Sovann Chak & Michael Starks

Project 1 Module 1

ECEN5803 – 002

Module 1 – To C or Not To C

1.

ASM Project (Lab_Exercise_1) program size:

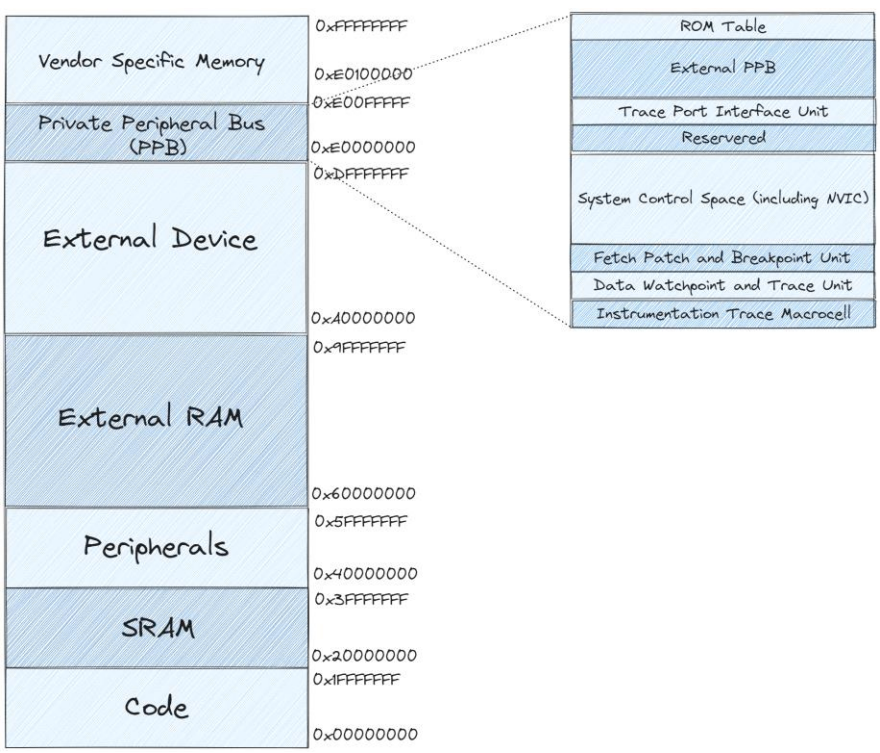
Program Size: Code=3200 RO-data=452 RW-data=24 ZI-data=120

M1String (C functions – code is in Appendix A) program size:

Program Size: Code=3216 RO-data=452 RW-data=24 ZI-data=120

The assembly language functions (Lab_Exercise_1) used less memory than the program with functions implemented in the C programming language.

2. The M1String memory map snippet from .map file can be found in the Appendix of this document. The following diagram is an image of the Cortex-M4 Memory Map.



The code block (0x0000_0000 – 0x1FFF_FFFF) at the bottom of the diagram is the code memory in which we store the program code. The memory map begins at the Reset Section which is

0x0800_0000, the reset vector points to register 0x0800_0215 which stores a program startup_stm32f401xe.o. The code block can be used for data memory, but primarily the SRAM Region block is where data is stored.

The SRAM block (0x2000_0000 – 0x3FFF_FFFF), second from the bottom of the diagram, is SRAM or SDRAM which is primarily used to store data and variables that can change during execution time. In the .map file you can see these variables:

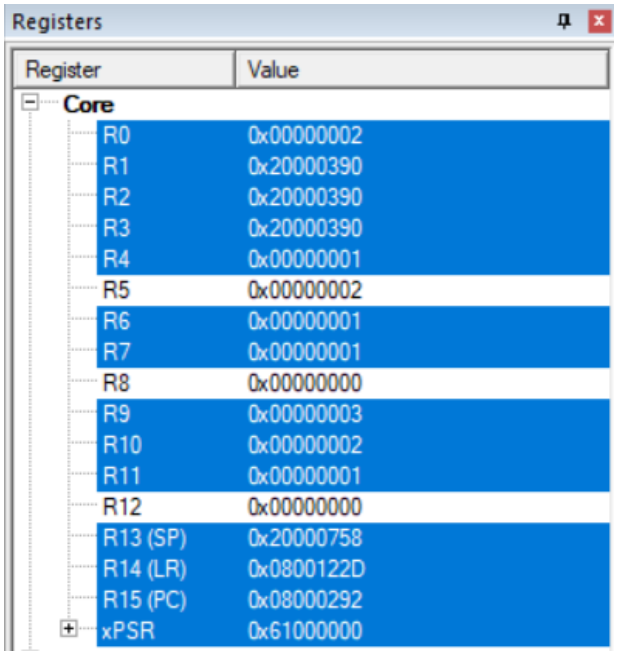
```
PreviousVal                    0x20000194   Data          4  hal_tick.o(.data)
SystemCoreClock                0x2000019c   Data          4  system_stm32f4xx.o(.data)
TimMasterHandle                0x200001ac   Data         60  hal_tick.o(.bss)
```

