Spring 2022, Homework 4 (20 points in total)

Q1. (5 pts)
Consider the following IT (IF-THEN) block-based ARM assembly program.

```
CMP     R0, R1
ITTEE   GE
SUBSGE  R3, R0, R2
MOVGE   R0, #25
SUBSLT  R4, R2, R0
MOVLT   R0, #15
```

1. (3pts) Write the equivalent C or C++ program. The C or C++ program variable names must be the same as the register labels. You code must be executable, i.e., I expect to run "./a.out" after compiling your code through command "g++ testq1.cpp". You should simply include your code in the pdf file that you show your answers to Q1 and Q2.

2. (1pt) For the initial values of R0 = 20, R1 = 10, and R2 = 5, what will be the final value of R0 after the execution of the above program?

3. (1pt) Repeat part 2 for R0 = 10, R1 = 20, and R2 = 5.

if $R0 \geq R1$ $R3 = R0 - R2$

if $R0 \geq R1$ $R0 = 25$

if $R0 < R1$ $R4 = R2 - R0$

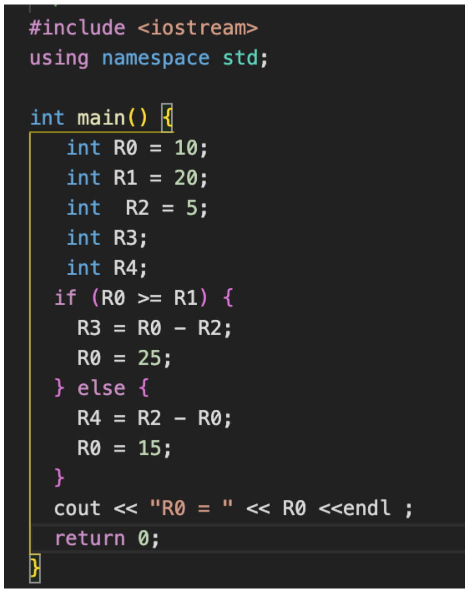
if $R0 < R1$ $R0 = 15$

When $R0 = 20$

$R1 = 10$

$R2 = 5$

$\boxed{R_0 = 25}$



```
#include <iostream>
using namespace std;

int main() {
    int R0 = 10;
    int R1 = 20;
    int  R2 = 5;
    int R3;
    int R4;
  if (R0 >= R1) {
    R3 = R0 - R2;
    R0 = 25;
  } else {
    R4 = R2 - R0;
    R0 = 15;
  }
  cout << "R0 = " << R0 <<endl ;
  return 0;
}
```

When $R_0 = 10$

$R_1 = 20$

$R_2 = 5$

$\boxed{R_0 = 15}$

Q2. (7 pts)

Assume that **src1 DCD 0xFF00FF00** and **src2 DCD 0x00AA00AA** are located consecutively in the READONLY area of memory (see the code below). Also, assume that **src1 starts at address 0x00000008** and **SP = 0x20000400**.

1. For each of the following instructions, calculate the values of registers involved. **Write the values in the comment spaces of the following code as instructed** . For example, "**R0 =**"means you need to fille out like *R0 = 0x00000008*; "**SP =** $\boxed{?}$ " means you need to show the transition of SP contents like *SP = 0x20000400* $\boxed{?}$ *0x20003FC.*

```
                THUMB
StackSize       EQU     0x00000400

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
MyStackMem      SPACE   StackSize

                AREA    RESET, READONLY
                EXPORT  __Vectors
__Vectors
                DCD     MyStackMem + StackSize
                DCD     Reset_Handler

                AREA    MYDATA, DATA, READWRITE

                AREA    MYDCODE, CODE, READONLY
src1            DCD     0xFF00FF00
src2            DCD     0x00AA00AA

                ALIGN
                ENTRY
                EXPORT  Reset_Handler
Reset_Handler
                ; WHAT ARE THE CONTENTS OF EACH REGISTER BELOW? Please fill in as comments
HW4_2           LDR     R0, =src1        ; R0 = 0x00000008
                ADD     R0, R0, #1       ; R0 = 0x00000009
                LDRB    R1, [R0]         ; R1 = 0x000000FF
draw_mem1       PUSH    {R1}             ; SP = 0x200003FC            -> 0x0000001A
                ADD     R0, #4           ; R0 = 0x0000000D
                LDRSB   R2, [R0]         ; R2 = 0x00000000
draw_mem2       PUSH    {R2}             ; SP = 0x200003F8            -> 0x00000024
                MOV     R3, SP           ; R3 = 0x200003F8
                ADD     R3, #3           ; R3 = 0x200003FB
                LDR     R4, [R3]         ; R4 = 0x0000FF00
                SUB     R4, #0x004C004C; R4 = 0xFFB4FEB4
draw_mem3       STRB    R4, [R3]         ; R3 = 0x0000FF00
                POP     {R2}             ; SP = 0x200003F8  ->, 0x00000034   R2 = 0xB4000000
                POP     {R1}             ; SP = 0x200003FC  ->, 0x00000034   R1 = 0x000000FF
```

