University of Dhaka

Department of Computer Science and Engineering

CSE 3113 - Microprocessor and Assembly Language Lab

Batch 28 / 3RD Year 1ST Semester

Lab 2

Submitted To:

Dr. Upama Kabir
Dr. Mosarrat Jahan
Mr. Jargis Ahmed
Mr. Palash Roy

Submitted By:

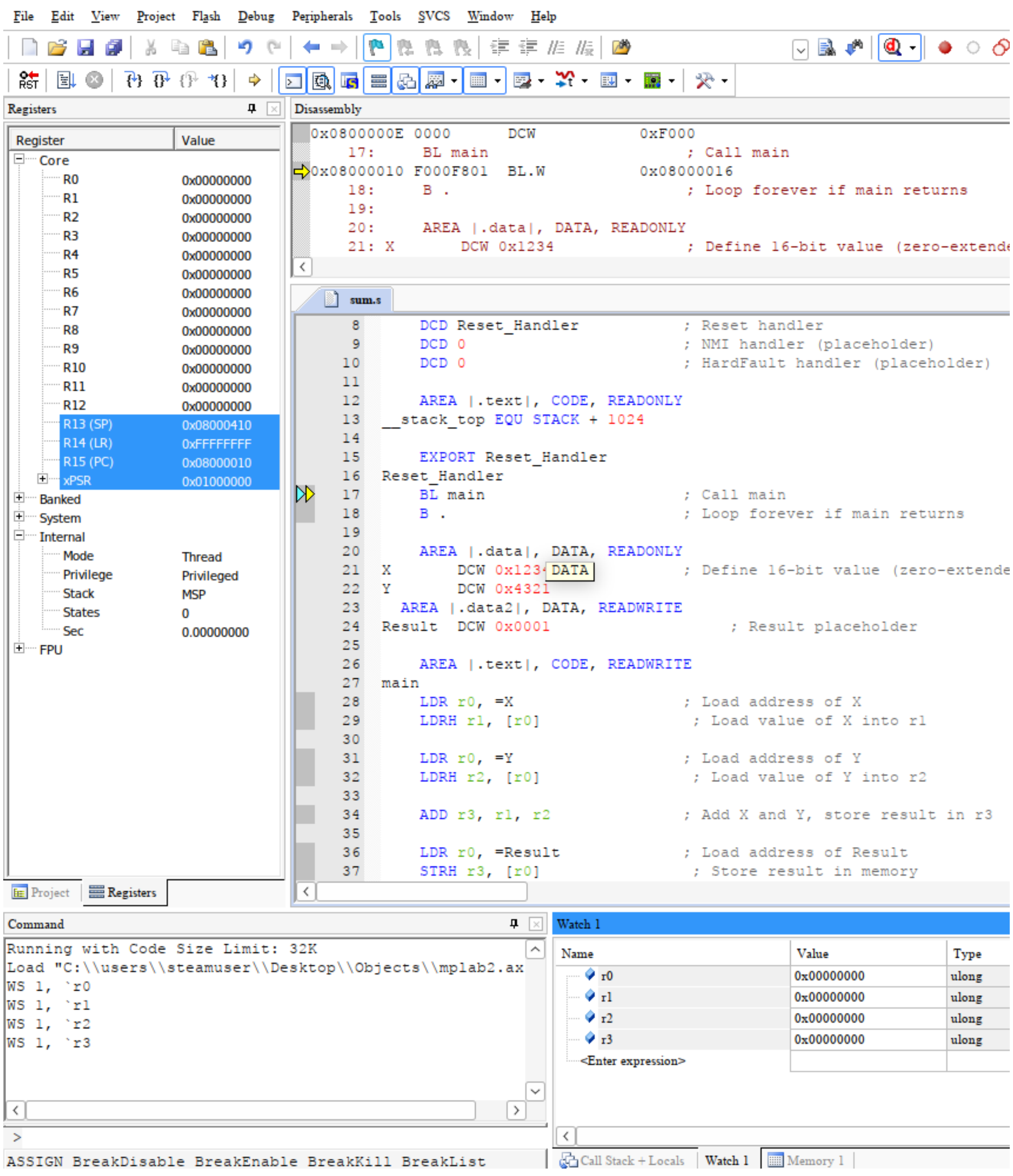
Farzana Tasnim (14)

1 Lab Tasks

1.1 Task 1

Write an assembly language to perform to add the contents of the 16-bit variable X to the contents of the 16-bit variable Y and place the result in the 16-bit variable Result.

1.1.1 Screenshot that shows the state of the system after the code has been loaded.



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000010
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000
FPU	

```
0x0800000E 0000      DCW          0xF000
    17:      BL main                  ; Call main
0x08000010 F000F801  BL.W         0x08000016
    18:      B .                      ; Loop forever if main returns
    19:
    20:      AREA |.data|, DATA, READONLY
    21: X        DCW 0x1234           ; Define 16-bit value (zero-extende
```

```
    DCD Reset_Handler              ; Reset handler
    DCD 0                          ; NMI handler (placeholder)
    DCD 0                          ; HardFault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main                        ; Call main
    B .                            ; Loop forever if main returns

    AREA |.data|, DATA, READONLY
X       DCW 0x123 DATA             ; Define 16-bit value (zero-extende
Y       DCW 0x4321
  AREA |.data2|, DATA, READWRITE
Result  DCW 0x0001                      ; Result placeholder

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                     ; Load address of X
    LDRH r1, [r0]                   ; Load value of X into r1

    LDR r0, =Y                     ; Load address of Y
    LDRH r2, [r0]                   ; Load value of Y into r2

    ADD r3, r1, r2                 ; Add X and Y, store result in r3

    LDR r0, =Result                ; Load address of Result
    STRH r3, [r0]                   ; Store result in memory
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
```

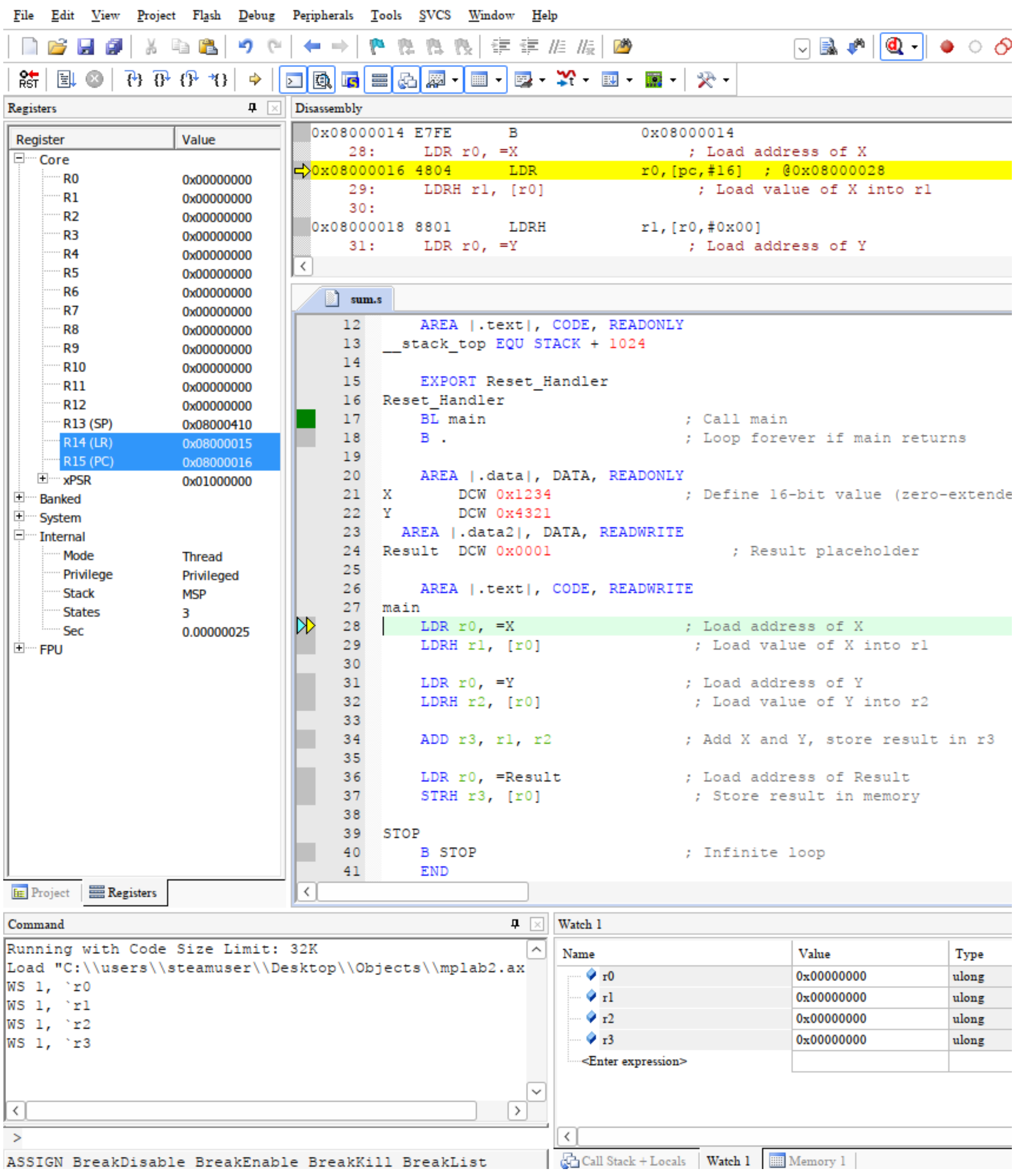
```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000000	ulong
r1	0x00000000	ulong
r2	0x00000000	ulong
r3	0x00000000	ulong
<Enter expression>		

Figure 1: After build and debugging, this is the state

1.1.2 Screenshot that shows the situation after the code has been executed.

In this section, the R15 which is **PC**(program Counter) is changing it's value to point to the next instruction. It is a 32-bit register that holds the address of the next instruction to be executed. R13 is **stack pointer**(Points to the top of the stack), R14 is **link Register**(Holds the return address after a function call), **xPSR**(Program Status Register) holds the status flags.



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000016
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	3
Sec	0.00000025
FPU	

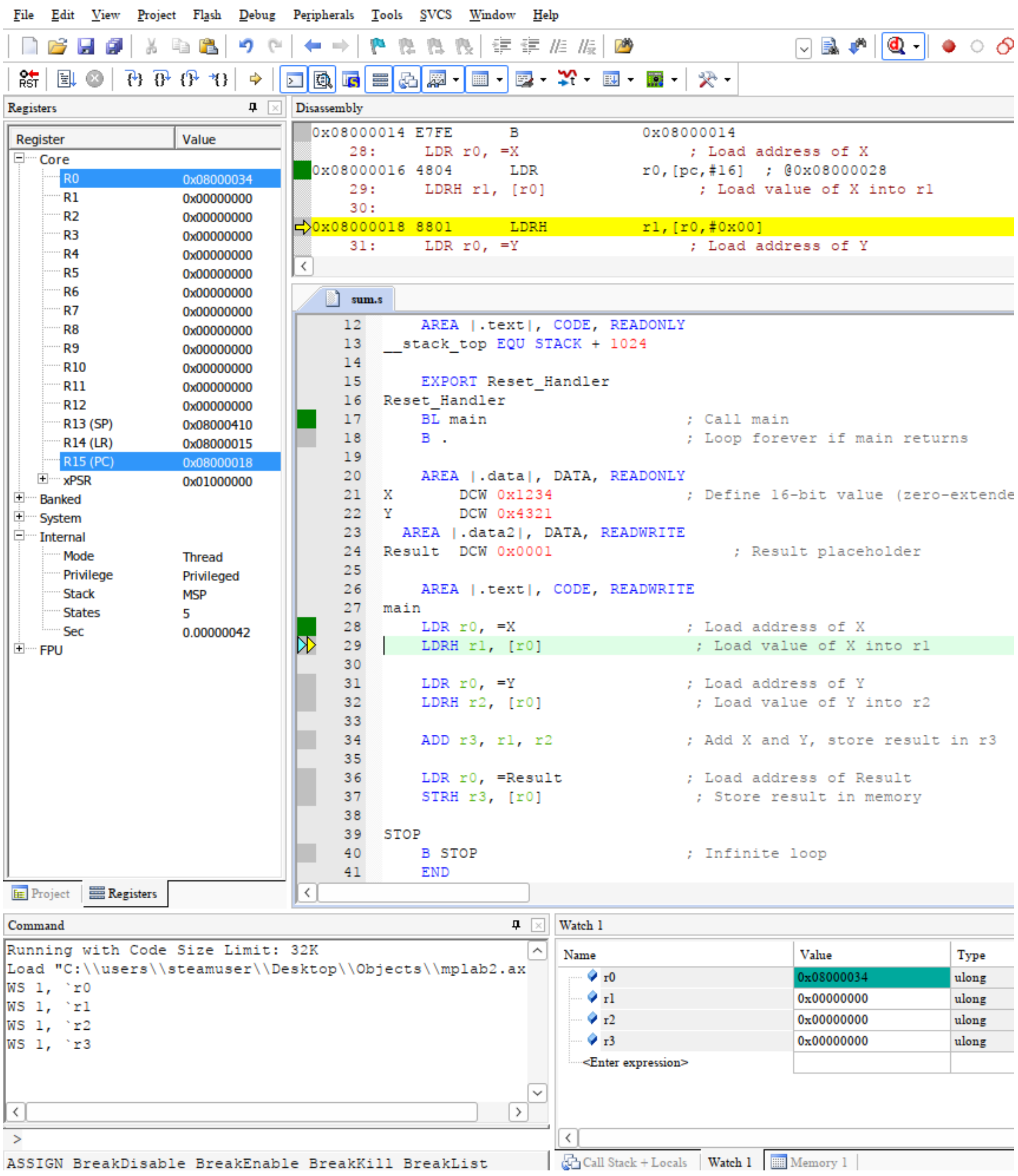
```
0x08000014 E7FE      B          0x08000014
    28:     LDR r0, =X                  ; Load address of X
0x08000016 4804      LDR        r0,[pc,#16]  ; @0x08000028
    29:     LDRH r1, [r0]               ; Load value of X into r1
    30:
0x08000018 8801      LDRH       r1,[r0,#0x00]
    31:     LDR r0, =Y                  ; Load address of Y
```

```
    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main                         ; Call main
    B .                             ; Loop forever if main returns

    AREA |.data|, DATA, READONLY
X       DCW 0x1234                  ; Define 16-bit value (zero-extende
Y       DCW 0x4321
  AREA |.data2|, DATA, READWRITE
Result  DCW 0x0001                       ; Result placeholder

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                      ; Load address of X
    LDRH r1, [r0]                    ; Load value of X into r1

    LDR r0, =Y                      ; Load address of Y
    LDRH r2, [r0]                    ; Load value of Y into r2

    ADD r3, r1, r2                  ; Add X and Y, store result in r3

    LDR r0, =Result                 ; Load address of Result
    STRH r3, [r0]                    ; Store result in memory

STOP
    B STOP                          ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000000	ulong
r1	0x00000000	ulong
r2	0x00000000	ulong
r3	0x00000000	ulong
<Enter expression>		