The startup_stm32f401xe.o program declares the stack/heap areas, it initializes the vector table at 0x0800_0000, contains the reset handler, and sets up definitions for IRQHandlers for the device specific peripherials (DMA, RTC, ETC). The peripherial region (0x4000_0000 – 0x5FFF_FFFF) is primarily used for these device peripherials and on-chip peripherials too.

```
Reset_Handler                  0x08000215   Thumb Code    8  startup_stm32f401xe.o(.text)
NMI_Handler                    0x0800021d   Thumb Code    2  startup_stm32f401xe.o(.text)
HardFault_Handler              0x0800021f   Thumb Code    2  startup_stm32f401xe.o(.text)
MemManage_Handler              0x08000221   Thumb Code    2  startup_stm32f401xe.o(.text)
BusFault_Handler               0x08000223   Thumb Code    2  startup_stm32f401xe.o(.text)
UsageFault_Handler             0x08000225   Thumb Code    2  startup_stm32f401xe.o(.text)
SVC_Handler                    0x08000227   Thumb Code    2  startup_stm32f401xe.o(.text)
DebugMon_Handler               0x08000229   Thumb Code    2  startup_stm32f401xe.o(.text)
PendSV_Handler                 0x0800022b   Thumb Code    2  startup_stm32f401xe.o(.text)
SysTick_Handler                0x0800022d   Thumb Code    2  startup_stm32f401xe.o(.text)
ADC_IRQHandler                 0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA1_Stream0_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA1_Stream1_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA1_Stream2_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA1_Stream3_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA1_Stream4_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA1_Stream5_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA1_Stream6_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA1_Stream7_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA2_Stream0_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA2_Stream1_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA2_Stream2_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA2_Stream3_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA2_Stream4_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA2_Stream5_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA2_Stream6_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
DMA2_Stream7_IRQHandler        0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
EXTI0_IRQHandler               0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
EXTI15_10_IRQHandler           0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
EXTI1_IRQHandler               0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
EXTI2_IRQHandler               0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
EXTI3_IRQHandler               0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
EXTI4_IRQHandler               0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
EXTI9_5_IRQHandler             0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
FLASH_IRQHandler               0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
FPU_IRQHandler                 0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
I2C1_ER_IRQHandler             0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
I2C1_EV_IRQHandler             0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
I2C2_ER_IRQHandler             0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
I2C2_EV_IRQHandler             0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
I2C3_ER_IRQHandler             0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
I2C3_EV_IRQHandler             0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
OTG_FS_IRQHandler              0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
OTG_FS_WKUP_IRQHandler         0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
PVD_IRQHandler                 0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
RCC_IRQHandler                 0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
RTC_Alarm_IRQHandler           0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
RTC_WKUP_IRQHandler            0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
SDIO_IRQHandler                0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
SPI1_IRQHandler                0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
SPI2_IRQHandler                0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
SPI3_IRQHandler                0x0800022f   Thumb Code    0  startup_stm32f401xe.o(.text)
```

After the startup_stm32f401 program runs, the main program is run from code memory (0x0800_01b0).

