ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO



Experiência 3

Relatório

Turma: 02

11/04/2023

Nome	Número USP
Natanael Magalhães Cardoso	8914122
Tomas Gorescu Caldeira	9300881

Enunciado: Assume that a signed long multiplication instruction is not available. Write a program that performs long multiplications, producing 64 bits of result. Use only the UMULL instruction and logical operations such as MVN to invert, XOR, and ORR. Run the program using the two operands –2 and –4 to verify.

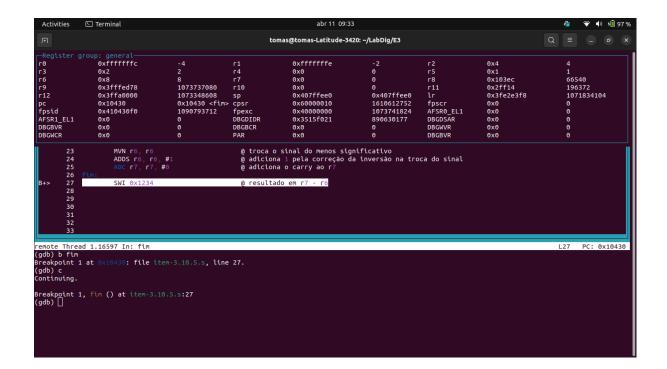
O programa abaixo faz a multiplicação com sinal de duas palavras de 64-bits e armazena o resultado nos registradores R7 (mais significativo) e R6 (menos significativo)

A ideia é fazer a divisão longa sem sinal, verificar e corrigir o sinal posteriormente.

```
text
globl main
main:
 LDR r0, = -4 @ operando 1
 LDR r1, = -2
```

```
preparingRegisters:
 MOV r4, #0 @ r4 recebe 0
 MOVS r2, r0 @ r2 recebe operando 1 com atualização de
 RSBMI r2, r0, #0 @ r2 = modulo(operando 1) - (r2 = 0 - r2 se r2
é negativo)
 MOVMI r4, #1 @ se r2 é negativo marca r4 como 1
 MOVS r3, r1
 EORMI r4, r4, #1 @ se operando 2 também é negativo, r4 fica 0
 RSBMI r3, r1, #0 @ r3 = modulo(operando 2) -> (r3 = 0 - r3 \text{ se } r3)
é negativo)
multiplication:
 UMULL r6, r7, r2, r3 @ multiplicação de r2 por r3
 CMP r4, #0
 BNE invert @ se r4 != 0
 B fim
invert:
 MVN r7, r7 @ troca o sinal do mais significativo
 MVN r6, r6
 ADDS r6, r6, #1
 ADC r7, r7, #0 @ adiciona o carry ao r7
fim:
 SWI 0x1234
```