Figure 2: Call of main



Register	Value
Core	
R0	0x08000034
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000018
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	5
Sec	0.00000042
FPU	

```
0x08000014 E7FE      B              0x08000014
    28:      LDR r0, =X                   ; Load address of X
0x08000016 4804      LDR            r0,[pc,#16]  ; @0x08000028
    29:      LDRH r1, [r0]                 ; Load value of X into r1
    30:
0x08000018 8801      LDRH           r1,[r0,#0x00]
    31:      LDR r0, =Y                   ; Load address of Y
```

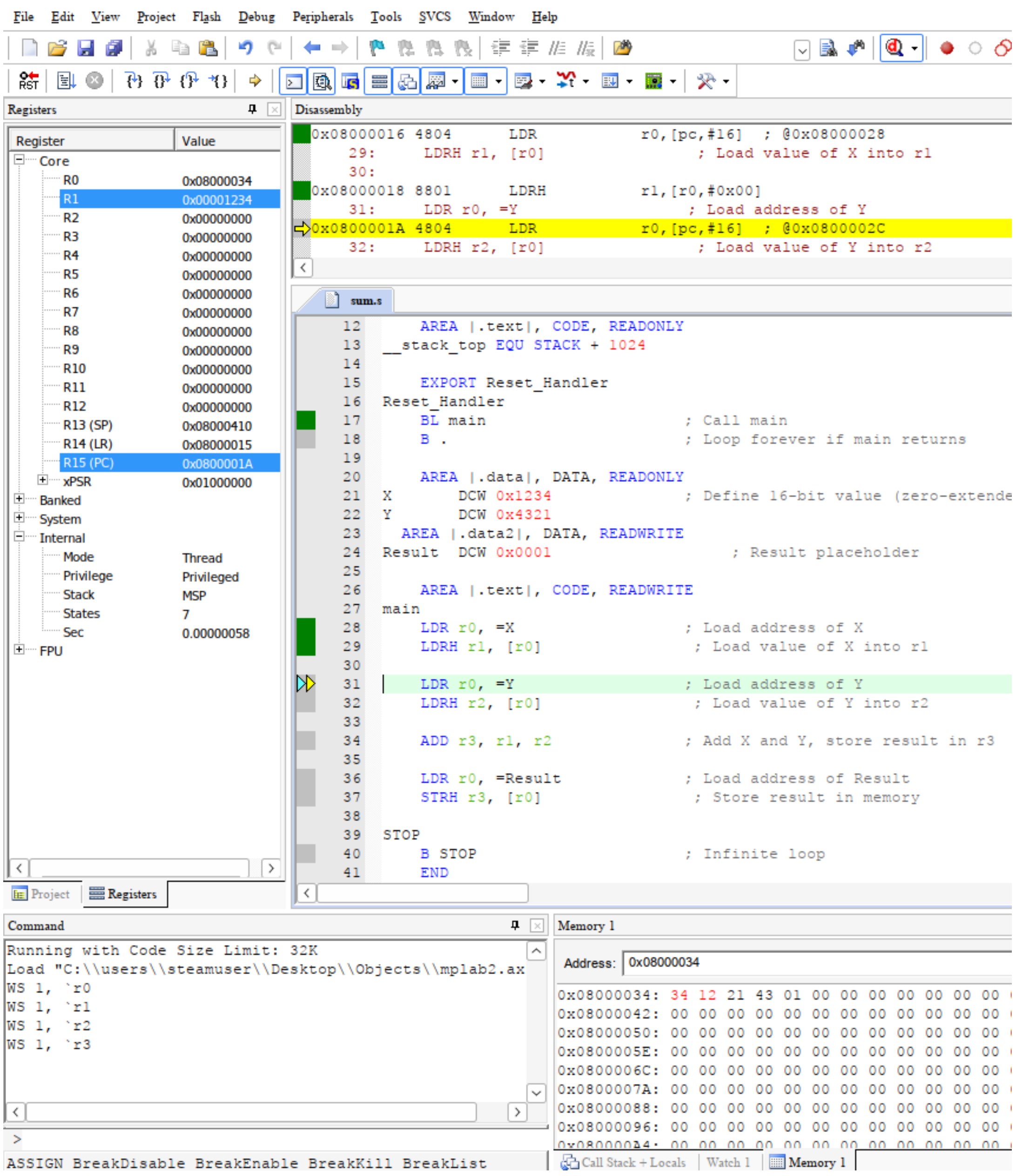
```
    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main                         ; Call main
    B .                             ; Loop forever if main returns

    AREA |.data|, DATA, READONLY
X       DCW 0x1234                  ; Define 16-bit value (zero-extende
Y       DCW 0x4321
  AREA |.data2|, DATA, READWRITE
Result  DCW 0x0001                       ; Result placeholder

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                      ; Load address of X
    LDRH r1, [r0]                    ; Load value of X into r1

    LDR r0, =Y                      ; Load address of Y
    LDRH r2, [r0]                    ; Load value of Y into r2

    ADD r3, r1, r2                  ; Add X and Y, store result in r3

    LDR r0, =Result                 ; Load address of Result
    STRH r3, [r0]                    ; Store result in memory

STOP
    B STOP                          ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
```

```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x08000034	ulong
r1	0x00000000	ulong
r2	0x00000000	ulong
r3	0x00000000	ulong
<Enter expression>		

Figure 3: Loaded the address of X in r0 register



Register	Value
Core	
R0	0x08000034
R1	0x00001234
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x0800001A
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	7
Sec	0.00000058
FPU	

```
0x08000016 4804      LDR          r0,[pc,#16]  ; @0x08000028
    29:      LDRH r1, [r0]                   ; Load value of X into r1
    30:
0x08000018 8801      LDRH         r1,[r0,#0x00]
    31:      LDR r0, =Y                     ; Load address of Y
0x0800001A 4804      LDR          r0,[pc,#16]  ; @0x0800002C
    32:      LDRH r2, [r0]                   ; Load value of Y into r2
```

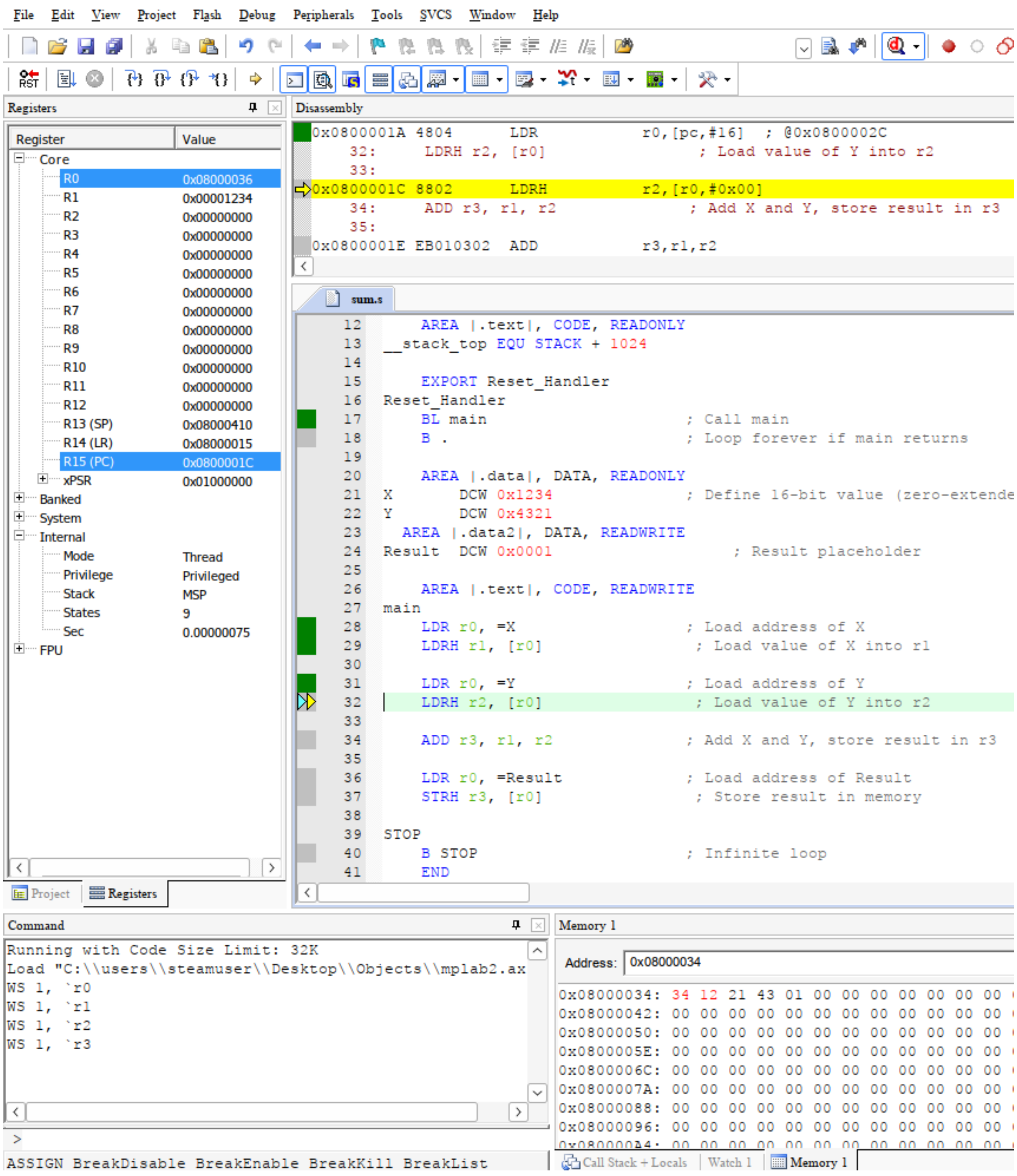
```
    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main                      ; Call main
    B .                          ; Loop forever if main returns

    AREA |.data|, DATA, READONLY
X       DCW 0x1234               ; Define 16-bit value (zero-extende
Y       DCW 0x4321
  AREA |.data2|, DATA, READWRITE
Result  DCW 0x0001                    ; Result placeholder

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                   ; Load address of X
    LDRH r1, [r0]                 ; Load value of X into r1

    LDR r0, =Y                   ; Load address of Y
    LDRH r2, [r0]                 ; Load value of Y into r2

    ADD r3, r1, r2               ; Add X and Y, store result in r3

    LDR r0, =Result              ; Load address of Result
    STRH r3, [r0]                 ; Store result in memory

STOP
    B STOP                       ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x08000034

0x08000034: 34 12 21 43 01 00 00 00 00 00 00 00
0x08000042: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000050: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800005E: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800006C: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800007A: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000088: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000096: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000A4: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 4: Loaded the content of X in r1

We can see the data of a particular address of memory in the memory view. CortexM4 is **little-endian**. Little-endian is a byte-ordering method where the least significant byte of a multi-byte data value is stored at the lowest memory address.


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Project
Registers
sum.s
Call Stack + Locals
Watch 1
Memory 1

Register	Value
Core	
R0	0x08000036
R1	0x00001234
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x0800001C
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	9
Sec	0.00000075
FPU	

Disassembly

```
0x0800001A 4804      LDR          r0,[pc,#16]  ; @0x0800002C
    32:      LDRH r2, [r0]                ; Load value of Y into r2
    33:
0x0800001C 8802      LDRH         r2,[r0,#0x00]
    34:      ADD r3, r1, r2              ; Add X and Y, store result in r3
    35:
0x0800001E EB010302  ADD          r3,r1,r2
```

```
12      AREA |.text|, CODE, READONLY
13  __stack_top EQU STACK + 1024
14
15      EXPORT Reset_Handler
16  Reset_Handler
17      BL main                        ; Call main
18      B .                            ; Loop forever if main returns
19
20      AREA |.data|, DATA, READONLY
21  X        DCW 0x1234                ; Define 16-bit value (zero-extende
22  Y        DCW 0x4321
23    AREA |.data2|, DATA, READWRITE
24  Result   DCW 0x0001                     ; Result placeholder
25
26      AREA |.text|, CODE, READWRITE
27  main
28      LDR r0, =X                     ; Load address of X
29      LDRH r1, [r0]                   ; Load value of X into r1
30
31      LDR r0, =Y                     ; Load address of Y
32      LDRH r2, [r0]                   ; Load value of Y into r2
33
34      ADD r3, r1, r2                 ; Add X and Y, store result in r3
35
36      LDR r0, =Result                ; Load address of Result
37      STRH r3, [r0]                   ; Store result in memory
38
39  STOP
40      B STOP                         ; Infinite loop
41      END
```

Command

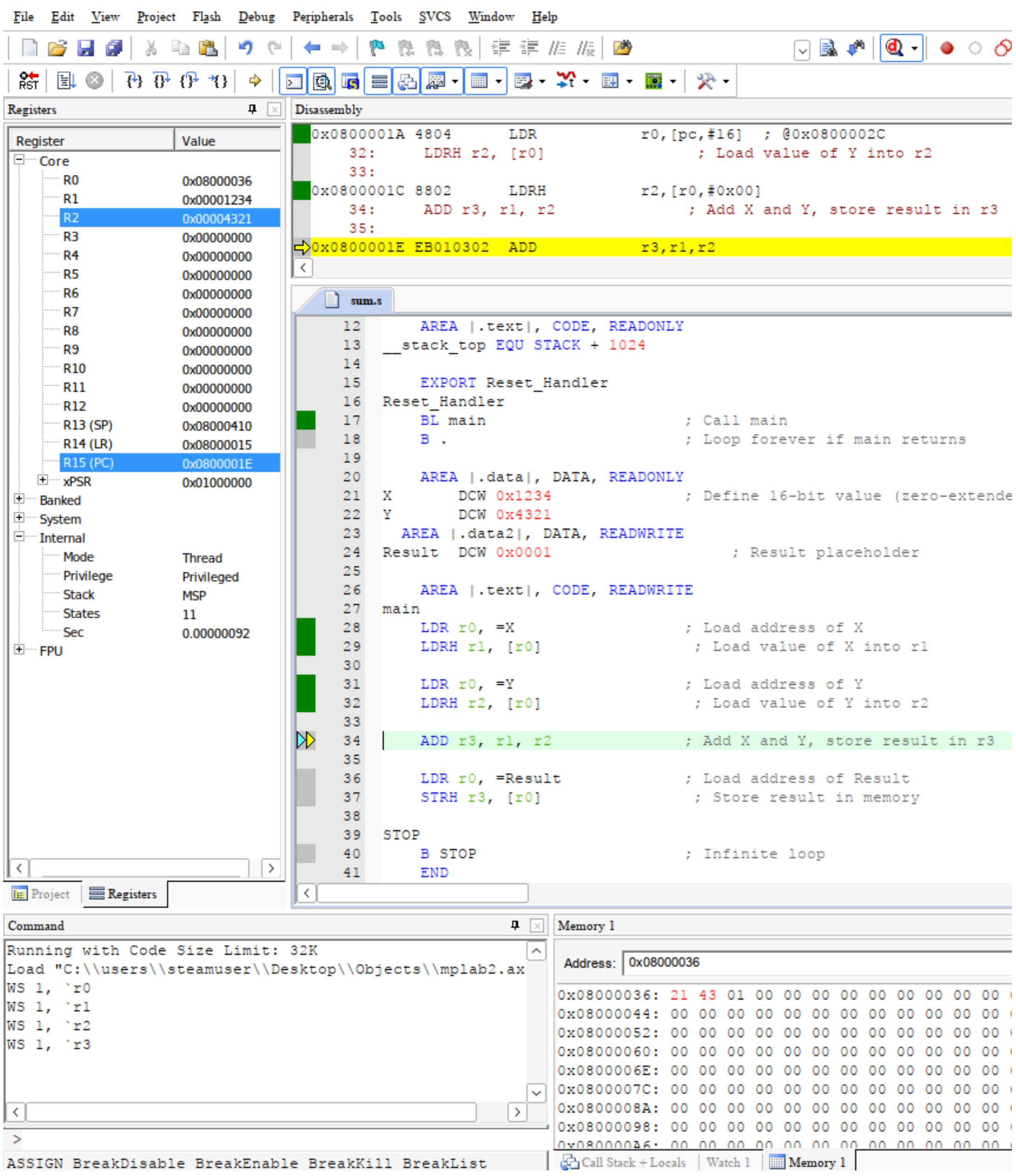
```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Memory 1

Address: 0x08000034