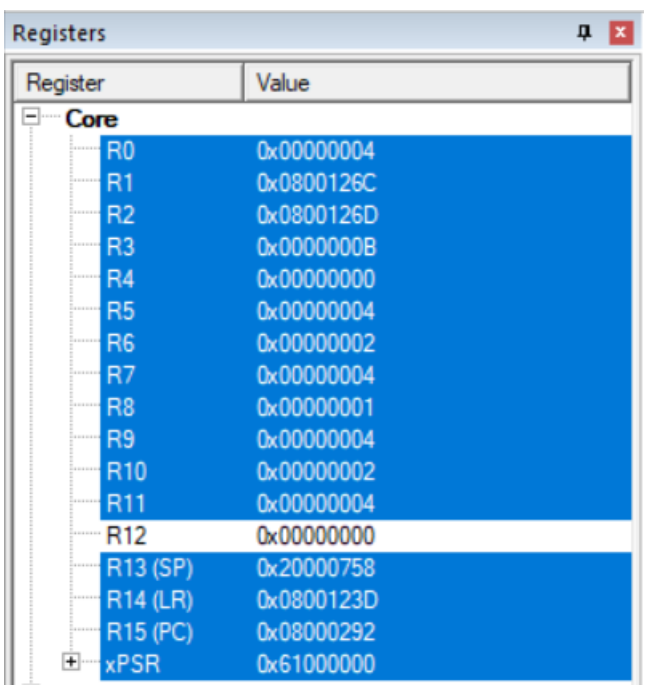
3. Testing sqrt function: Argument is passed in through R0 and output is seen in R6 before being moved to R0 to be returned from function
    a. Sqrt(2) = ~1:



Registers

Register	Value
Core	
R0	0x00000002
R1	0x20000390
R2	0x20000390
R3	0x20000390
R4	0x00000001
R5	0x00000002
R6	0x00000001
R7	0x00000001
R8	0x00000000
R9	0x00000003
R10	0x00000002
R11	0x00000001
R12	0x00000000
R13 (SP)	0x20000758
R14 (LR)	0x0800122D
R15 (PC)	0x08000292
xPSR	0x61000000

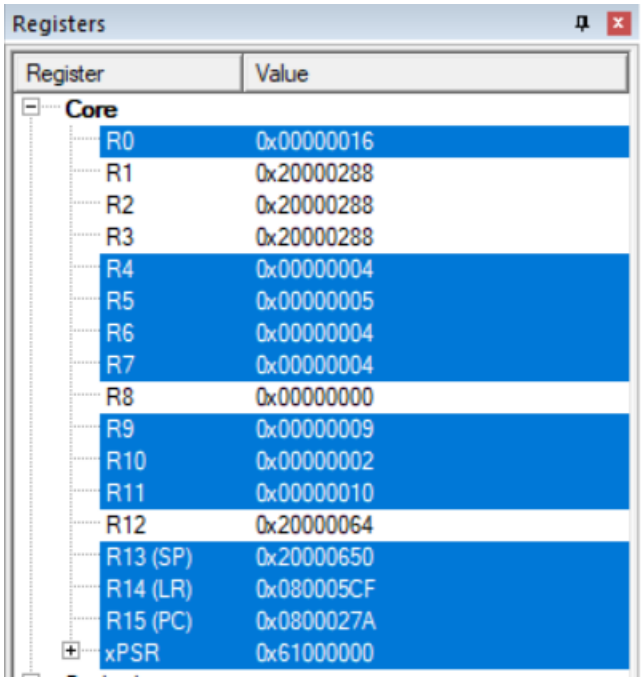
    b. Sqrt(4) = 2:



Registers

Register	Value
Core	
R0	0x00000004
R1	0x0800126C
R2	0x0800126D
R3	0x0000000B
R4	0x00000000
R5	0x00000004
R6	0x00000002
R7	0x00000004
R8	0x00000001
R9	0x00000004
R10	0x00000002
R11	0x00000004
R12	0x00000000
R13 (SP)	0x20000758
R14 (LR)	0x0800123D
R15 (PC)	0x08000292
xPSR	0x61000000