Abaixo a imagem da depuração do programa implementado

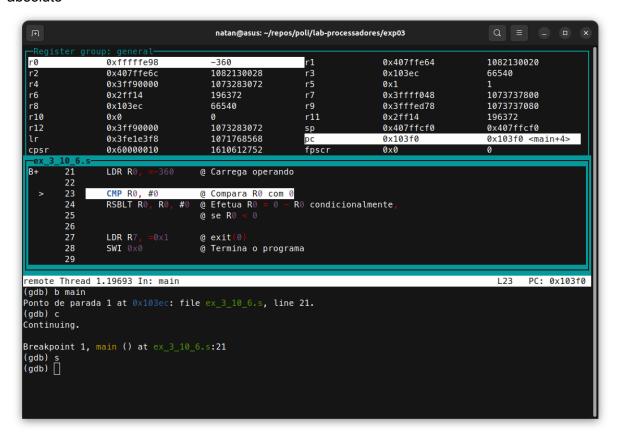


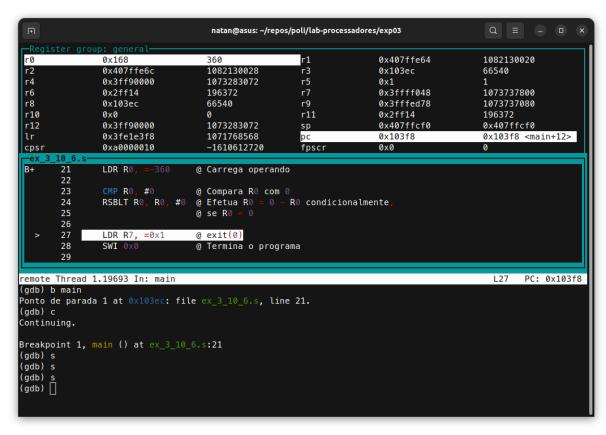
Enunciado: Write ARM assembly to perform the function of absolute value. Register r0 contains the initial value, and r1 contains the absolute value. Try to use only two instructions, not counting the SWI to terminate the program.

Este programa compara o valor armazenado no registrador R0 e efetua a operação R0 = 0 - R0 apenas se o valor for menor que zero.

```
global main
main:
  LDR R0, =-360 @ Carrega operando
  CMP R0, #0 @ Compara R0 com 0
  RSBLT R0, R0, #0 @ Efetua R0 = 0 - R0 condicionalmente,
  LDR R7, =0x1
  SWI 0x0
```

As Figuras abaixo mostram o valor do registrador R0 antes e depois da operação valor absoluto





Enunciado: Write ARM assembly to perform the function of division. Registers r1 and r2 contain the dividend and divisor, r3 contains the quotient, and r5 contains the remainder. For this operation, you can either use a single shift-subtract algorithm or another more complicated one.

O programa abaixo faz a divisão de um número armazenado no registrador R1 por outro armazenado no registrador R2 e coloca o quociente no registrador R3 e o resto no registrador R5.

O programa funciona descolando o divisor para esquerda até que seu valor seja o máximo possível menor que o dividendo. Então, começa o loop de multiplicação com tantas iterações quantas ocorreram na etapa de alinhamento.

A cada etapa, verifica-se se o dividendo é maior que 2ⁿ * divisor , sendo n o número de shifts a esquerda. Se o dividendo for maior, subtrai-se 2ⁿ divisor do dividendo, adiciona-se 2ⁿ ao quociente e volta-se a decrementar n, até que n seja 0.

O que resta das consecutivas subtrações no dividendo é o resto, é o quociente é o resultado das somas ao longo da iteração.

```
.globl main
main:
  LDR r1, =8914122 @ inicializa dividendo
  LDR r2, =1000 @ inicializa divisor
  MOV r3, #0
  MOV r5, #0
                      @ inicializa resto
  MOV r4, r2
  MOV r9, r1 @ copia do dividendo
  MOV r7, #0
                      @ inicializa contador
  MOV r6, #1
alinhamento:
  MOV r4, r4, LSL #1 @ divisor shiftado para esquerda
  CMP r4, r9
dividendo
  BLT alinhamento @ volta para mais um shift left
  MOV r4, r4, LSR #1 @ garantia que ao final do loop da pra
dividir um pelo outro
  SUB r7, r7, #1 @ decrementa o contador após o shift right
loop:
  CMP r9, r4
  SUBGE r9, r9, r4
  ADDGE r3, r3, r6, LSL r7 @ quociente recebe resto 1 shiftado do
numero de alinhamentos
  SUB r7, r7, #1
  MOV r4, r4, LSR #1
  CMP r7, #0
  BPL loop
  MOV r5, r9
fim:
  SWI 0x0
```