2. Stack memory: fill out blanks upon the completion of each event: draw_mem1, draw_mem2, and draw_mem3. Each cell should store only one byte.

Address	draw_mem1	draw_mem2	draw_mem3
	PUSH {R1}	PUSH {R2}	STR R4, [R3]
0x200003F8			R4=0xFF00
0x200003F9		LR2=address(MOV, R3, SP)	LR2=address(MOV, R3, SP)
0x200003FA		R2=)x00	R2=)x00
0x200003FB		R0=0x0D	R3=0xFB
0x200003FC	LR=address(ADD R0, #4)	LR=address(Add R0, #4)	LR=address(Add R0, #4)
0x200003FD	R1=0xFF	R1=0xFF	R1=0xFF
0x200003FE	R0=0x09	R0=0x08	R0=0x08
0x200003FF	R1=src1	R1=src1	R1=src1
0x20000400			

Q3. (8 pts) Parity-Bit Adding Program

Write a program in ARM assembly code that adds an odd parity to each ASCII character. While you may use parity.s runnable on VisUAL (which you can find in Canvas/files/folder/code/VisUAL), your HW4-Q3 program must run on Keil uVersion. Your code must also satisfy the following specifications:

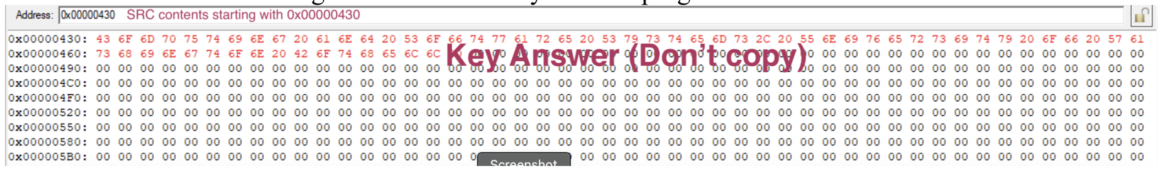
1. To run Keil uVersion,
   a. Create the HW4 folder on your Windows desktop, start uVersion, choose "new uVersion projection".
   b. Select this HW4 folder.
   c. Use HW4 for HW4.uvproj/.uvprojx file names.
   d. Choose Texas Instruments/Tiva C Series/TM4C123x Series/TM4C1233H6PM (the same platform as in lab1b).
   e. Create main.s under Project HW4/Target 1/Source Group1/.

2. Simply comment out line 236 of startup_TM4C123.s: **236 ; BLX R0**
   so that startup_TM4C123.s skips calling SystemInit and directly invokes __main at line 238. (**This is only a modification you need to do in startup_TM4C123.s.**)

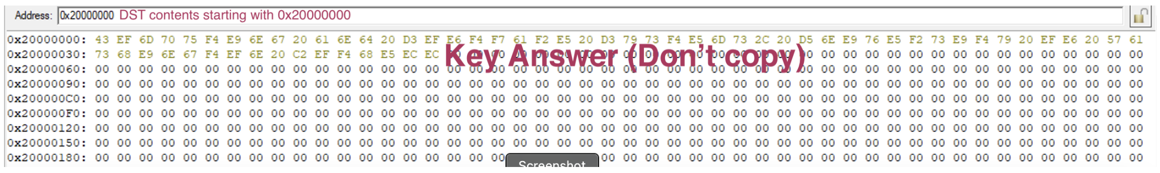
3. In your main.s, get started with the following code snippet:

```
        AREA      MyData,  DATA,  READWRITE
dst     SPACE     200

        AREA      MyCODE,  DATA,  READONLY
src     DCB       "Computing  and  Software  Systems,  University  of  Washington  Bothell",  0

        AREA      |.text|,  CODE,  READONLY

        EXPORT    __main                    [WEAK]
        ENTRY
__main
        LDR                R0,  =src          ;  src  address
        LDR                R1,  =dst          ;  dst  address
        ;                  R2   =  the  current  ASCCI  Char
        ;                  R3   =  counter  from  7  to  0
        ;                  R4   =  #ones
```