```
0x08000034: 34 12 21 43 01 00 00 00 00 00 00 00
0x08000042: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000050: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800005E: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800006C: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800007A: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000088: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000096: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000A4: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 5: Loaded the address of Y in r0 register



Register	Value
Core	
R0	0x08000036
R1	0x00001234
R2	0x00004321
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x0800001E
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	11
Sec	0.00000092
FPU	

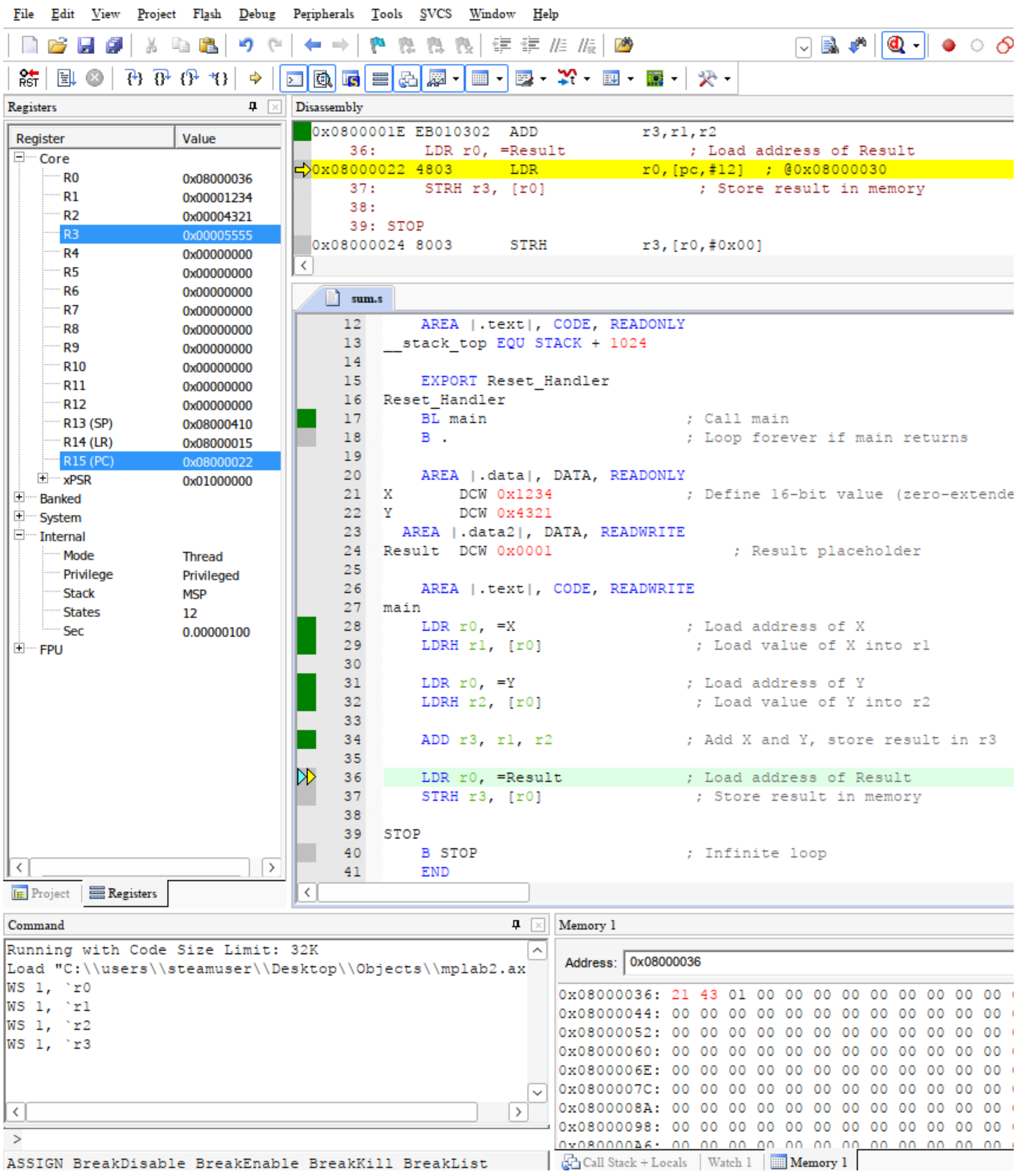
```
0x0800001A 4804      LDR          r0,[pc,#16]  ; @0x0800002C
    32:     LDRH r2, [r0]                 ; Load value of Y into r2
    33:
0x0800001C 8802      LDRH         r2,[r0,#0x00]
    34:     ADD r3, r1, r2               ; Add X and Y, store result in r3
    35:
0x0800001E EB010302  ADD          r3,r1,r2
```

```
    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main                         ; Call main
    B .                             ; Loop forever if main returns

    AREA |.data|, DATA, READONLY
X       DCW 0x1234                  ; Define 16-bit value (zero-extende
Y       DCW 0x4321
  AREA |.data2|, DATA, READWRITE
Result  DCW 0x0001                       ; Result placeholder

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                      ; Load address of X
    LDRH r1, [r0]                    ; Load value of X into r1

    LDR r0, =Y                      ; Load address of Y
    LDRH r2, [r0]                    ; Load value of Y into r2

    ADD r3, r1, r2                  ; Add X and Y, store result in r3

    LDR r0, =Result                 ; Load address of Result
    STRH r3, [r0]                    ; Store result in memory

STOP
    B STOP                          ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x08000036

0x08000036: 21 43 01 00 00 00 00 00 00 00 00 00
0x08000044: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000052: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000060: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800006E: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800007C: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800008A: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000098: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000A6: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 6: Loaded the content of Y in r2



Register	Value
Core	
R0	0x08000036
R1	0x00001234
R2	0x00004321
R3	0x00005555
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000022
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	12
Sec	0.00000100
FPU	

```
0x0800001E EB010302  ADD          r3,r1,r2
    36:     LDR r0, =Result              ; Load address of Result
0x08000022 4803      LDR          r0,[pc,#12]  ; @0x08000030
    37:     STRH r3, [r0]                 ; Store result in memory
    38:
    39: STOP
0x08000024 8003      STRH         r3,[r0,#0x00]
```

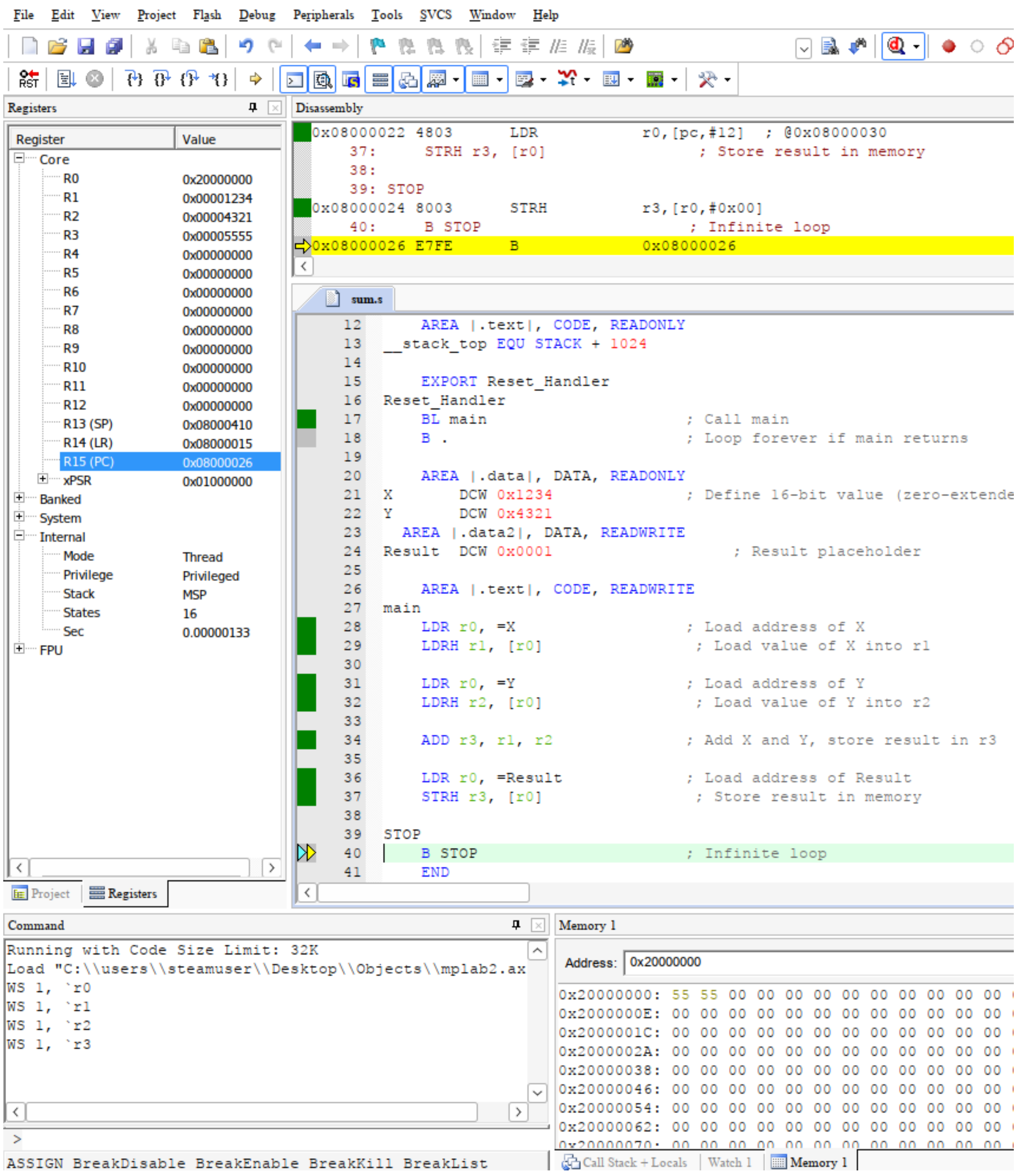
```
    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main                          ; Call main
    B .                              ; Loop forever if main returns

    AREA |.data|, DATA, READONLY
X       DCW 0x1234                   ; Define 16-bit value (zero-extende
Y       DCW 0x4321
  AREA |.data2|, DATA, READWRITE
Result  DCW 0x0001                        ; Result placeholder

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                       ; Load address of X
    LDRH r1, [r0]                     ; Load value of X into r1

    LDR r0, =Y                       ; Load address of Y
    LDRH r2, [r0]                     ; Load value of Y into r2

    ADD r3, r1, r2                   ; Add X and Y, store result in r3

    LDR r0, =Result                  ; Load address of Result
    STRH r3, [r0]                     ; Store result in memory

STOP
    B STOP                           ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Address: 0x08000036

```
0x08000036: 21 43 01 00 00 00 00 00 00 00 00 00
0x08000044: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000052: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000060: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800006E: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800007C: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800008A: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000098: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000A6: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 7: Added the value of r1, r2 and placed the result in r3



Register	Value
Core	
R0	0x20000000
R1	0x00001234
R2	0x00004321
R3	0x00005555
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000026
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	16
Sec	0.00000133
FPU	

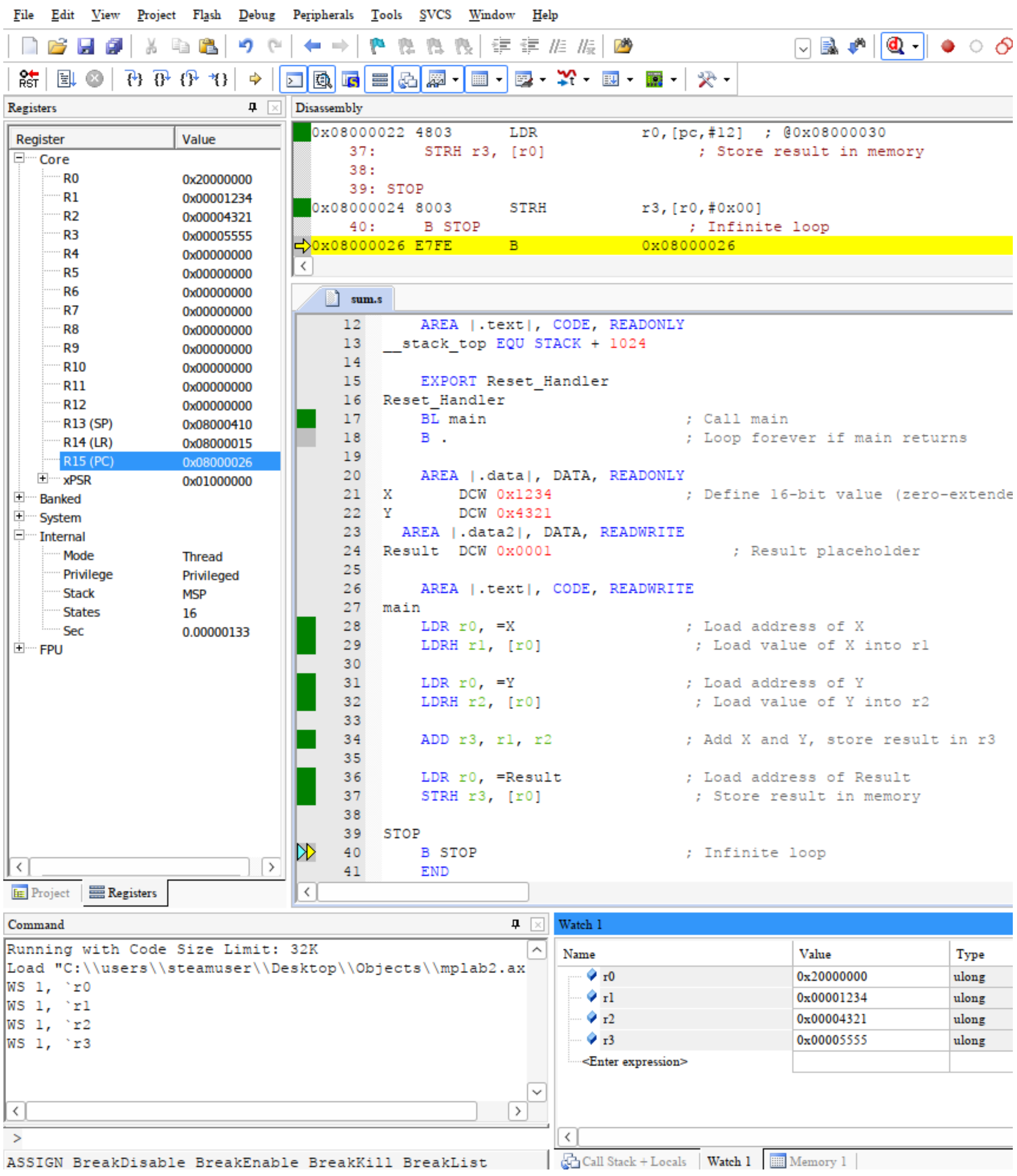
```
0x08000022 4803      LDR          r0,[pc,#12]  ; @0x08000030
    37:     STRH r3, [r0]             ; Store result in memory
    38:
    39: STOP
0x08000024 8003      STRH         r3,[r0,#0x00]
    40:     B STOP                   ; Infinite loop
0x08000026 E7FE      B            0x08000026
```