As Figuras abaixo mostram a depuração do programa para as três operações indicadas.

```
natan@asus: ~/repos/poli/lab-processadores/exp03
                                                                                                    Q =
                                                                                                   1234567
 r0
                 0×1
                                                            r1
                                                                             0x12d687
                  0x4d2
                                        1234
                                                            r3
                                                                             0x3e8
                                                                                                   1000
  r4
                  0x269
                                        617
                                                            r5
                                                                             0x237
                                                                                                   567
                                                            r7
                                                                             0xffffffff
                  0×1
                                                                                                   -1
                  0x103ec
                                        66540
                                                                             0x237
                                                                                                   567
 r10
                  0x0
                                                            r11
                                                                             0x2ff14
                                                                                                   196372
 r12
                  0x3ff90000
                                        1073283072
                                                                             0x407ffcf0
                                                                                                   0x407ffcf0
                                                            sp
                  0x3fe1e3f8
                                        1071768568
                                                                                                   0x10444 <fim>
 cpsr
                 0xa0000010
                                        -1610612720
                                                            fpscr
                                                                            0x0
                 CMP r7, #0
BPL loop
         44
                                              @ compara contador com
         45
                                                              @ continua se positivo ou \emptyset
         46
                                                            @ r5 recebe o resto
         47
        48
 B+>
                  SWI 0x0
                                              @ fim do programa
        49
50
         52
remote Thread 1.23727 In: fim
                                                                                                     L48 PC: 0x10444
(gdb) c
Continuing.
Breakpoint 1, main () at ex_3_10_7.s:23
(gdb) b fim Ponto de parada 2 at 0x10444: file ex_3_10_7.s, line 48.
(gdb) c
Continuing.
Breakpoint 2, fim () at ex_3_10_7.s:48
(gdb)
```

1234567/1000 = (1234*1000) + 567

```
Q =
                                        natan@asus: ~/repos/poli/lab-processadores/exp03
                                                                           0x75bcd15
                                                                                                 123456789
 r0
                 0x1
                                                          r1
                 0x4d2
                                       1234
                                                                           0x186ce
                                                                                                 100046
  r4
                 0x269
                                       617
                                                           r5
                                                                           0x19
                                                                                                 25
 r6
                 0x1
                                                           r7
                                                                           0xffffffff
                                                                                                 -1
 r8
r10
                                       66540
                 0x103ec
                                                           r9
                                                                           0x19
                                                                                                 25
                 0x0
                                       0
                                                           r11
                                                                           0x2ff14
                                                                                                 196372
                 0x3ff90000
                                       1073283072
                                                           sp
                                                                           0x407ffcf0
                                                                                                 0x407ffcf0
                                       1071768568
                 0x3fe1e3f8
                                                           рc
                                                                          0×10444
                                                                                                0x10444 <fim>
 cpsr
                 0xa0000010
                                       -1610612720
                                                          fpscr
                                                                           0x0
        44
                                             @ compara contador com 0
        45
                 BPL loop
                                                            @ continua se positivo ou 0
        46
                                                          @ r5 recebe o resto
                 SWI 0x0
                                             @ fim do programa
        49
        50
        53
remote Thread 1.23952 In: fim
                                                                                                   L48 PC: 0×10444
(gdb) c
Continuing.
Breakpoint 1, main () at ex_3_10_7.s:23
(gdb) b fim
Ponto de parada 2 at 0 \times 10444: file ex_3_10_7.s, line 48.
(gdb) c
Continuing.
Breakp<u>o</u>int 2, fim () at ex_3_10_7.s:48
```

123456789 = (100046*1234) + 25

```
natan@asus: ~/repos/poli/lab-processadores/exp03
                  0x1
                                                              r1
                                                                                0x64
 r0
                                                                                                        100
 r2
                  0xa
                                                               r3
                                                                                                        10
                                          10
                                                                                0xa
 r4
                                          5
                  0x5
                                                               r5
                                                                                0x0
                                                                                                        0
 r6
                                                               r7
                   0x1
                                                                                0xffffffff
                                                                                                        -1
 r8
r10
r12
                   0x103ec
                                          66540
                                                                                0x0
                                                                                                        0
                                                               r9
                   0x0
                                                               r11
                                                                                0x2ff14
                   0x3ff90000
                                          1073283072
                                                               sp
                                                                                0x407ffcf0
                                                                                                        0x407ffcf0
                  0x3fe1e3f8
                                          1071768568
                                                                                                        0x10444 <fim>
 cpsr
                                                              fpscr
                                                                                0x0
         44
                  CMP r7, #0
BPL loop
MOV r5, r9
                                                 @ compara contador com \emptyset
         45
46
47
48
49
50
51
52
                                                                 @ continua se positivo ou \emptyset
                                                              @ r5 recebe o resto
                                                @ fim do programa
                SWI 0x0
 B+>
remote Thread 1.24073 In: fim
                                                                                                          L48 PC: 0x10444
(gdb) c
Continuing.
Breakpoint 1, main () at ex_3_10_7.s:23
(gdb) b fim
Ponto de parada 2 at 0x10444: file ex_3_10_7.s, line 48. (gdb) c
Continuing.
Breakpoint 2, fim () at ex_3_10_7.s:48 (gdb)
```