4. Use R0, R1, R2, R3, and R4 as specified in the above code snipped:
   a. To read a new ASCII character from src, you should use: `LDRB     R2,  [R0],  #1`
   b. To write an ASCII character with an odd parity to dst, use: `STRB     R2,  [R1],  #1`
5. **If you use parity.s from Canvas, make sure:**
   a. R0 should be R2 in HW4.
   b. R1 should be R4 in HW4.
   c. R2 should be R3 in HW4.
6. To check if R4 is odd, use: `LSRS     R4,  #1` If a carry is set, R4 was odd, in which case don't add a parity bit at bit 7 of R2. Otherwise it must have been even, and thus you need to set bit 7 of R2.
7. Unlike VisUAL, a real ARM processor does not stop with END. **The very last statement in main.s should be an infinite loop like:** `end_of_main  B       end_of_main`
8. Run your program, capture snapshots of **src** and **dst** memory contents, and verify if your **dst** memory contents got correct odd parities. The following is the key answer's results. Don't copy them. You need to generate those with your own program.



```
Address: 0x00000430   SRC contents starting with 0x00000430

0x00000430: 43 6F 6D 70 75 74 69 6E 67 20 61 6E 64 20 53 6F 66 74 77 61 72 65 20 53 79 73 74 65 6D 73 2C 20 55 6E 69 76 65 72 73 69 74 79 20 6F 66 20 57 61
0x00000460: 73 68 69 6E 67 74 6F 6E 20 42 6F 74 68 65 6C 6C Key Answer (Don't copy) 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000490: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000004C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000004F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000520: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000550: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000580: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000005B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Screenshot 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



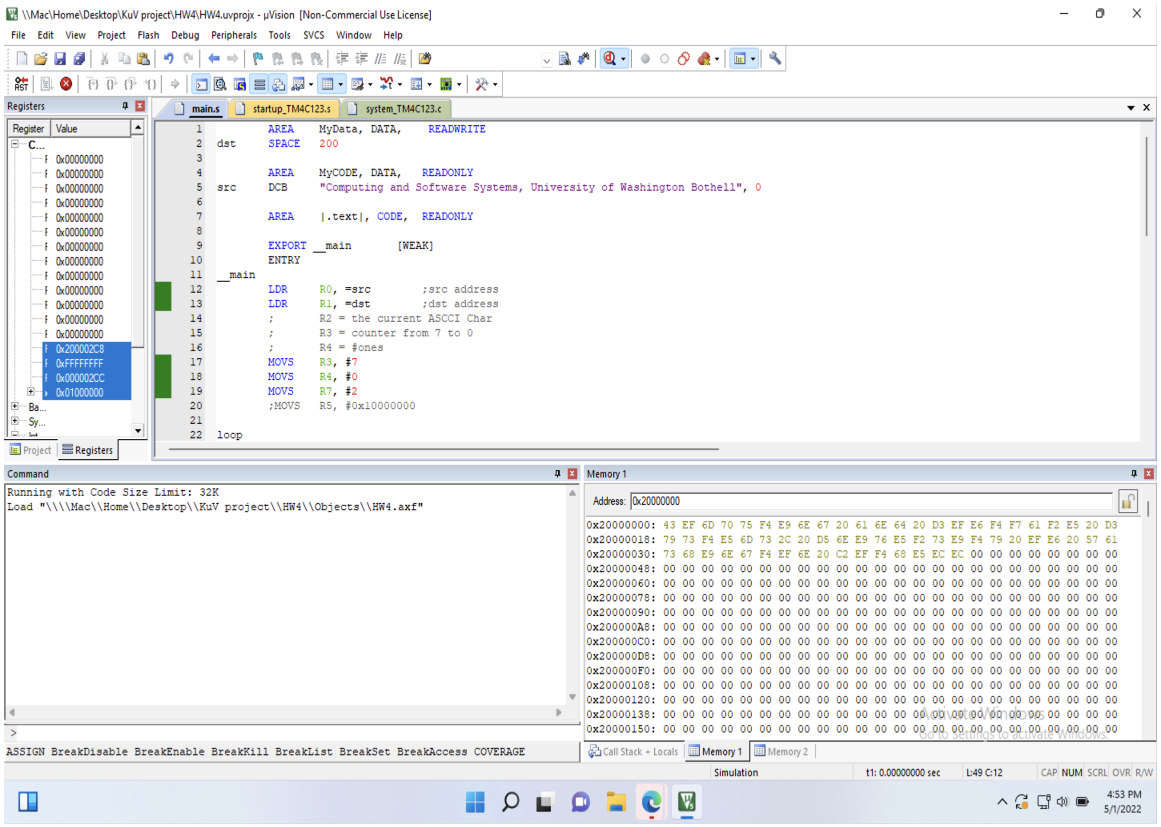
Address: 0x20000000 DST contents starting with 0x20000000

Key Answer (Don't copy)