```
    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main                         ; Call main
    B .                             ; Loop forever if main returns

    AREA |.data|, DATA, READONLY
X       DCW 0x1234                  ; Define 16-bit value (zero-extende
Y       DCW 0x4321
  AREA |.data2|, DATA, READWRITE
Result  DCW 0x0001                       ; Result placeholder

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                      ; Load address of X
    LDRH r1, [r0]                    ; Load value of X into r1

    LDR r0, =Y                      ; Load address of Y
    LDRH r2, [r0]                    ; Load value of Y into r2

    ADD r3, r1, r2                  ; Add X and Y, store result in r3

    LDR r0, =Result                 ; Load address of Result
    STRH r3, [r0]                    ; Store result in memory

STOP
    B STOP                          ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x20000000

0x20000000: 55 55 00 00 00 00 00 00 00 00 00 00
0x2000000E: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000001C: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000002A: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000038: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000046: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000054: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000062: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 8: Loaded the address of Result in r3 and store the data of r3 in Result



```
Registers
Register          Value
Core
  R0              0x20000000
  R1              0x00001234
  R2              0x00004321
  R3              0x00005555
  R4              0x00000000
  R5              0x00000000
  R6              0x00000000
  R7              0x00000000
  R8              0x00000000
  R9              0x00000000
  R10             0x00000000
  R11             0x00000000
  R12             0x00000000
  R13 (SP)        0x08000410
  R14 (LR)        0x08000015
  R15 (PC)        0x08000026
  xPSR            0x01000000
Banked
System
Internal
  Mode            Thread
  Privilege       Privileged
  Stack           MSP
  States          16
  Sec             0.00000133
FPU
```

```
Disassembly
0x08000022 4803      LDR        r0,[pc,#12]  ; @0x08000030
    37:      STRH r3, [r0]              ; Store result in memory
    38:
    39: STOP
0x08000024 8003      STRH       r3,[r0,#0x00]
    40:      B STOP                     ; Infinite loop
0x08000026 E7FE      B          0x08000026
```

```
sum.s
    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main                        ; Call main
    B .                            ; Loop forever if main returns

    AREA |.data|, DATA, READONLY
X       DCW 0x1234                 ; Define 16-bit value (zero-extende
Y       DCW 0x4321
  AREA |.data2|, DATA, READWRITE
Result  DCW 0x0001                      ; Result placeholder

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                     ; Load address of X
    LDRH r1, [r0]                   ; Load value of X into r1

    LDR r0, =Y                     ; Load address of Y
    LDRH r2, [r0]                   ; Load value of Y into r2

    ADD r3, r1, r2                 ; Add X and Y, store result in r3

    LDR r0, =Result                ; Load address of Result
    STRH r3, [r0]                   ; Store result in memory

STOP
    B STOP                         ; Infinite loop
    END
```

```
Command
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Watch 1

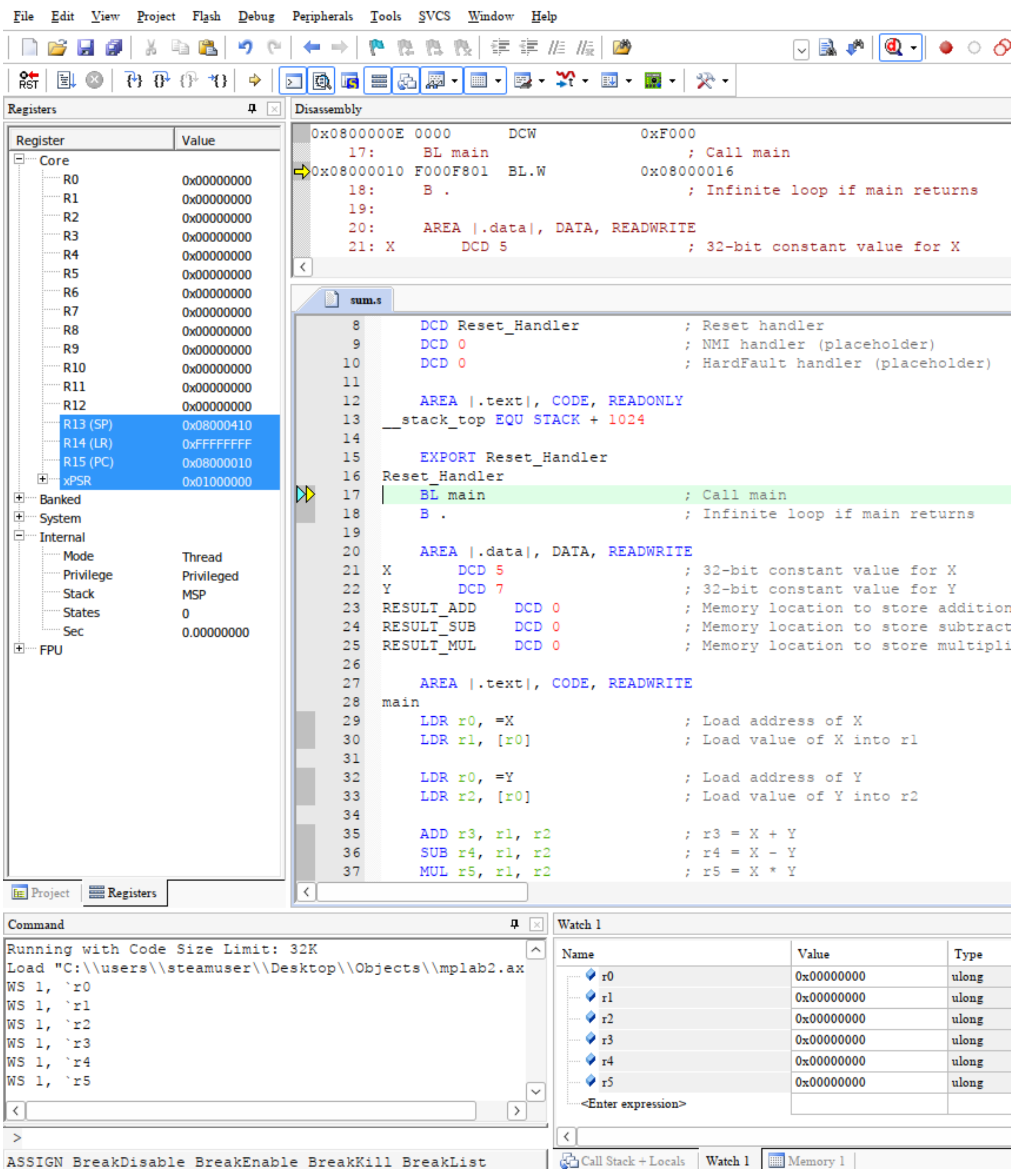
Name	Value	Type
r0	0x20000000	ulong
r1	0x00001234	ulong
r2	0x00004321	ulong
r3	0x00005555	ulong
<Enter expression>		

Figure 9: End state

1.2 Task 2

Write an assembly language to perform all the arithmetic operations (Addition, Subtraction and Multiplication) on two variables X and Y. You don't have to handle overflow. You will put the data in memory in the form of constants before the program runs.

1.2.1 Screenshot that shows the state of the system after the code has been loaded.



```
Register        Value
Core
  R0            0x00000000
  R1            0x00000000
  R2            0x00000000
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x08000410
  R14 (LR)      0xFFFFFFFF
  R15 (PC)      0x08000010
  xPSR          0x01000000
Banked
System
Internal
  Mode          Thread
  Privilege     Privileged
  Stack         MSP
  States        0
  Sec           0.00000000
FPU
```

```
0x0800000E 0000      DCW          0xF000
    17:       BL main                 ; Call main
0x08000010 F000F801  BL.W         0x08000016
    18:       B .                     ; Infinite loop if main returns
    19:
    20:       AREA |.data|, DATA, READWRITE
    21: X         DCD 5               ; 32-bit constant value for X
```

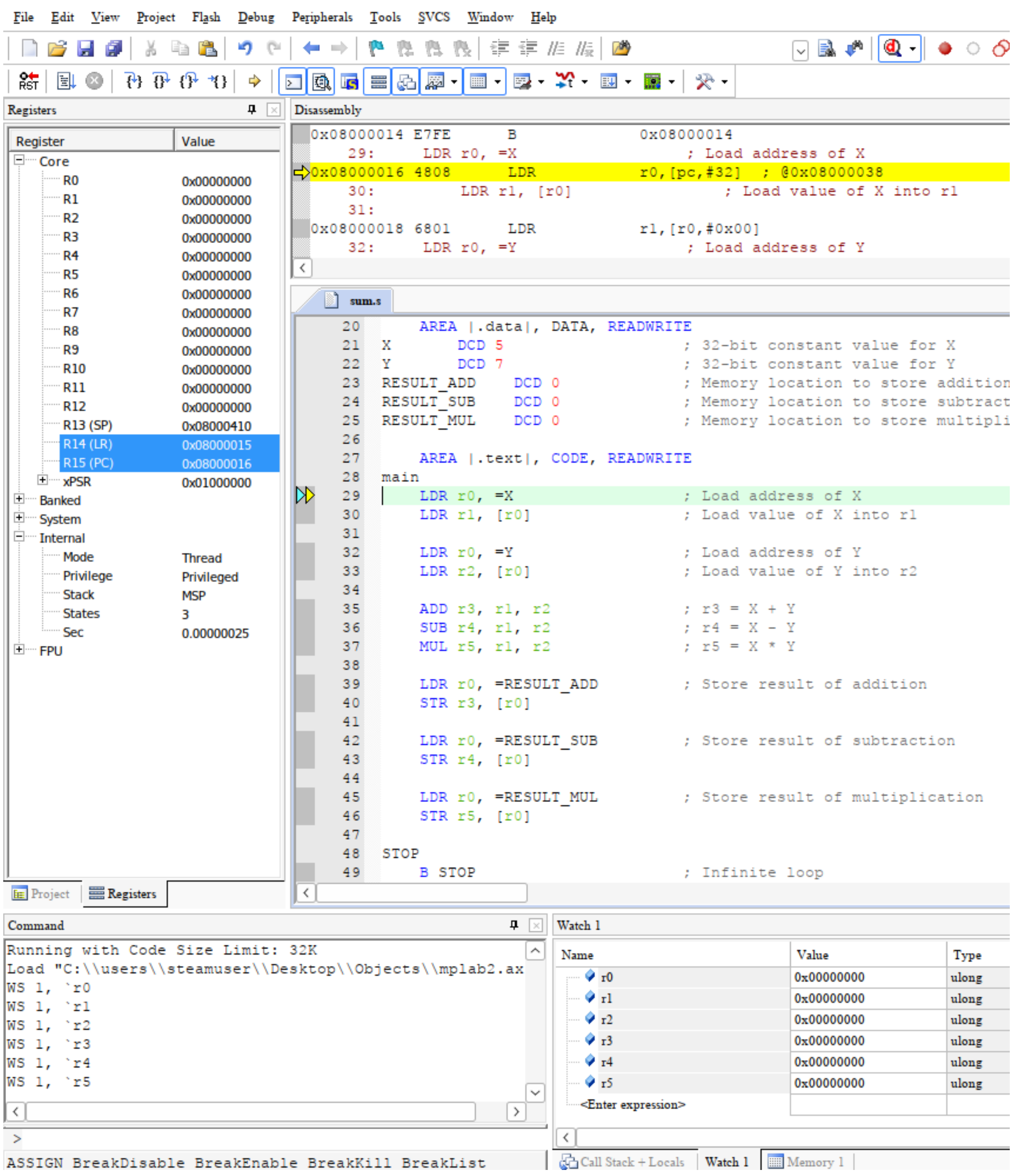
```
    DCD Reset_Handler                  ; Reset handler
    DCD 0                              ; NMI handler (placeholder)
    DCD 0                              ; HardFault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main                            ; Call main
    B .                                ; Infinite loop if main returns

    AREA |.data|, DATA, READWRITE
X       DCD 5                          ; 32-bit constant value for X
Y       DCD 7                          ; 32-bit constant value for Y
RESULT_ADD    DCD 0                    ; Memory location to store addition
RESULT_SUB    DCD 0                    ; Memory location to store subtract
RESULT_MUL    DCD 0                    ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                         ; Load address of X
    LDR r1, [r0]                       ; Load value of X into r1

    LDR r0, =Y                         ; Load address of Y
    LDR r2, [r0]                       ; Load value of Y into r2

    ADD r3, r1, r2                     ; r3 = X + Y
    SUB r4, r1, r2                     ; r4 = X - Y
    MUL r5, r1, r2                     ; r5 = X * Y
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000000	ulong
r1	0x00000000	ulong
r2	0x00000000	ulong
r3	0x00000000	ulong
r4	0x00000000	ulong
r5	0x00000000	ulong
<Enter expression>		

Figure 10: After build and debugging, this is the state

1.2.2 Screenshot that shows the situation after the code has been executed.



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000016
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	3
Sec	0.00000025
FPU	

