/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* \file main.c

\* \brief LAB EXERCISE 5.2 - SQUARE ROOT APPROXIMATION

\*

\* Write an assembly code subroutine to approximate the square root of an

\* argument using the bisection method. All math is done with integers, so the

\* resulting square root will also be an integer

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* GOOD LUCK!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "stdint.h"

/\*\* @cond DO\_NOT\_INCLUDE\_WITH\_DOXYGEN \*/

#include "string.h"

/\*\* @endcond \*/

/\*\*

\* @brief 5863 Write a brief description of the function here

\*

\* Detailed description of the function after one line gap

\*

\* @note You can also add a note this way

\*

\* @param[in]

\* You can also start writing on a new line.

\* And continue on the next line. This is how you describe an input parameter

\*

\* @return

\* This is how you describe the return value

\*

\*/

\_\_asm int my\_sqrt(int x){

//R6 = x

//R1 = a = 0

//R2 = b = square root of largest possible argument (e.g. ~216).

//R3 = c = -1

//R5 = c\_old

MOV R6, R0

MOVS r1, #0

MOVS r3, #0XFF

MOVS r2, #255

MOVS r5, #0

//do {

loop MOV R5,R3

// c\_old <- c

// c <- (a+b)/2

MOV R0,R1

ADD R1,R1,R2

LSRS R1,R1,#1

MOV R3,R1

MOV R1,R0

// if (c\*c == x) {

MOV R0,R3

MULS R3,R3,R3

MOV R7,R3

MOV R3,R0

CMP R7,R6

// done = 1

BEQ exit\_loop

// } else if (c\*c < x) {

CMP R7,R6

BGT GT\_LABEL

// a <- c

MOV R1,R3

B COMP\_DONE

// } else {

// b <- c

GT\_LABEL MOV R2,R3

// }

//} while (!done) && (c != c\_old)

COMP\_DONE CMP R5,R3

BNE loop

//return c

exit\_loop BX LR

}

\_\_asm int mul(int a, int b) {

MOVS R2, #0

MOVS R3, #1

loop2 CMP R1, #0

BEQ exit\_loop1

ANDS R1, R3, R1

CMP R1, #0

BEQ loop1

ADD R2, R2, R0

loop1

LSLS R0,R0,#1

LSRS R1,R1,#1

B loop2

exit\_loop1 BX LR

}

\_\_asm void q4(void) {

MOVS R4, #7

MOVS R5, #4

MOVS R6, #4

again MOV R7, R4

ADD R4, R5, R4

MOV R5, R7

SUBS R6, R6, #1

BNE again

BX LR

}

/\*----------------------------------------------------------------------------

MAIN function

\*----------------------------------------------------------------------------\*/

/\*\*

\* @brief Main function

\*

\* Detailed description of the main

\*/

int main(void){

q4();

int r1 = 2;

int r0 = 4;

int r2 = mul(r0, r1);

volatile int r, a, b, c, d, j=0;

int i;

r = my\_sqrt(0); // should be 0

r = my\_sqrt(25); // should be 5

r = my\_sqrt(133); // should be 11

a = my\_sqrt(2);

b = my\_sqrt(4);

c = my\_sqrt(22);

d = my\_sqrt(121);

for (i=0; i<10000; i++){

r = my\_sqrt(i);

j+=r;

}

while(1)

;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ARM University Program Copyright © ARM Ltd 2014\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/