADD    r0, r0, #1     @ incrmenta contador
    B      loop
end:
    STR    r1, [r7]
    SWI    0x12345
dados:
    .word 1, 30, 2, 5, 19, 21, 25, 29, 4, 20, 32, 6, 35
```

Exercício 5.5.4

1

```
@ Exercicio 5.5.4.1 do livro
@ Para debugar este codigo:
@ gcc finiteState1.s && gdb a.out
    .text
    .globl main

main:
    MOV r8, #0x5      @ = 0101
    MOV r9, #0x4      @ Tamanho
    LDR r1, =0xAAA5555 @ = 101010101010101001010101010101
    MOV r2, #0x0      @ 0 inicial, espera-se
00001010101010101000010101010101 ou AAA1555 na saída
    MOV r3, r1

begin:
    MOV r5, #0x1      @ r5 = vetor de adicao
    MOV r6, #32       @ r6 = 32
    SUB r6, r6, r9     @ r6 = 32 - r9
    MOV r7, #0x1      @ r7 = contador
    MOV r2, #0x0      @ r2 = resultado
    MOV r5, r5, ROR r9 @ Rotaciona r5 pelo tamanho de Y

loop:
    MOV r4, r3, LSR r6 @ Shifta r3 por r6 vezes e guarda em r4
    CMP r4, r8        @ Compara r4 com r8
    ADDEQ r2, r2, r5   @ Se r4 e r8 iguais, r5 + r2
    ADD r7, r7, #1     @ r7++
    MOV r5, r5, ROR #1 @ Rotaciona o vetor de adicao
    MOV r3, r3, LSL #1 @ Shifta X
    CMP r7, #32       @ Compara r7 e 32
    BLS loop          @ Se r7 maior que 32, itera de novo

end:
    SWI 0x12345
```

2

```
@ Exercicio 5.5.4.2 do livro
@ Para debugar este codigo:
@ gcc finiteState2.s && gdb a.out
    .text
    .globl main

main:
    MOV r8, #0x36      @ padrao procurado 0110110
    MOV r9, #0x7       @ Tamanho do padrao
```

```
LDR r1, =0x6c6d8000    @ palavra onde será feita a busca
MOV r2, #0x0           @ registrador resultado
MOV r3, r1              @ auxiliar da palavra
```

begin:

```
MOV r5, #0x1           @ r5 flag de match
MOV r6, #32             @ r6 shift maximo da flag de match
SUB r6, r6, r9           @ r6 recebe 32 menos o tamanho do padrao
MOV r7, #0x1           @ r7 = contador
MOV r5, r5, ROR r9       @ Rotaciona flag de match pelo tamanho de Y ->
equivalente a shiftar para alinhar o padrao ao comeco da palavra
```

loop:

```
MOV r4, r3, LSR r6      @ Guarda em r4 o pedaço da palavra a ser
comparado
```

```
CMP r4, r8              @ Compara r4 com r8 (pedaço da palavra com padrao)
ADDEQ r2, r2, r5         @ Se o padrão corresponder ao pedaço, adiciona
adiciona flag de match ao resultado
```

```
MOV r5, r5, ROR #1      @ Rotaciona a flag de match em 1 bit p direita
MOV r3, r3, LSL #1       @ Shifta palavra em 1 bit pra esquerda
```

```
ADD r7, r7, #1          @ contador++
CMP r7, #32             @ Compara contador e 32
BLS loop                @ Se contador maior que 32, itera de novo
```

end:

```
SWI 0x12345
```

Exercício 5.5.5

```
@ Exercicio 5.5.5 do livro
@ Para debugar este codigo:
@ gcc parity.s && gdb a.out
.text
.globl main
main:
    ldr r0, =0b11100111 @ sequencia de paridade
    mov r1, #0

    mov r2, #32          @ count = 32
    mov r3, #1          @ mask

loop:
    and r4, r0, r3        @ count-esimo bit
    mov r0, r0, lsr #1    @ tira o ultimo bit da sequencia

    subs r4, r4, r3        @ tira 1 do bit
    addeq r1, r1, #1      @ muda a paridade

    subs r2, r2, #1        @ count--
    bne loop              @ continua se count != 0

    and r1, r1, r3
    swi 0x0
```