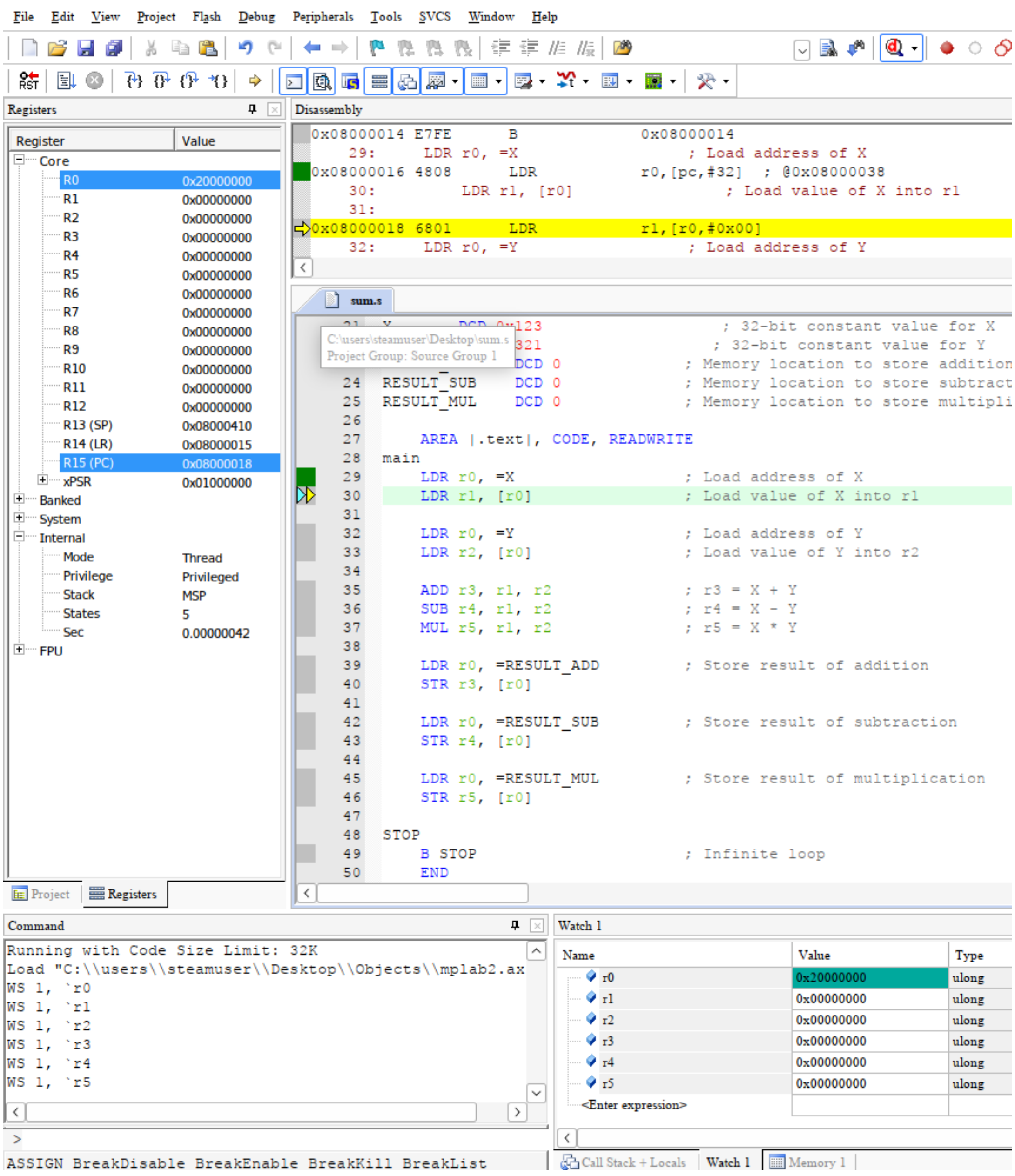
```
0x08000014 E7FE      B              0x08000014
    29:     LDR r0, =X                ; Load address of X
0x08000016 4808      LDR            r0,[pc,#32]  ; @0x08000038
    30:         LDR r1, [r0]               ; Load value of X into r1
    31:
0x08000018 6801      LDR            r1,[r0,#0x00]
    32:     LDR r0, =Y                ; Load address of Y
```

```
    AREA |.data|, DATA, READWRITE
X       DCD 5                           ; 32-bit constant value for X
Y       DCD 7                           ; 32-bit constant value for Y
RESULT_ADD    DCD 0                     ; Memory location to store addition
RESULT_SUB    DCD 0                     ; Memory location to store subtract
RESULT_MUL    DCD 0                     ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                          ; Load address of X
    LDR r1, [r0]                        ; Load value of X into r1

    LDR r0, =Y                          ; Load address of Y
    LDR r2, [r0]                        ; Load value of Y into r2

    ADD r3, r1, r2                      ; r3 = X + Y
    SUB r4, r1, r2                      ; r4 = X - Y
    MUL r5, r1, r2                      ; r5 = X * Y

    LDR r0, =RESULT_ADD                 ; Store result of addition
    STR r3, [r0]

    LDR r0, =RESULT_SUB                 ; Store result of subtraction
    STR r4, [r0]

    LDR r0, =RESULT_MUL                 ; Store result of multiplication
    STR r5, [r0]

STOP
    B STOP                              ; Infinite loop
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000000	ulong
r1	0x00000000	ulong
r2	0x00000000	ulong
r3	0x00000000	ulong
r4	0x00000000	ulong
r5	0x00000000	ulong
<Enter expression>		

Figure 11: Call of main



Register	Value
Core	
R0	0x20000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000018
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	5
Sec	0.00000042
FPU	

```
0x08000014 E7FE      B          0x08000014
    29:      LDR r0, =X                  ; Load address of X
0x08000016 4808      LDR        r0,[pc,#32]  ; @0x08000038
    30:          LDR r1, [r0]                ; Load value of X into r1
    31:
0x08000018 6801      LDR        r1,[r0,#0x00]
    32:      LDR r0, =Y                  ; Load address of Y
```

sum.s

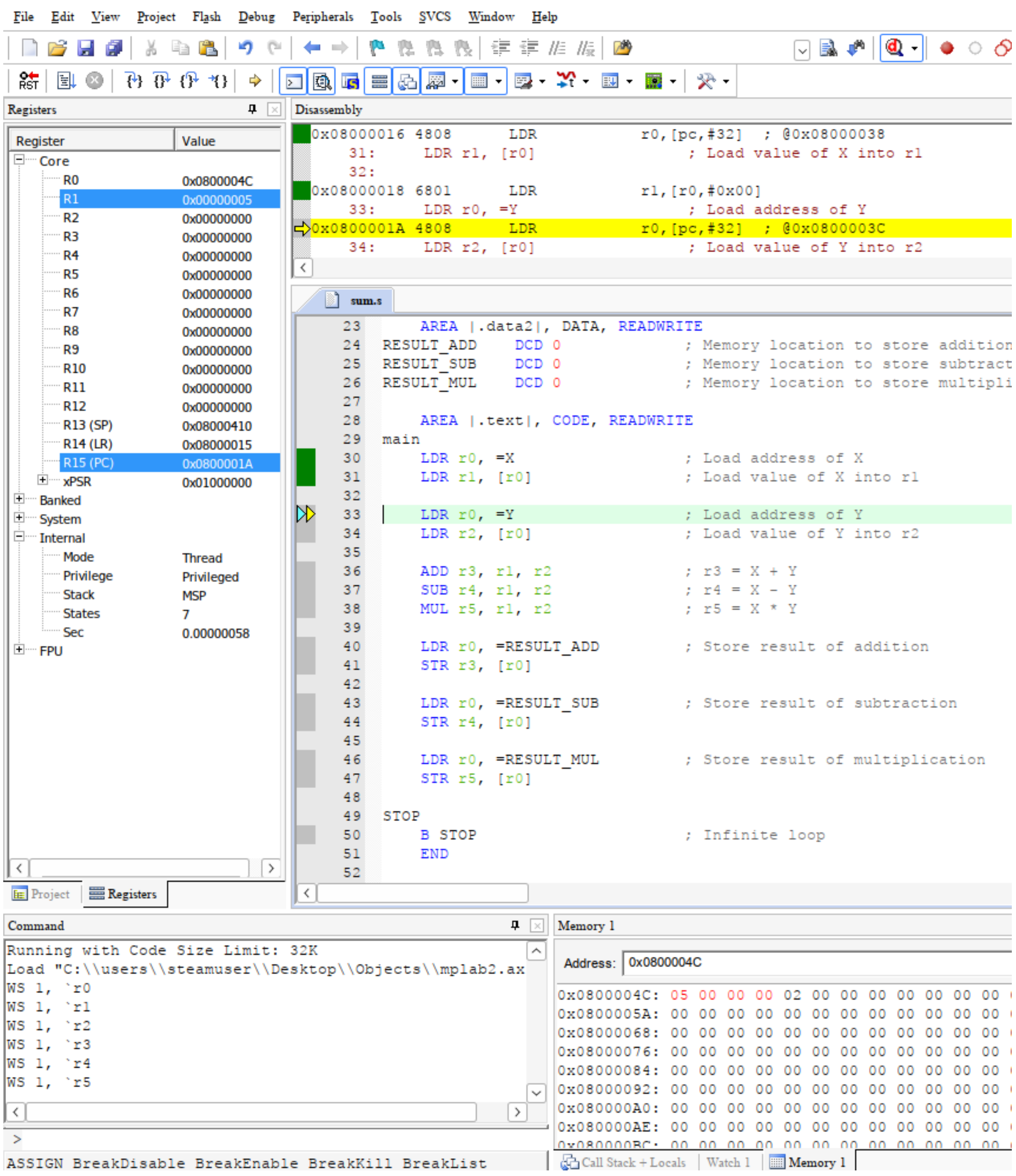
```
X           DCD 0x123                  ; 32-bit constant value for X
            321                       ; 32-bit constant value for Y
            DCD 0                  ; Memory location to store addition
RESULT_SUB  DCD 0                  ; Memory location to store subtract
RESULT_MUL  DCD 0                  ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                     ; Load address of X
    LDR r1, [r0]                   ; Load value of X into r1

    LDR r0, =Y                     ; Load address of Y
    LDR r2, [r0]                   ; Load value of Y into r2

    ADD r3, r1, r2                 ; r3 = X + Y
    SUB r4, r1, r2                 ; r4 = X - Y
    MUL r5, r1, r2                 ; r5 = X * Y

    LDR r0, =RESULT_ADD            ; Store result of addition
    STR r3, [r0]

    LDR r0, =RESULT_SUB            ; Store result of subtraction
    STR r4, [r0]

    LDR r0, =RESULT_MUL            ; Store result of multiplication
    STR r5, [r0]

STOP
    B STOP                         ; Infinite loop
    END
```

C:\users\steamuser\Desktop\sum.s
Project Group: Source Group 1

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x20000000	ulong
r1	0x00000000	ulong
r2	0x00000000	ulong
r3	0x00000000	ulong
r4	0x00000000	ulong
r5	0x00000000	ulong
<Enter expression>		

Figure 12: Loaded the address of X in r0 register, From the **window**(in below right position) view, we can see the value of register content(data)



Register	Value
Core	
R0	0x0800004C
R1	0x00000005
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x0800001A
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	7
Sec	0.00000058
FPU	

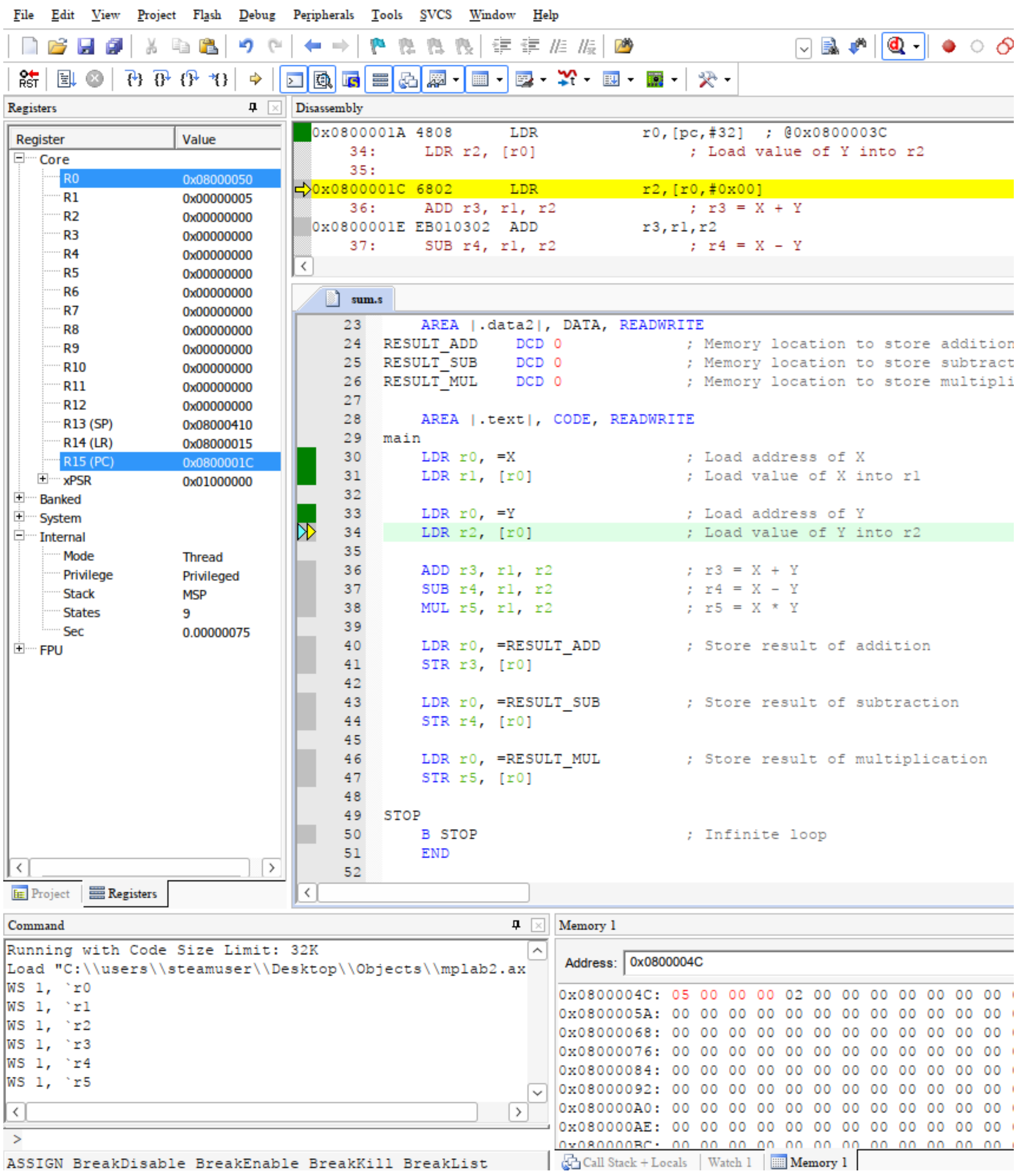
```
0x08000016 4808      LDR          r0,[pc,#32]  ; @0x08000038
    31:      LDR r1, [r0]              ; Load value of X into r1
    32:
0x08000018 6801      LDR          r1,[r0,#0x00]
    33:      LDR r0, =Y                ; Load address of Y
0x0800001A 4808      LDR          r0,[pc,#32]  ; @0x0800003C
    34:      LDR r2, [r0]              ; Load value of Y into r2
```

```
     AREA |.data2|, DATA, READWRITE
RESULT_ADD     DCD 0                    ; Memory location to store addition
RESULT_SUB     DCD 0                    ; Memory location to store subtract
RESULT_MUL     DCD 0                    ; Memory location to store multipli

     AREA |.text|, CODE, READWRITE
main
     LDR r0, =X                         ; Load address of X
     LDR r1, [r0]                       ; Load value of X into r1

     LDR r0, =Y                         ; Load address of Y
     LDR r2, [r0]                       ; Load value of Y into r2

     ADD r3, r1, r2                     ; r3 = X + Y
     SUB r4, r1, r2                     ; r4 = X - Y
     MUL r5, r1, r2                     ; r5 = X * Y

     LDR r0, =RESULT_ADD                ; Store result of addition
     STR r3, [r0]

     LDR r0, =RESULT_SUB                ; Store result of subtraction
     STR r4, [r0]

     LDR r0, =RESULT_MUL                ; Store result of multiplication
     STR r5, [r0]

STOP
     B STOP                             ; Infinite loop
     END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x0800004C

0x0800004C: 05 00 00 00 02 00 00 00 00 00 00 00
0x0800005A: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000068: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000076: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000084: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000092: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000A0: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000AE: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000BC: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 13: Loaded the content of X in r1



Register	Value
Core	
R0	0x08000050
R1	0x00000005
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x0800001C
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	9
Sec	0.00000075
FPU	

```
0x0800001A 4808      LDR         r0,[pc,#32]  ; @0x0800003C
    34:     LDR r2, [r0]              ; Load value of Y into r2
    35:
0x0800001C 6802      LDR         r2,[r0,#0x00]
    36:     ADD r3, r1, r2            ; r3 = X + Y
0x0800001E EB010302  ADD         r3,r1,r2
    37:     SUB r4, r1, r2            ; r4 = X - Y
```