100 = 10 * 10 + 0

Enunciado: A Gray code is an ordering of 2 n binary numbers such that only one bit changes from one entry to the next. One example of a 2-bit Gray code is b10 11 01 00. The spaces in this example are for readability. Write ARM assembly to turn a 2-bit Gray code held in r1 into a 3-bit Gray code in r2.

Para a solução deste exercício, usamos o algoritmo demonstrado em aula, que difere um pouco do utilizado no livro texto. Para um código de gray de n bits, podemos obter o código de gray de n+1 fazendo:

- Duplicamos a sequência, invertendo a ordem
- Na primeira parte, adicionamos um 0 como pré-fixo
- Na segunda parte, adicionamos um 1 como pré fixo

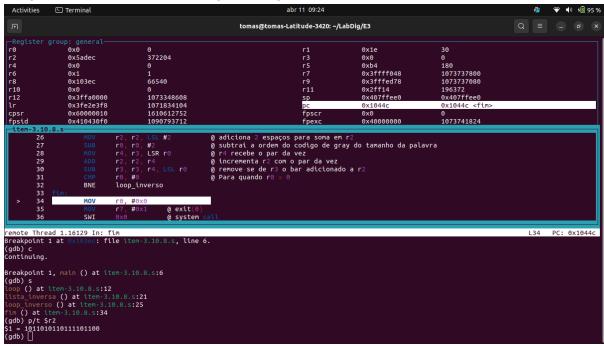
A sequência dos passos para gray de 2 para 3 bits, ficaria:

- Partimos da sequência 00 01 11 10
- Duplicamos a sequência: 00 01 11 10 || 10 11 01 00
- Adicionamos os 0's na primeira sequência: 000 001 011 010 || 10 11 01 00
- Adicionamos os 1's na segunda sequência: 000 001 011 010 || 110 111 101 100

No código abaixo seguimos o algoritmo descrito. Nós usamos os Shift right para isolar cada um dos segmentos da sequência de 2 bits. Também é importante ressaltar que para poder isolar os fragmentos que não são "extremos", precisamos ter um cópia de r1 da qual vamos subtraindo os pares pelos quais já passamos (r3).

```
MOV r3, r1
  MOV r2, #0
  MOV r6, #1
loop:
  SUB r0, r0, #2
tamanho da palavra
  MOV r4, r3, LSR r0 @ r4 recebe o par da vez
  ADD r2, r2, r4 @ incrementa r2 com o par da vez
  SUB r3, r4, LSL r0 @ remove-se de r3 o bar adicionado a r2
  MOV r2, r2, LSL #3
espaço p/ soma
  CMP r0, #0
  BNE loop
  MOV r2, r2, LSR #3 @ Remove os bits adicionados a mais
lista_inversa:
  LDR r5, =0b10110100 @ r1 recebe código inverso de r1
transformado
  MOV r0, #8 @ r0 recebe o tamanho da palavra de r1
  MOV r3, r5 @ move r5 para registrador de trabalho
loop_inverso:
  ADD r2, r6, r2, LSL #1 @ adiciona um 1 a direita em r2
  MOV r2, r2, LSL #2 @ adiciona 2 espaços para soma em r2
  SUB r0, r0, #2
  MOV r4, r3, LSR r0 @ r4 recebe o par da vez
  SUB r3, r3, r4, LSL r0 @ remove-se de r3 o bar adicionado a r2
  CMP r0, #0
  BNE loop_inverso @ continua loop se r0 != 0
fim:
  MOV r7, #0x1
```

A Figura abaixo mostra a depuração do algoritmo implementado



Como o comando p/t não mostra os bits da direita, temos que saber que a sequência é "liderada" por 0's, formando:

000 001 011 010 110 111 101 100

que é o código de Gray de 3 bits que esperávamos formar.