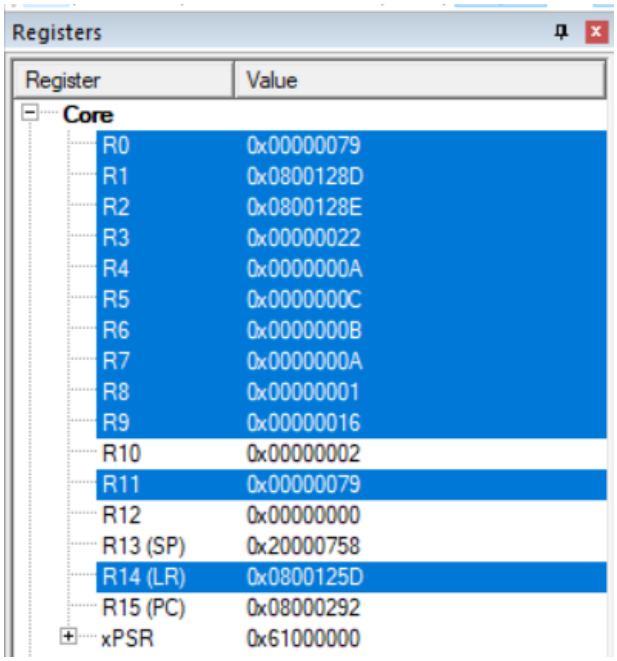
    c. Sqrt(22) = ~4



Registers

Register	Value
Core	
R0	0x00000016
R1	0x20000288
R2	0x20000288
R3	0x20000288
R4	0x00000004
R5	0x00000005
R6	0x00000004
R7	0x00000004
R8	0x00000000
R9	0x00000009
R10	0x00000002
R11	0x00000010
R12	0x20000064
R13 (SP)	0x20000650
R14 (LR)	0x080005CF
R15 (PC)	0x0800027A
xPSR	0x61000000

d. Sqrt(121) = 11

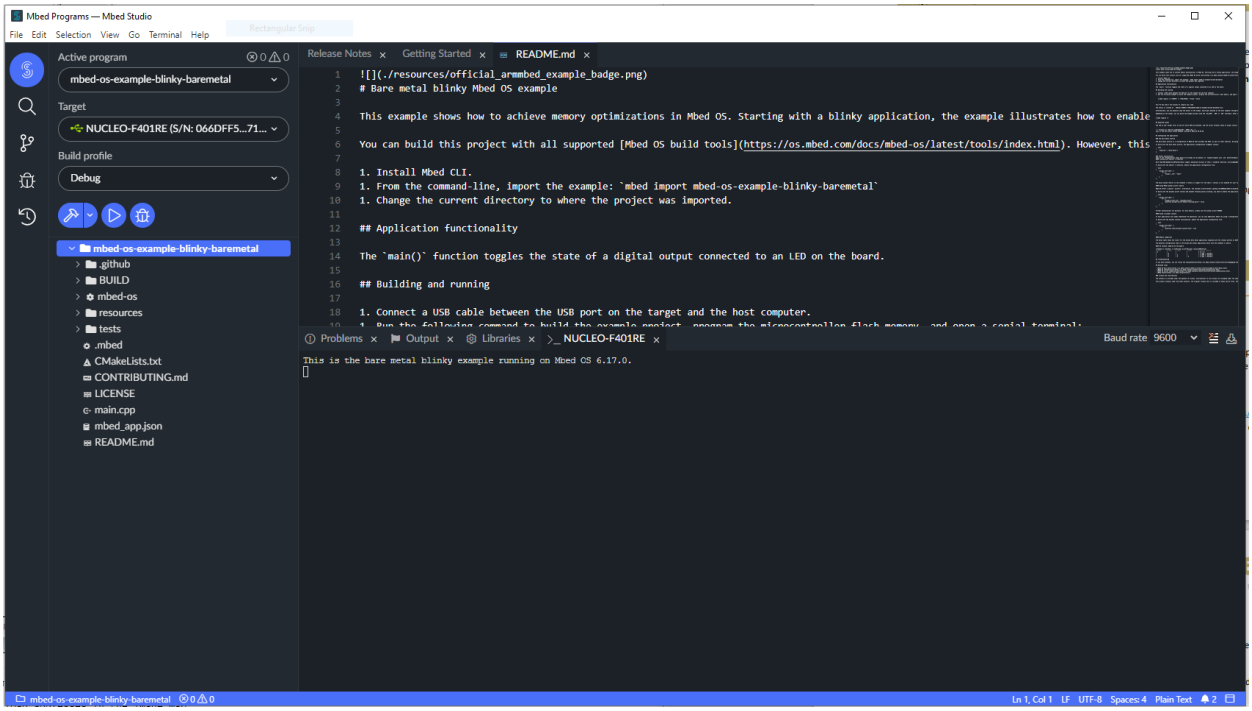


Registers

Register	Value
Core	
R0	0x00000079
R1	0x0800128D
R2	0x0800128E
R3	0x00000022
R4	0x0000000A
R5	0x0000000C
R6	0x0000000B
R7	0x0000000A
R8	0x00000001
R9	0x00000016
R10	0x00000002
R11	0x00000079
R12	0x00000000
R13 (SP)	0x20000758
R14 (LR)	0x0800125D
R15 (PC)	0x08000292
xPSR	0x61000000

4. The minimum number of cycles needed to complete the operation is approximately 26. The max number of cycles is approximately 344 cycles based on finding the square root of the smallest positive integer.