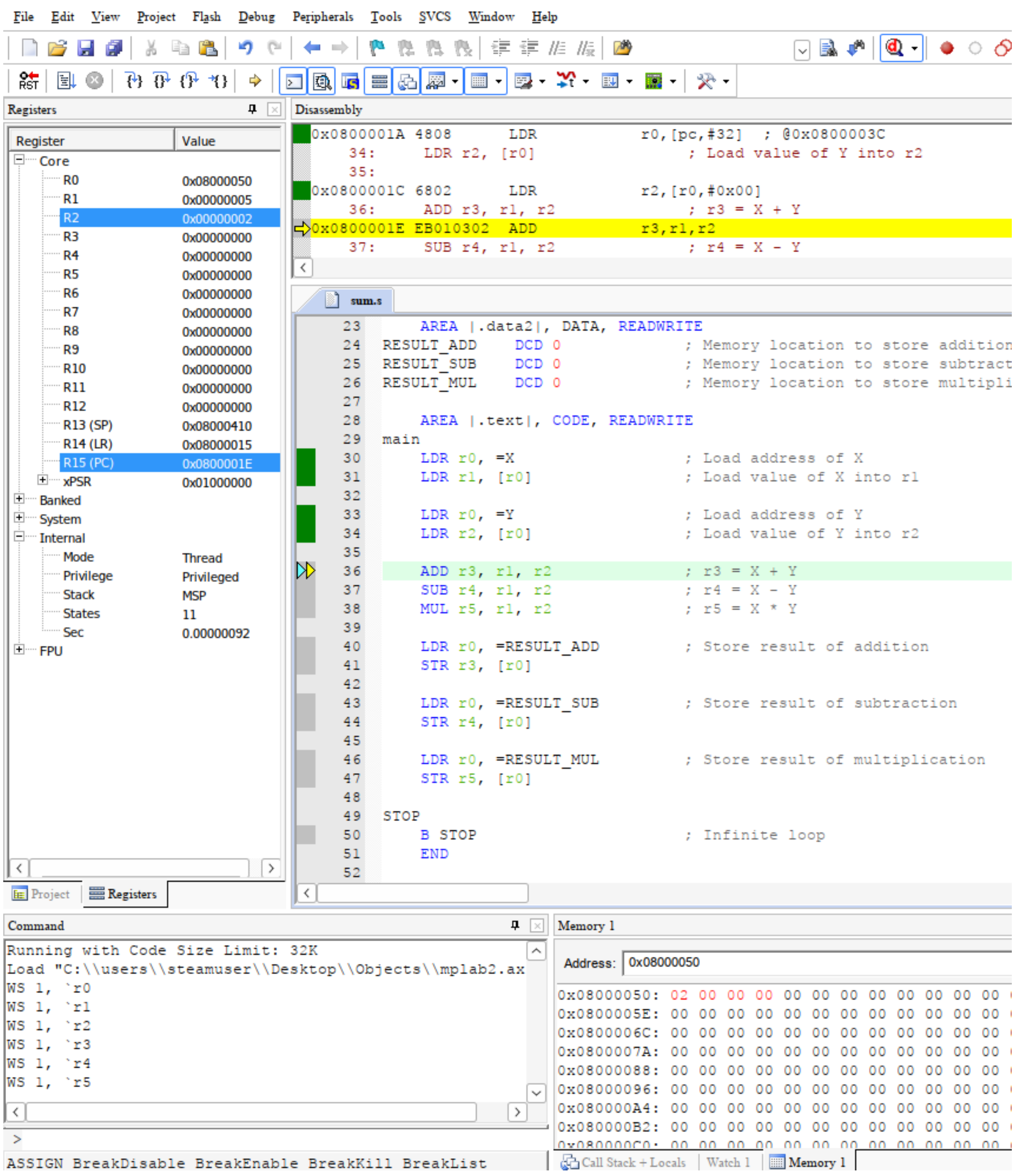
```
 23     AREA |.data2|, DATA, READWRITE
 24 RESULT_ADD     DCD 0              ; Memory location to store addition
 25 RESULT_SUB     DCD 0              ; Memory location to store subtract
 26 RESULT_MUL     DCD 0              ; Memory location to store multipli
 27
 28     AREA |.text|, CODE, READWRITE
 29 main
 30     LDR r0, =X                    ; Load address of X
 31     LDR r1, [r0]                  ; Load value of X into r1
 32
 33     LDR r0, =Y                    ; Load address of Y
 34     LDR r2, [r0]                  ; Load value of Y into r2
 35
 36     ADD r3, r1, r2                ; r3 = X + Y
 37     SUB r4, r1, r2                ; r4 = X - Y
 38     MUL r5, r1, r2                ; r5 = X * Y
 39
 40     LDR r0, =RESULT_ADD           ; Store result of addition
 41     STR r3, [r0]
 42
 43     LDR r0, =RESULT_SUB           ; Store result of subtraction
 44     STR r4, [r0]
 45
 46     LDR r0, =RESULT_MUL           ; Store result of multiplication
 47     STR r5, [r0]
 48
 49 STOP
 50     B STOP                        ; Infinite loop
 51     END
 52
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Address: 0x0800004C

```
0x0800004C: 05 00 00 00 02 00 00 00 00 00 00 00
0x0800005A: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000068: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000076: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000084: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000092: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000A0: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000AE: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000BC: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 14: Loaded the address of Y in r0 register



Register	Value
Core	
R0	0x08000050
R1	0x00000005
R2	0x00000002
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x0800001E
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	11
Sec	0.00000092
FPU	

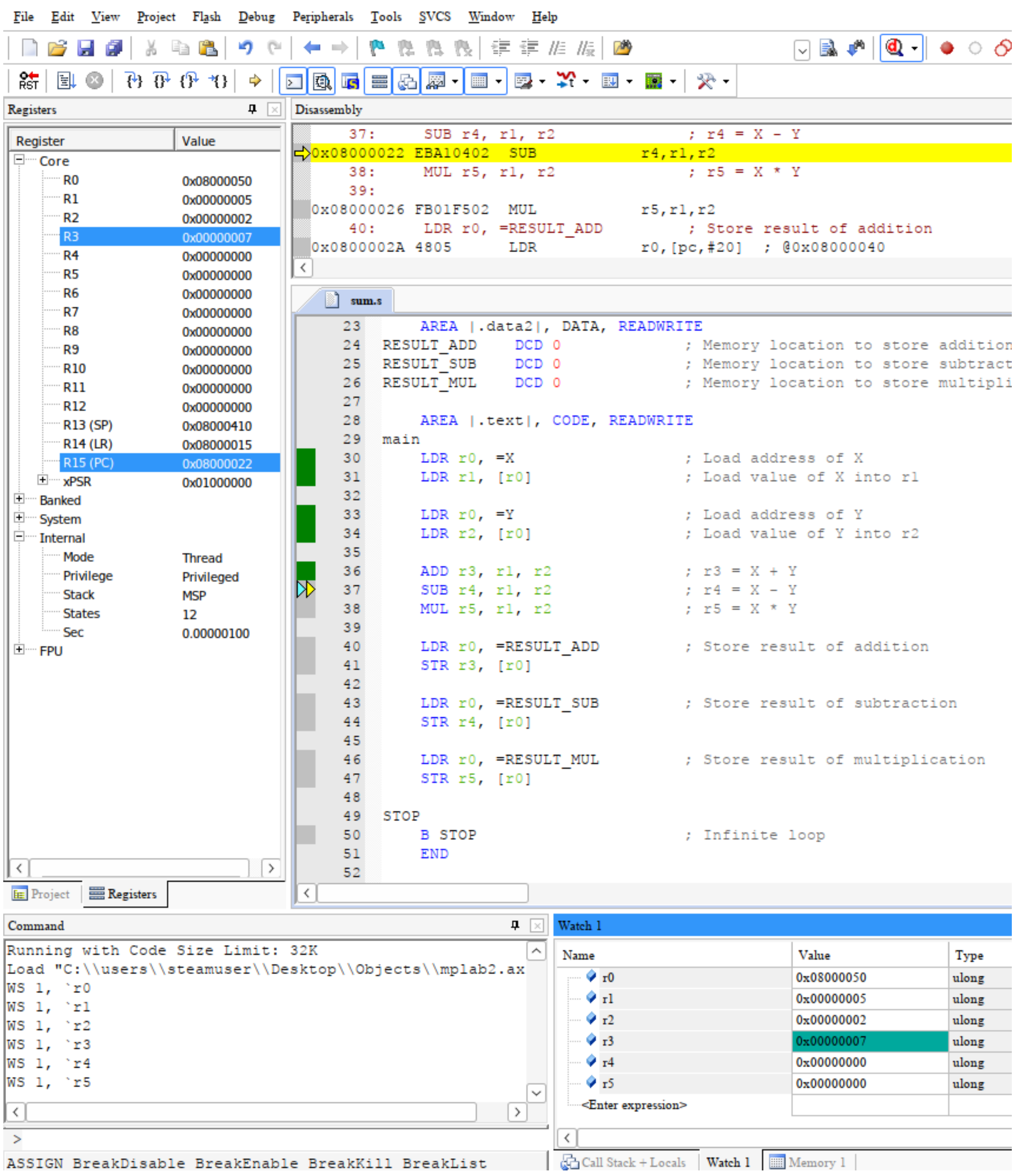
```
0x0800001A 4808      LDR          r0,[pc,#32]  ; @0x0800003C
    34:     LDR r2, [r0]          ; Load value of Y into r2
    35:
0x0800001C 6802      LDR          r2,[r0,#0x00]
    36:     ADD r3, r1, r2        ; r3 = X + Y
0x0800001E EB010302  ADD          r3,r1,r2
    37:     SUB r4, r1, r2        ; r4 = X - Y
```

```
    AREA |.data2|, DATA, READWRITE
RESULT_ADD     DCD 0               ; Memory location to store addition
RESULT_SUB     DCD 0               ; Memory location to store subtract
RESULT_MUL     DCD 0               ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                     ; Load address of X
    LDR r1, [r0]                   ; Load value of X into r1

    LDR r0, =Y                     ; Load address of Y
    LDR r2, [r0]                   ; Load value of Y into r2

    ADD r3, r1, r2                 ; r3 = X + Y
    SUB r4, r1, r2                 ; r4 = X - Y
    MUL r5, r1, r2                 ; r5 = X * Y

    LDR r0, =RESULT_ADD            ; Store result of addition
    STR r3, [r0]

    LDR r0, =RESULT_SUB            ; Store result of subtraction
    STR r4, [r0]

    LDR r0, =RESULT_MUL            ; Store result of multiplication
    STR r5, [r0]

STOP
    B STOP                         ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x08000050

0x08000050: 02 00 00 00 00 00 00 00 00 00 00 00
0x0800005E: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800006C: 00 00 00 00 00 00 00 00 00 00 00 00
0x0800007A: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000088: 00 00 00 00 00 00 00 00 00 00 00 00
0x08000096: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000A4: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000B2: 00 00 00 00 00 00 00 00 00 00 00 00
0x080000C0: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 15: Loaded the content of Y in r2



Register	Value
Core	
R0	0x08000050
R1	0x00000005
R2	0x00000002
R3	0x00000007
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000022
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	12
Sec	0.00000100
FPU	

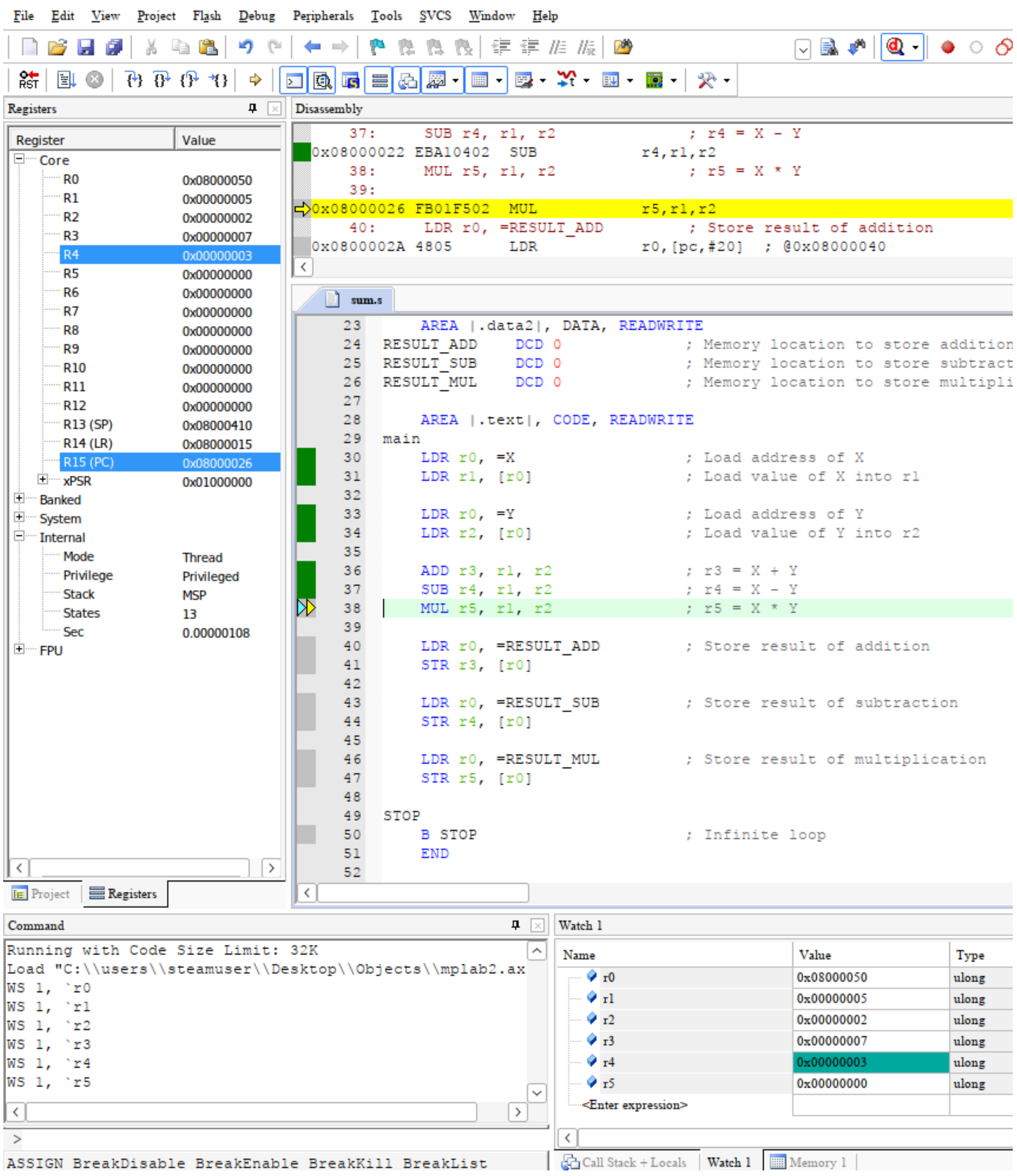
```
    37:     SUB r4, r1, r2                  ; r4 = X - Y
0x08000022 EBA10402  SUB          r4,r1,r2
    38:     MUL r5, r1, r2                  ; r5 = X * Y
    39:
0x08000026 FB01F502  MUL          r5,r1,r2
    40:     LDR r0, =RESULT_ADD             ; Store result of addition
0x0800002A 4805      LDR          r0,[pc,#20]  ; @0x08000040
```