```
0x20000000: 43 EF 6D 70 75 F4 E9 6E 67 20 61 6E 64 20 D3 EF E6 F4 F7 61 F2 E5 20 D3 79 73 F4 E5 6D 73 2C 20 D5 6E E9 76 E5 F2 73 E9 F4 79 20 EF E6 20 57 61
0x20000030: 73 68 E9 6E 67 F4 EF 6E 20 C2 EF F4 68 E5 EC EC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

For this question, you need to submit the followings:

1. The source file: main.s. **(If no source file, then you got zero point for this question)**
2. The screenshots of memory view (**src** and **dst** contents) to support your correct results. Please include these screenshots in the same pdf file that you show your answers to Q1 and Q2.
3. After the screenshots, copy the code in main.s into the same pdf file that you show your answers to Q1 and Q2.



\\Mac\Home\Desktop\KuV project\HW4\HW4.uvprojx - µVision [Non-Commercial Use License]

Register	Value
C...	
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x00000000
	0x200002C8
	0xFFFFFFFF
	0x000002CC
	0x01000000
Ba...	
Sy...	

```
        AREA    MyData, DATA,    READWRITE
dst     SPACE   200

        AREA    MyCODE, DATA,   READONLY
src     DCB     "Computing and Software Systems, University of Washington Bothell", 0

        AREA    |.text|, CODE,  READONLY

        EXPORT __main       [WEAK]
        ENTRY
__main
        LDR     R0, =src        ;src address
        LDR     R1, =dst        ;dst address
        ;       R2 = the current ASCCI Char
        ;       R3 = counter from 7 to 0
        ;       R4 = #ones
        MOVS    R3, #7
        MOVS    R4, #0
        MOVS    R7, #2
        ;MOVS   R5, #0x10000000

loop
```

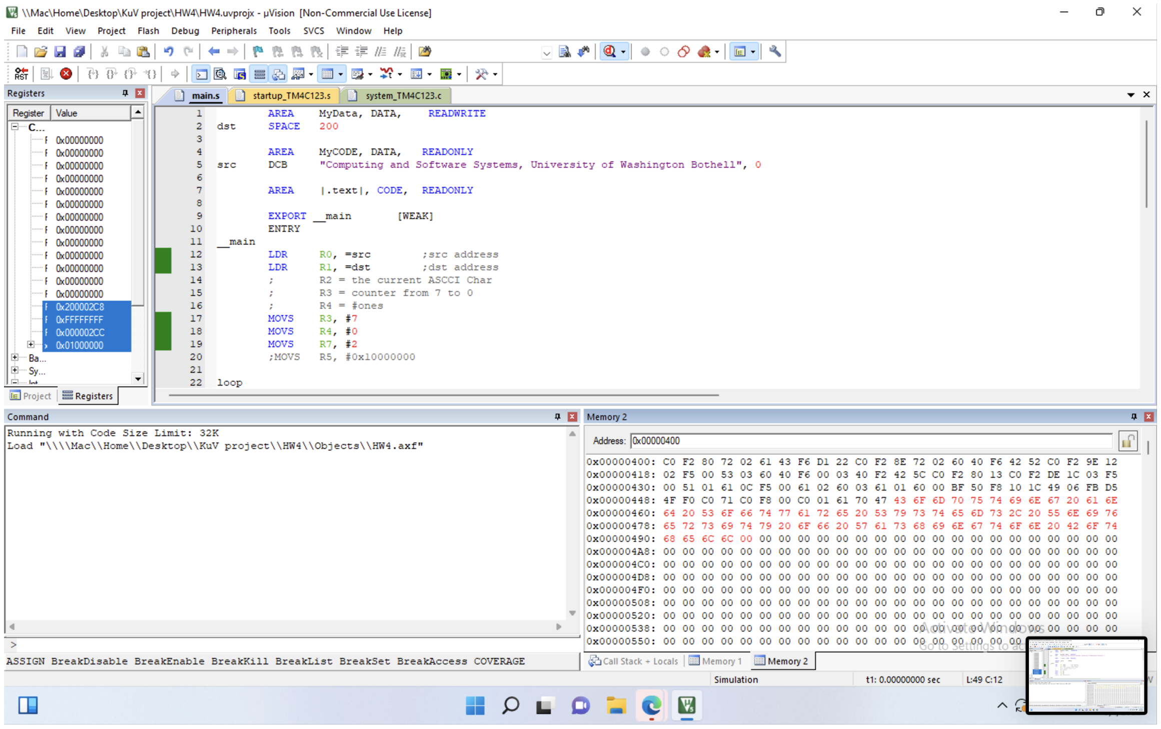
```
Running with Code Size Limit: 32K
Load "\\\\Mac\\Home\\Desktop\\KuV project\\HW4\\Objects\\HW4.axf"
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

Address: 0x20000000

