**Carlos Eduardo Sena Rebouças Garcia 9345255**

**Diego Boccoli Gallego 9344744**

**Gabriel Rocha Amorim 9395112**

**3.10.5 Signed multiplication**

**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.**

MOV R0, #0xFFFFFFFE ;CARREGA -2

MOV R1, #0xFFFFFFFC ;CARREGA -4

AND R2, R0, # ;RECUPERA 1 BIT

CMP R0,#0 ;VERIFICA SE É NEGATIVO

RSBMI R0, R0, #0 ;OBTÉM VALOR ABSOLUTO

AND R3, R1, =0X80000000 ;MESMO PARA R1

CMP R1,#0

RSBMI R1, R1, #0

UMULL R5,R4,R0,R1 ; MULTIPLICA

EOR R2,R2,R3 ;DECIDE SINAL

CMN R2, #0

MVNMI R4, R4 ; INVERTE BITS MAIS SIGNIFICATIVOS

MVNMI R5, R5 ; INVERTE BITS MENOS SIGNIFICATIVOS

ADDMI R5, R5, #1 ; ADICIONA 1

**3.10.6 Absolute value**

**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.**

CMP R0, #0

RSBMI R1, R0, #0

SWI 0x80

**3.10.7 Division**

**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.**

main: ldr r1, =1023

ldr r2, =10

mov r3, #0

mov r5, r1

mov r0, #1

and r8, r1, #0x80000000

cmp r1,#0

rsbmi r1,r1,#0

and r9, r2, #0x80000000

cmp r2,#0

rsbmi r2,r2,#0

loop1: cmp r5, r2

bmi fim

mov r4, r2

mov r7, #0

loop2: mov r6, r4, lsl#1

cmp r5, r6

bmi out

mov r4, r4, lsl #1

add r7, r7, #1

b loop2

out: add r3, r3, r0, lsl r7

sub r5, r5, r4

b loop1

fim: eors r8,r8,r9

mvnmi r3,r3

addmi r3,r3,#1

swi 0x123456

**3.10.8 Gray codes**

**A Gray code is an ordering of 2n 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.**

**Note**

**The 2-bit Gray code occupies only bits [7:0] of r1, and the 3-bit Gray code occupies only bits [23:0] of r2. You can ignore the leading zeros.**

**One way to build an n-bit Gray code from an (n – 1)-bit Gray code is to prefix every (n – 1)-bit element of the code with 0. Then create the additional n-bit Gray code elements by reversing each (n – 1)-bit Gray code element and prefixing it with a one. For example, the 2-bit Gray code above becomes b010 011 001 000 101 111 110 100.**