Bonus – Install desktop IDE for Mbed



Appendix

String Copy/Capitalize Source Code (Problem 1)

```
// File: main.c

// Description: This file contains the C functions to copy (my_strcpy)

// and capitalize (my_capitalize) strings.

// Authors: Michael Starks and Sovann Chak

// Tools: Keil uVision with Compiler version 5


void my_strcpy(char* s1, char* s2)

{

   int i = 0;


   while(1)

   {

      s2[i] = s1[i];


      if(s1[i] == '\0')

      {

         return;

      }

                        i++;

   }

}


void my_capitalize(char* s)

{

   int i=0;

   while(1)

   {
```

```
        if(s[i] > 'a' && s[i] <= 'z' )
        {
            s[i] = s[i]-32;
        }

        if(s[i] == '\0')
        {
            return;
        }
        i++;
    }
}

int main(void)
{
            const char a[] = "Hello world!";
  char b[20];

  my_strcpy((char*) a, b);
  my_capitalize(b);

  while (1);
}
```

Memory Map (problem 2)

```
Memory Map of the image

  Image Entry point : 0x08000195

  Load Region LR_IROM1 (Base: 0x08000000, Size: 0x00000e6c, Max: 0x00080000, ABSOLUTE)

    Execution Region ER_IROM1 (Exec base: 0x08000000, Load base: 0x08000000, Size: 0x00000e54, Max: 0x00080000, ABSOLUTE)

    Exec Addr    Load Addr    Size         Type   Attr      Idx    E Section Name        Object

    0x08000000   0x08000000   0x00000194   Data   RO         1585    RESET               startup_stm32f401xe.o
    0x08000194   0x08000194   0x00000000   Code   RO         2180  * .ARM.Collect$$$$00000000  mc_w.l(entry.o)
    0x08000194   0x08000194   0x00000004   Code   RO         2506    .ARM.Collect$$$$00000001  mc_w.l(entry2.o)
    0x08000198   0x08000198   0x00000004   Code   RO         2509    .ARM.Collect$$$$00000004  mc_w.l(entry5.o)
    0x0800019c   0x0800019c   0x00000004   Code   RO         2182    .ARM.Collect$$$$00000006  mc_w.l(entry6b.o)
    0x080001a0   0x080001a0   0x00000000   Code   RO         2511    .ARM.Collect$$$$00000008  mc_w.l(entry7b.o)
    0x080001a0   0x080001a0   0x00000000   Code   RO         2513    .ARM.Collect$$$$0000000A  mc_w.l(entry8b.o)
    0x080001a0   0x080001a0   0x00000008   Code   RO         2514    .ARM.Collect$$$$0000000B  mc_w.l(entry9a.o)
    0x080001a8   0x080001a8   0x00000004   Code   RO         2521    .ARM.Collect$$$$0000000E  mc_w.l(entry12b.o)
    0x080001ac   0x080001ac   0x00000000   Code   RO         2516    .ARM.Collect$$$$0000000F  mc_w.l(entry10a.o)
    0x080001ac   0x080001ac   0x00000000   Code   RO         2518    .ARM.Collect$$$$00000011  mc_w.l(entry11a.o)
    0x080001ac   0x080001ac   0x00000004   Code   RO         2507    .ARM.Collect$$$$00002712  mc_w.l(entry2.o)
    0x080001b0   0x080001b0   0x00000064   Code   RO            1    .text               main.o
    0x08000214   0x08000214   0x00000024   Code   RO         1586    .text               startup_stm32f401xe.o
    0x08000238   0x08000238   0x00000040   Code   RO         2195    .text               mc_w.l(memmovea.o)
    0x08000278   0x08000278   0x00000024   Code   RO         2611    .text               mc_w.l(init.o)
    0x0800029c   0x0800029c   0x00000012   Code   RO         1604    i.$Sub$$main        retarget.o
    0x080002ae   0x080002ae   0x00000002   PAD
    0x080002b0   0x080002b0   0x0000000c   Code   RO         1237    i.HAL_GetTick       stm32f4xx_hal.o
    0x080002bc   0x080002bc   0x00000010   Code   RO         1238    i.HAL_IncTick       stm32f4xx_hal.o
    0x080002cc   0x080002cc   0x00000034   Code   RO         1239    i.HAL_Init          stm32f4xx_hal.o
    0x08000300   0x08000300   0x000000a8   Code   RO           50    i.HAL_InitTick      hal_tick.o
    0x080003a8   0x080003a8   0x00000002   Code   RO         1242    i.HAL_MspInit       stm32f4xx_hal.o