```
0x20000000: 43 EF 6D 70 75 F4 E9 6E 67 20 61 6E 64 20 D3 EF E6 F4 F7 61 F2 E5 20 D3
0x20000018: 79 73 F4 E5 6D 73 2C 20 D5 6E E9 76 E5 F2 73 E9 F4 79 20 EF E6 20 57 61
0x20000030: 73 68 E9 6E 67 F4 EF 6E 20 C2 EF F4 68 E5 EC EC 00 00 00 00 00 00 00 00
0x20000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000D8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000108: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000138: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Activate Windows
Go to Settings to activate Windows.



\\Mac\Home\Desktop\KuV project\HW4\HW4.uvprojx - µVision [Non-Commercial Use License]

```
        AREA    MyData, DATA,    READWRITE
dst     SPACE   200

        AREA    MyCODE, DATA,   READONLY
src     DCB     "Computing and Software Systems, University of Washington Bothell", 0

        AREA    |.text|, CODE,  READONLY

        EXPORT __main       [WEAK]
        ENTRY
__main
        LDR     R0, =src        ;src address
        LDR     R1, =dst        ;dst address
        ;       R2 = the current ASCCI Char
        ;       R3 = counter from 7 to 0
        ;       R4 = #ones
        MOVS    R3, #7
        MOVS    R4, #0
        MOVS    R7, #2
        ;MOVS   R5, #0x10000000

loop
```

Registers:

```
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x200002C8
0xFFFFFFFF
0x000002CC
0x01000000
```

Command:

```
Running with Code Size Limit: 32K
Load "\\\\Mac\\Home\\Desktop\\KuV project\\HW4\\Objects\\HW4.axf"
```