```
     AREA |.data2|, DATA, READWRITE
RESULT_ADD     DCD 0                     ; Memory location to store addition
RESULT_SUB     DCD 0                     ; Memory location to store subtract
RESULT_MUL     DCD 0                     ; Memory location to store multipli

     AREA |.text|, CODE, READWRITE
main
     LDR r0, =X                          ; Load address of X
     LDR r1, [r0]                        ; Load value of X into r1

     LDR r0, =Y                          ; Load address of Y
     LDR r2, [r0]                        ; Load value of Y into r2

     ADD r3, r1, r2                      ; r3 = X + Y
     SUB r4, r1, r2                      ; r4 = X - Y
     MUL r5, r1, r2                      ; r5 = X * Y

     LDR r0, =RESULT_ADD                 ; Store result of addition
     STR r3, [r0]

     LDR r0, =RESULT_SUB                 ; Store result of subtraction
     STR r4, [r0]

     LDR r0, =RESULT_MUL                 ; Store result of multiplication
     STR r5, [r0]

STOP
     B STOP                              ; Infinite loop
     END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x08000050	ulong
r1	0x00000005	ulong
r2	0x00000002	ulong
r3	0x00000007	ulong
r4	0x00000000	ulong
r5	0x00000000	ulong
<Enter expression>		

Figure 16: Added the value of r1, r2 register and placed it in r3 register



Registers

Register	Value
Core	
R0	0x08000050
R1	0x00000005
R2	0x00000002
R3	0x00000007
R4	0x00000003
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000026
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	13
Sec	0.00000108
FPU	

Disassembly

```
    37:     SUB r4, r1, r2              ; r4 = X - Y
0x08000022 EBA10402  SUB          r4,r1,r2
    38:     MUL r5, r1, r2              ; r5 = X * Y
    39:
0x08000026 FB01F502  MUL          r5,r1,r2
    40:     LDR r0, =RESULT_ADD         ; Store result of addition
0x0800002A 4805      LDR          r0,[pc,#20]  ; @0x08000040
```

sum.s

```
    AREA |.data2|, DATA, READWRITE
RESULT_ADD    DCD 0                   ; Memory location to store addition
RESULT_SUB    DCD 0                   ; Memory location to store subtract
RESULT_MUL    DCD 0                   ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                        ; Load address of X
    LDR r1, [r0]                      ; Load value of X into r1

    LDR r0, =Y                        ; Load address of Y
    LDR r2, [r0]                      ; Load value of Y into r2

    ADD r3, r1, r2                    ; r3 = X + Y
    SUB r4, r1, r2                    ; r4 = X - Y
    MUL r5, r1, r2                    ; r5 = X * Y

    LDR r0, =RESULT_ADD               ; Store result of addition
    STR r3, [r0]

    LDR r0, =RESULT_SUB               ; Store result of subtraction
    STR r4, [r0]

    LDR r0, =RESULT_MUL               ; Store result of multiplication
    STR r5, [r0]

STOP
    B STOP                            ; Infinite loop
    END
```

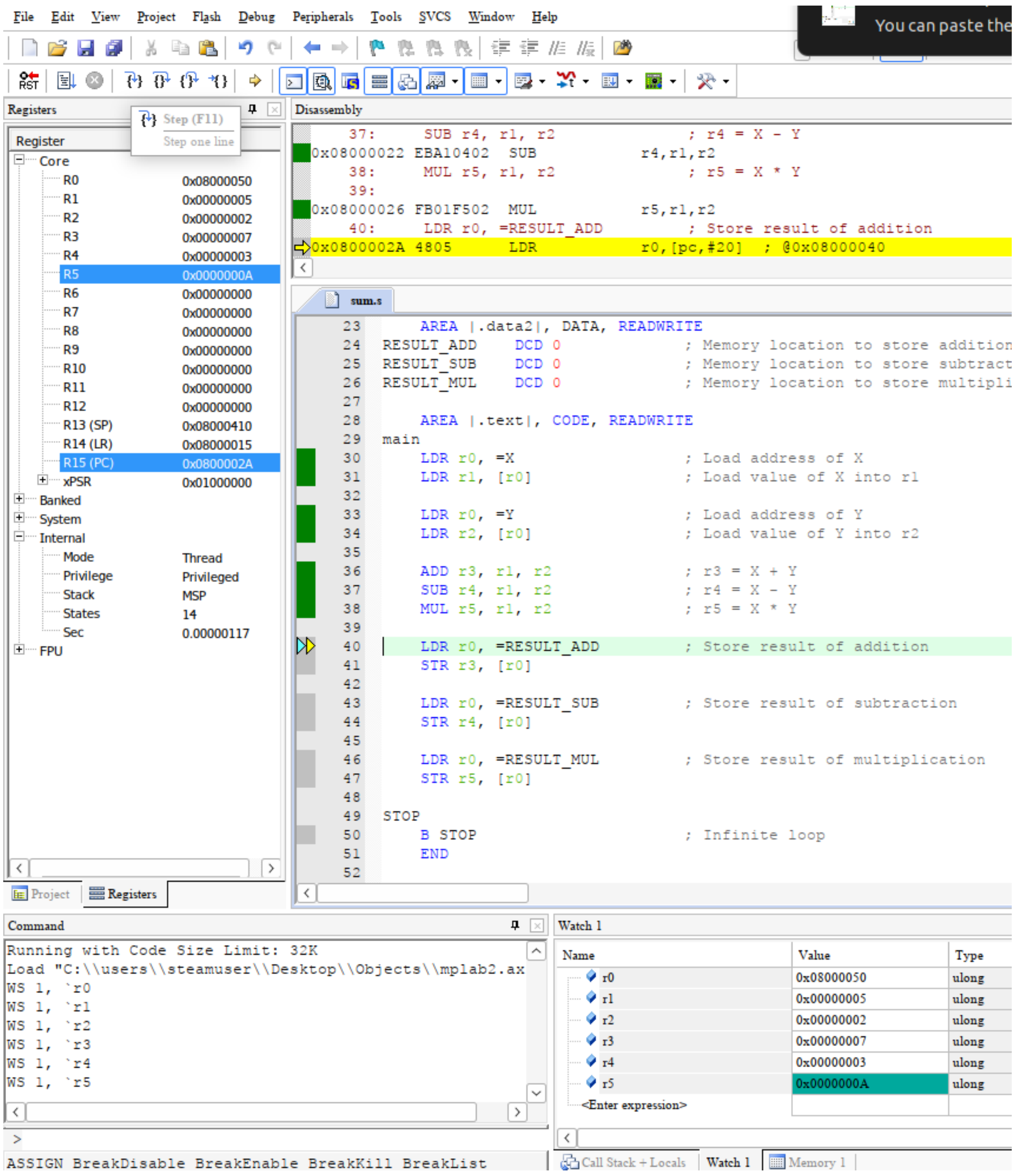
Command

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Watch 1

Name	Value	Type
r0	0x08000050	ulong
r1	0x00000005	ulong
r2	0x00000002	ulong
r3	0x00000007	ulong
r4	0x00000003	ulong
r5	0x00000000	ulong
<Enter expression>		

Figure 17: Subtracted the value of r1, r2 register and placed it in r4 register



Register	Value
Core	
R0	0x08000050
R1	0x00000005
R2	0x00000002
R3	0x00000007
R4	0x00000003
R5	0x0000000A
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x0800002A
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	14
Sec	0.00000117
FPU	

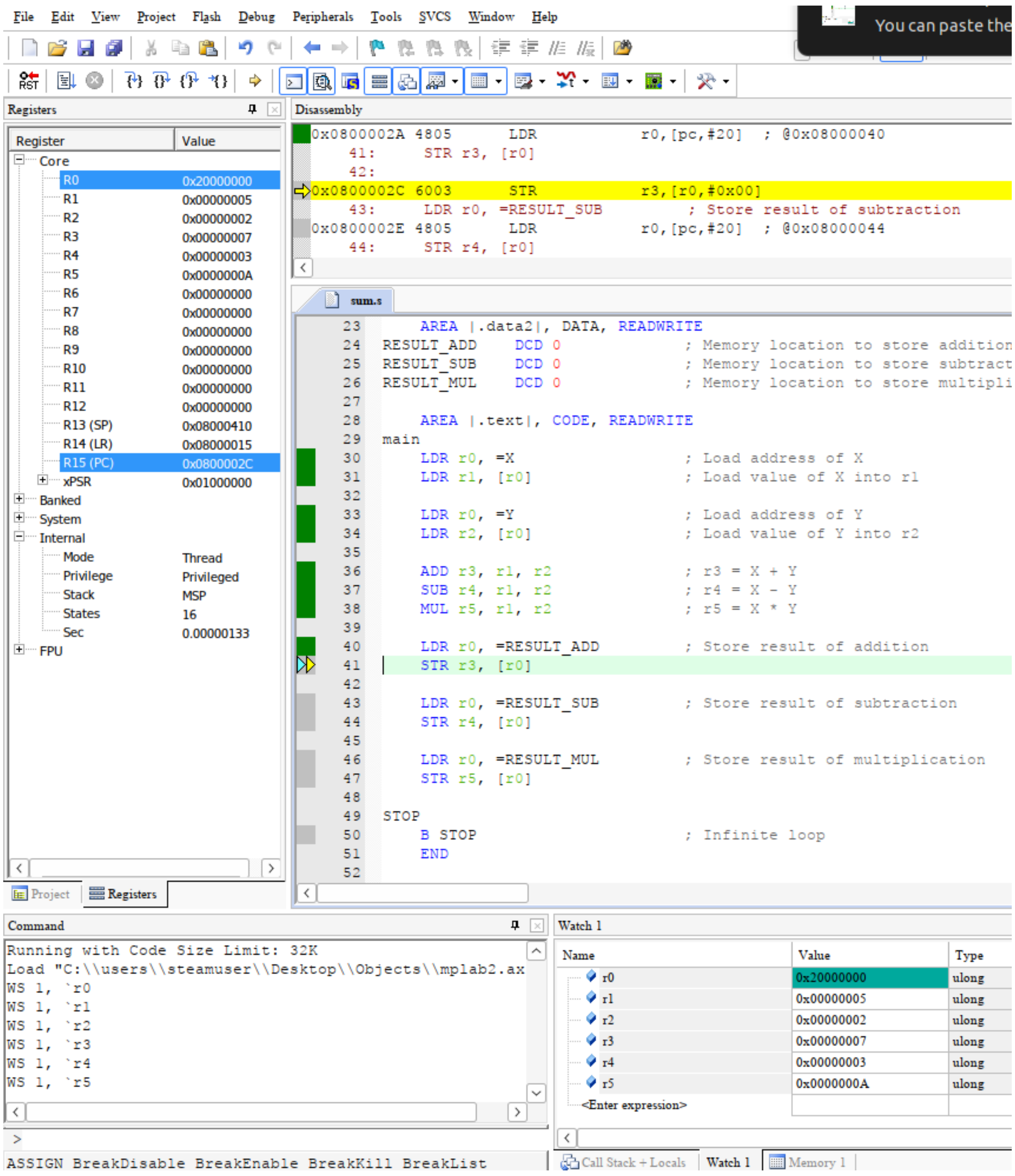
```
    37:     SUB r4, r1, r2              ; r4 = X - Y
0x08000022 EBA10402  SUB          r4,r1,r2
    38:     MUL r5, r1, r2              ; r5 = X * Y
    39:
0x08000026 FB01F502  MUL          r5,r1,r2
    40:     LDR r0, =RESULT_ADD         ; Store result of addition
0x0800002A 4805      LDR          r0,[pc,#20]  ; @0x08000040
```

```
        AREA |.data2|, DATA, READWRITE
RESULT_ADD     DCD 0                    ; Memory location to store addition
RESULT_SUB     DCD 0                    ; Memory location to store subtract
RESULT_MUL     DCD 0                    ; Memory location to store multipli

        AREA |.text|, CODE, READWRITE
main
        LDR r0, =X                      ; Load address of X
        LDR r1, [r0]                    ; Load value of X into r1

        LDR r0, =Y                      ; Load address of Y
        LDR r2, [r0]                    ; Load value of Y into r2

        ADD r3, r1, r2                  ; r3 = X + Y
        SUB r4, r1, r2                  ; r4 = X - Y
        MUL r5, r1, r2                  ; r5 = X * Y

        LDR r0, =RESULT_ADD             ; Store result of addition
        STR r3, [r0]

        LDR r0, =RESULT_SUB             ; Store result of subtraction
        STR r4, [r0]

        LDR r0, =RESULT_MUL             ; Store result of multiplication
        STR r5, [r0]

STOP
        B STOP                          ; Infinite loop
        END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x08000050	ulong
r1	0x00000005	ulong
r2	0x00000002	ulong
r3	0x00000007	ulong
r4	0x00000003	ulong
r5	0x0000000A	ulong
<Enter expression>		

Figure 18: Multiply the value of r1, r2 register and placed it in r5 register


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
You can paste the

Registers

Register	Value
Core	
R0	0x20000000
R1	0x00000005
R2	0x00000002
R3	0x00000007
R4	0x00000003
R5	0x0000000A
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x0800002C
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	16
Sec	0.00000133
FPU	

Disassembly

```
0x0800002A 4805      LDR          r0,[pc,#20]  ; @0x08000040
    41:      STR r3, [r0]
    42:
0x0800002C 6003      STR          r3,[r0,#0x00]
    43:      LDR r0, =RESULT_SUB          ; Store result of subtraction
0x0800002E 4805      LDR          r0,[pc,#20]  ; @0x08000044
    44:      STR r4, [r0]
```

sum.s