    0x080003aa   0x080003aa   0x00000002   PAD
    0x080003ac   0x080003ac   0x00000024   Code   RO          444    i.HAL_NVIC_SetPriorityGrouping  stm32f4xx_hal_cortex.o
    0x080003d0   0x080003d0   0x00000200   Code   RO          499    i.HAL_RCC_ClockConfig  stm32f4xx_hal_rcc.o
    0x080005d0   0x080005d0   0x000003b4   Code   RO          511    i.HAL_RCC_OscConfig  stm32f4xx_hal_rcc.o
    0x08000984   0x08000984   0x00000032   Code   RO          583    i.HAL_TIM_OC_Init   stm32f4xx_hal_tim.o
    0x080009b6   0x080009b6   0x00000002   Code   RO          585    i.HAL_TIM_OC_MspInit  stm32f4xx_hal_tim.o
    0x080009b8   0x080009b8   0x00000036   Code   RO          586    i.HAL_TIM_OC_Start  stm32f4xx_hal_tim.o
    0x080009ee   0x080009ee   0x00000002   PAD
    0x080009f0   0x080009f0   0x0000006c   Code   RO          394    i.NVIC_SetVector    cmsis_nvic.o
    0x08000a5c   0x08000a5c   0x0000007c   Code   RO         1360    i.SetSysClock_PLL_HSE  system_stm32f4xx.o
    0x08000ad8   0x08000ad8   0x00000084   Code   RO         1362    i.SystemCoreClockUpdate  system_stm32f4xx.o
    0x08000b5c   0x08000b5c   0x00000100   Code   RO         1363    i.SystemInit        system_stm32f4xx.o
    0x08000c5c   0x08000c5c   0x00000094   Code   RO          619    i.TIM_Base_SetConfig  stm32f4xx_hal_tim.o
    0x08000cf0   0x08000cf0   0x0000000e   Code   RO         2655    i.__scatterload_copy  mc_w.l(handlers.o)
    0x08000cfe   0x08000cfe   0x00000002   Code   RO         2656    i.__scatterload_null  mc_w.l(handlers.o)
    0x08000d00   0x08000d00   0x0000000e   Code   RO         2657    i.__scatterload_zeroinit  mc_w.l(handlers.o)
    0x08000d0e   0x08000d0e   0x00000002   Code   RO         1617    i.mbed_main         retarget.o
    0x08000d10   0x08000d10   0x0000000e   Code   RO         1329    i.mbed_sdk_init     mbed_overrides.o
    0x08000d1e   0x08000d1e   0x00000002   PAD
    0x08000d20   0x08000d20   0x00000058   Code   RO           51    i.timer_irq_handler  hal_tick.o
    0x08000d78   0x08000d78   0x00000014   Code   RO         1990    i.us_ticker_clear_interrupt  mbed.ar(us_ticker.o)
    0x08000d8c   0x08000d8c   0x00000014   Code   RO         1991    i.us_ticker_disable_interrupt  mbed.ar(us_ticker.o)
    0x08000da0   0x08000da0   0x00000040   Code   RO         1876    i.us_ticker_irq_handler  mbed.ar(us_ticker_api.o)
    0x08000de0   0x08000de0   0x00000030   Code   RO         1993    i.us_ticker_read    mbed.ar(us_ticker.o)
    0x08000e10   0x08000e10   0x00000014   Code   RO         1994    i.us_ticker_set_interrupt  mbed.ar(us_ticker.o)
    0x08000e24   0x08000e24   0x00000010   Data   RO         1364    .constdata          system_stm32f4xx.o
    0x08000e34   0x08000e34   0x00000020   Data   RO         2653    Region$$Table       anon$$obj.o


    Execution Region RW_IRAM1 (Exec base: 0x20000194, Load base: 0x08000e54, Size: 0x00000090, Max: 0x00017e6c, ABSOLUTE)

    Exec Addr    Load Addr    Size         Type   Attr      Idx    E Section Name        Object

    0x20000194   0x08000e54   0x00000004   Data   RW           53    .data               hal_tick.o
    0x20000198   0x08000e58   0x00000004   Data   RW         1245    .data               stm32f4xx_hal.o
    0x2000019c   0x08000e5c   0x00000004   Data   RW         1365    .data               system_stm32f4xx.o
    0x200001a0   0x08000e60   0x00000008   Data   RW         1879    .data               mbed.ar(us_ticker_api.o)
    0x200001a8   0x08000e68   0x00000004   Data   RW         1996    .data               mbed.ar(us_ticker.o)
    0x200001ac         -       0x0000003c   Zero   RW           52    .bss                hal_tick.o
    0x200001e8         -       0x0000003c   Zero   RW         1995    .bss                mbed.ar(us_ticker.o)
```