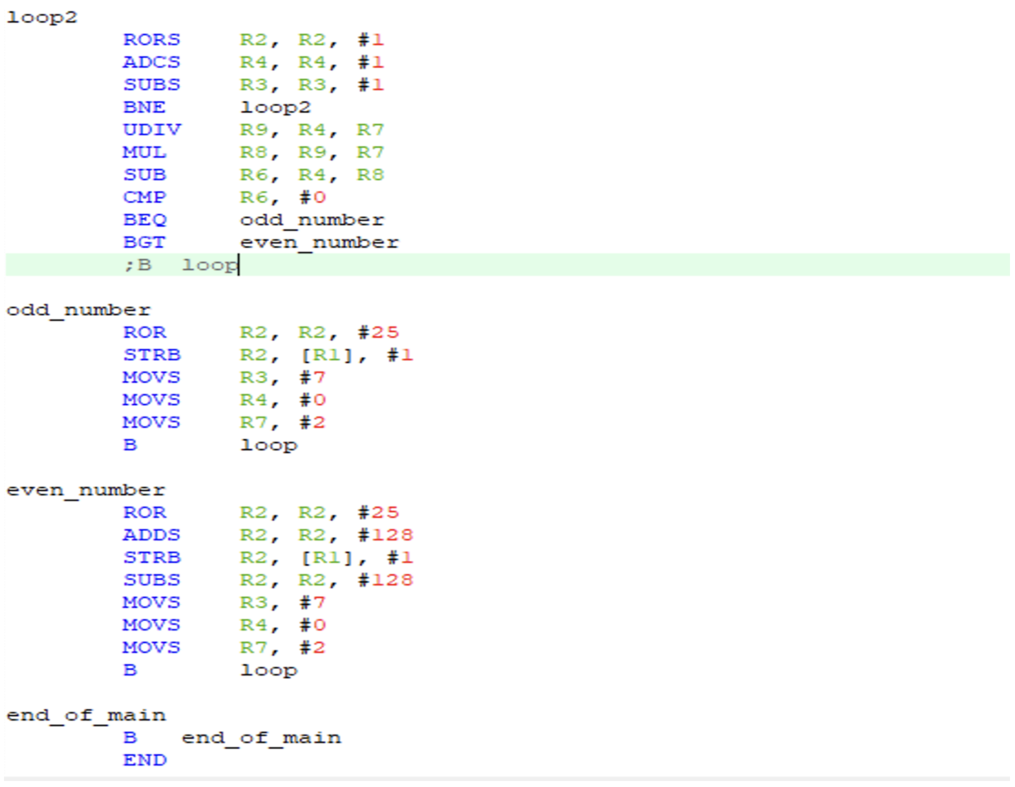
```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

Memory 2 — Address: 0x00000400

```
0x00000400: C0 F2 80 72 02 61 43 F6 D1 22 C0 F2 8E 72 02 60 40 F6 42 52 C0 F2 9E 12
0x00000418: 02 F5 00 53 03 60 40 F6 00 03 40 F2 42 5C C0 F2 80 13 C0 F2 DE 1C 03 F5
0x00000430: 00 51 01 61 0C F5 00 61 02 60 03 61 01 60 00 BF 50 F8 10 1C 49 06 FB D5
0x00000448: 4F F0 C0 71 C0 F8 00 C0 01 61 70 47 43 6F 6D 70 75 74 69 6E 67 20 61 6E
0x00000460: 64 20 53 6F 66 74 77 61 72 65 20 53 79 73 74 65 6D 73 2C 20 55 6E 69 76
0x00000478: 65 72 73 69 74 79 20 6F 66 20 57 61 73 68 69 6E 67 74 6F 6E 20 42 6F 74
0x00000490: 68 65 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000004A8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000004C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000004D8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000004F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000508: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000520: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000538: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000550: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Simulation   t1: 0.00000000 sec   L:49 C:12

```
        AREA    MyData, DATA,    READWRITE
dst     SPACE   200

        AREA    MyCODE, DATA,   READONLY
src     DCB     "Computing and Software Systems, University of Washington Bothell", 0

        AREA    |.text|, CODE,  READONLY

        EXPORT __main       [WEAK]
        ENTRY
__main
        LDR     R0, =src        ;src address
        LDR     R1, =dst        ;dst address
        ;       R2 = the current ASCCI Char
        ;       R3 = counter from 7 to 0
        ;       R4 = #ones
        MOVS    R3, #7
        MOVS    R4, #0
        MOVS    R7, #2
        ;MOVS   R5, #0x10000000

loop

        LDRB R2, [R0], #1
        CMP R2, #0
        ;BGT    loop
        BEQ end_of_main
```



```
loop2
        RORS    R2, R2, #1
        ADCS    R4, R4, #1
        SUBS    R3, R3, #1
        BNE     loop2
        UDIV    R9, R4, R7
        MUL     R8, R9, R7
        SUB     R6, R4, R8
        CMP     R6, #0
        BEQ     odd_number
        BGT     even_number
        ;B  loop

odd_number
        ROR     R2, R2, #25
        STRB    R2, [R1], #1
        MOVS    R3, #7
        MOVS    R4, #0
        MOVS    R7, #2
        B       loop

even_number
        ROR     R2, R2, #25
        ADDS    R2, R2, #128
        STRB    R2, [R1], #1
        SUBS    R2, R2, #128
        MOVS    R3, #7
        MOVS    R4, #0
        MOVS    R7, #2
        B       loop

end_of_main
        B   end_of_main
        END
```