Square Root Code

```
// File: main.c

// Description: This file contains an assembly function to find the square root of an integer

// Authors: Michael Starks and Sovann Chak

// Tools: Keil uVision with Compiler version 5



/**

* A function that takes in an int that the user would like to find the square root of

* and returns the closest int to the square root.

* @param x The number the user would like to find the square root of

* @return The closet int to the square root

*/

__asm int my_sqrt(int x)

{

   // Store the stack pointer and register states

   PUSH {R4, R5, R6, R7, R8, R10, R11, R12, lr}; // Push LR to regiter for safe keeping


   /// Function start

   MOV R4, #0x0; // Move the value 0 into A (R0)

   MOV R5, #0x010000; // Move the value 256 (0x100) in B (R1)

   MOV R6, #0xFFFFFFFF; // Move the value -1 (0xFFFFFFFF) into C (R2)

   MOV R7, #0x0; // Store 0 in C_OLD

   MOV R8, #0x0; // Holds DONE value for loop condition

MOV R1, #0xFFFFFFFE; // Holds the value -2

MOV R2, #0x0; // Holds the value 0


// Start of the LOOP
```

```
LOOP
MOV R7, R6; // Store the value of C (R6) in C_OLD (R7)
ADD R3, R4, R5; // Add A (R4) and B (R5) and store in R9
MOV R10, #0x2 // Store 2 (0x2) in R10
SDIV R6, R3, R10; // Divide the value in R9 by R10 (holding 2) and place in C (R6)
MUL R11, R6, R6; // C^2 and place in R10
SUB R11, R11, R0; // Subtract X (R0) from C^2
SUB R11, R11, #0x1; // Subtract 1 from (C^2 - X)

// Check if the value in R11 is less than negative 2
CMP R11, R1; // Compare the value in R11 to -2
BLT LESS; // Branch if less than
CMP R11, R2; // Compare the value in R11 to -1
BGT GREATER // Branch if greater than
B END;  // Else


LESS
MOV R4, R6; // Move C (Stored in R6) to A (Stored in R4)
CMP R6, R7; // Compare to see if C == OLD_C
BEQ END;
// Condition TRUE
BNE LOOP// Branch to LOOP

GREATER
MOV R5, R6; // Move C (Stored in R6) to B (Stored in R5)
CMP R6, R7;
BEQ END;        // Condition TRUE
BNE LOOP; // Branch to LOOP
```

```
// Loop has ended

END

MOV R0, R6;// Move C to return register (R0)

POP {R4, R5, R6, R7, R8, R10, R11, R12, pc}; // Pop contents of registers off the stack and resume program

}


/**

* Main function for testing sqrt function.

* @return return SUCCESS

*/


int main(void)

{

volatile int r, j = 0;

int i;

r = my_sqrt(2);  // Should be 1

r = my_sqrt(4); // Should be 2

r = my_sqrt(22);   // Should be 4

r = my_sqrt(121);   // Should be 4


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

r = my_sqrt(i);

j += r;

}


while (1);
```

```
return 0;
}

// ******************************ARM University Program Copyright © ARM Ltd 2016************************************
```