```
23      AREA |.data2|, DATA, READWRITE
24  RESULT_ADD     DCD 0                  ; Memory location to store addition
25  RESULT_SUB     DCD 0                  ; Memory location to store subtract
26  RESULT_MUL     DCD 0                  ; Memory location to store multipli
27
28      AREA |.text|, CODE, READWRITE
29  main
30      LDR r0, =X                        ; Load address of X
31      LDR r1, [r0]                      ; Load value of X into r1
32
33      LDR r0, =Y                        ; Load address of Y
34      LDR r2, [r0]                      ; Load value of Y into r2
35
36      ADD r3, r1, r2                    ; r3 = X + Y
37      SUB r4, r1, r2                    ; r4 = X - Y
38      MUL r5, r1, r2                    ; r5 = X * Y
39
40      LDR r0, =RESULT_ADD               ; Store result of addition
41      STR r3, [r0]
42
43      LDR r0, =RESULT_SUB               ; Store result of subtraction
44      STR r4, [r0]
45
46      LDR r0, =RESULT_MUL               ; Store result of multiplication
47      STR r5, [r0]
48
49  STOP
50      B STOP                            ; Infinite loop
51      END
52
```

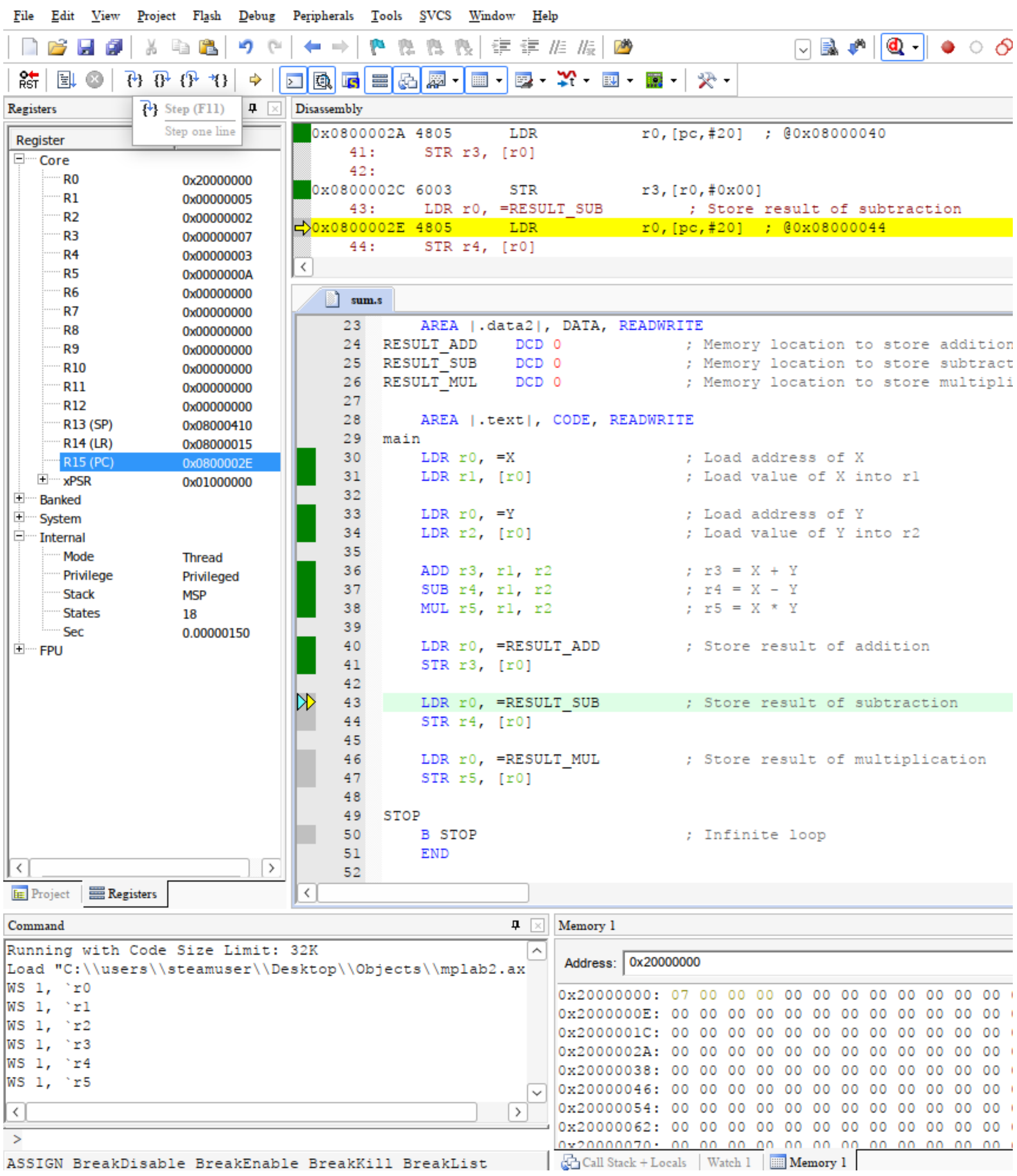
Command

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Watch 1

Name	Value	Type
r0	0x20000000	ulong
r1	0x00000005	ulong
r2	0x00000002	ulong
r3	0x00000007	ulong
r4	0x00000003	ulong
r5	0x0000000A	ulong
<Enter expression>		

Figure 19: Loaded the address of Result_add in r0 register



```
Register
Core
  R0         0x20000000
  R1         0x00000005
  R2         0x00000002
  R3         0x00000007
  R4         0x00000003
  R5         0x0000000A
  R6         0x00000000
  R7         0x00000000
  R8         0x00000000
  R9         0x00000000
  R10        0x00000000
  R11        0x00000000
  R12        0x00000000
  R13 (SP)   0x08000410
  R14 (LR)   0x08000015
  R15 (PC)   0x0800002E
  xPSR       0x01000000
Banked
System
Internal
  Mode       Thread
  Privilege  Privileged
  Stack      MSP
  States     18
  Sec        0.00000150
FPU
```

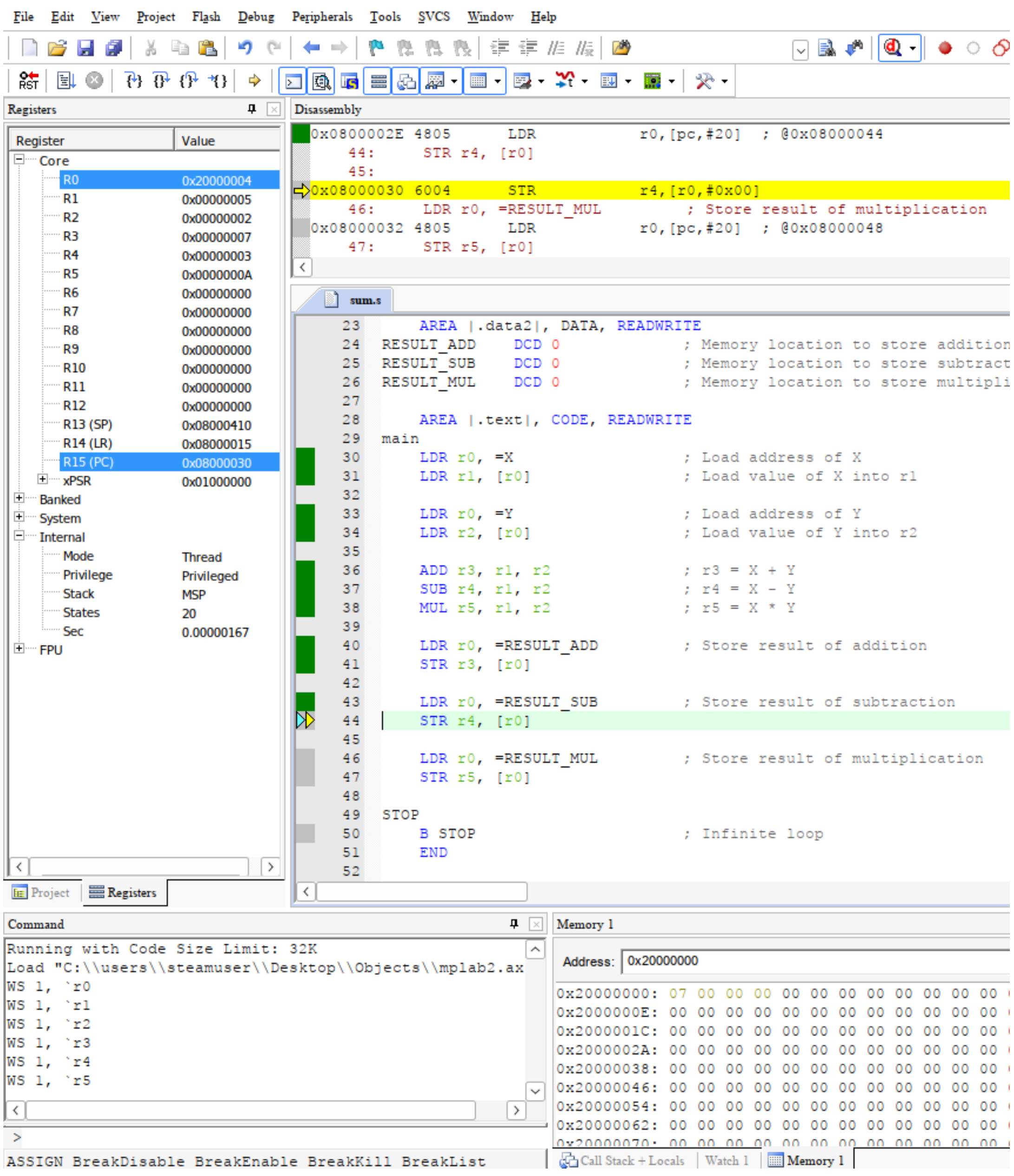
```
0x0800002A 4805      LDR          r0,[pc,#20]  ; @0x08000040
    41:     STR r3, [r0]
    42:
0x0800002C 6003      STR          r3,[r0,#0x00]
    43:     LDR r0, =RESULT_SUB         ; Store result of subtraction
0x0800002E 4805      LDR          r0,[pc,#20]  ; @0x08000044
    44:     STR r4, [r0]
```

```
    AREA |.data2|, DATA, READWRITE
RESULT_ADD     DCD 0                   ; Memory location to store addition
RESULT_SUB     DCD 0                   ; Memory location to store subtract
RESULT_MUL     DCD 0                   ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                         ; Load address of X
    LDR r1, [r0]                       ; Load value of X into r1

    LDR r0, =Y                         ; Load address of Y
    LDR r2, [r0]                       ; Load value of Y into r2

    ADD r3, r1, r2                     ; r3 = X + Y
    SUB r4, r1, r2                     ; r4 = X - Y
    MUL r5, r1, r2                     ; r5 = X * Y

    LDR r0, =RESULT_ADD                ; Store result of addition
    STR r3, [r0]

    LDR r0, =RESULT_SUB                ; Store result of subtraction
    STR r4, [r0]

    LDR r0, =RESULT_MUL                ; Store result of multiplication
    STR r5, [r0]

STOP
    B STOP                             ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x20000000

0x20000000: 07 00 00 00 00 00 00 00 00 00 00 00
0x2000000E: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000001C: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000002A: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000038: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000046: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000054: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000062: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000070: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 20: Store the value of r3 in Result_add address location



Register	Value
Core	
R0	0x20000004
R1	0x00000005
R2	0x00000002
R3	0x00000007
R4	0x00000003
R5	0x0000000A
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000030
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	20
Sec	0.00000167
FPU	

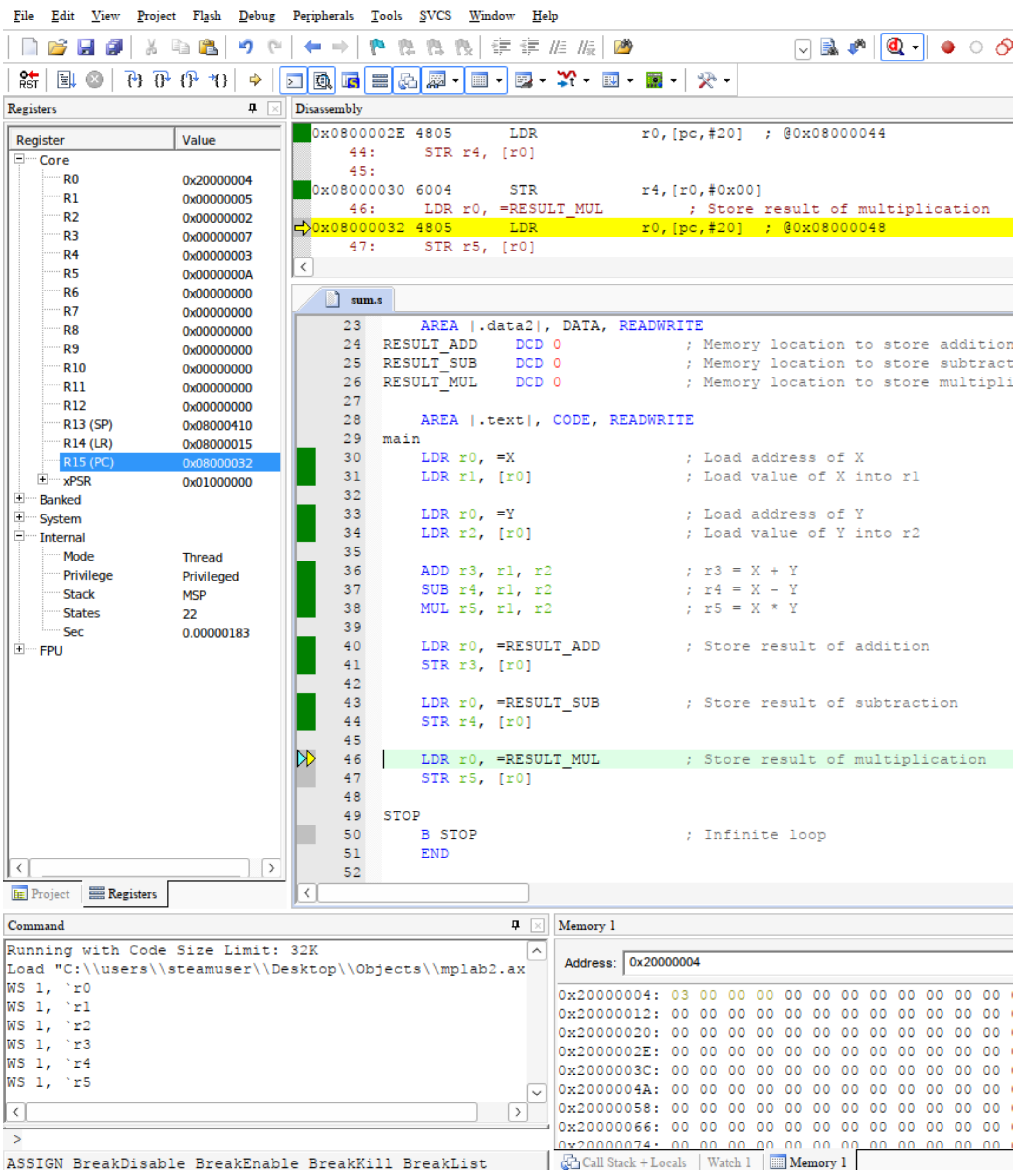
```
0x0800002E 4805      LDR          r0,[pc,#20]  ; @0x08000044
    44:      STR r4, [r0]
    45:
0x08000030 6004      STR          r4,[r0,#0x00]
    46:      LDR r0, =RESULT_MUL         ; Store result of multiplication
0x08000032 4805      LDR          r0,[pc,#20]  ; @0x08000048
    47:      STR r5, [r0]
```

```
    AREA |.data2|, DATA, READWRITE
RESULT_ADD      DCD 0                   ; Memory location to store addition
RESULT_SUB      DCD 0                   ; Memory location to store subtract
RESULT_MUL      DCD 0                   ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                          ; Load address of X
    LDR r1, [r0]                        ; Load value of X into r1

    LDR r0, =Y                          ; Load address of Y
    LDR r2, [r0]                        ; Load value of Y into r2

    ADD r3, r1, r2                      ; r3 = X + Y
    SUB r4, r1, r2                      ; r4 = X - Y
    MUL r5, r1, r2                      ; r5 = X * Y

    LDR r0, =RESULT_ADD                 ; Store result of addition
    STR r3, [r0]

    LDR r0, =RESULT_SUB                 ; Store result of subtraction
    STR r4, [r0]

    LDR r0, =RESULT_MUL                 ; Store result of multiplication
    STR r5, [r0]

STOP
    B STOP                              ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x20000000

0x20000000: 07 00 00 00 00 00 00 00 00 00 00 00
0x2000000E: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000001C: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000002A: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000038: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000046: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000054: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000062: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000070: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 21: Loaded the address of Result_sub in r0 register



Register	Value
Core	
R0	0x20000004
R1	0x00000005
R2	0x00000002
R3	0x00000007
R4	0x00000003
R5	0x0000000A
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000032
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	22
Sec	0.00000183
FPU	

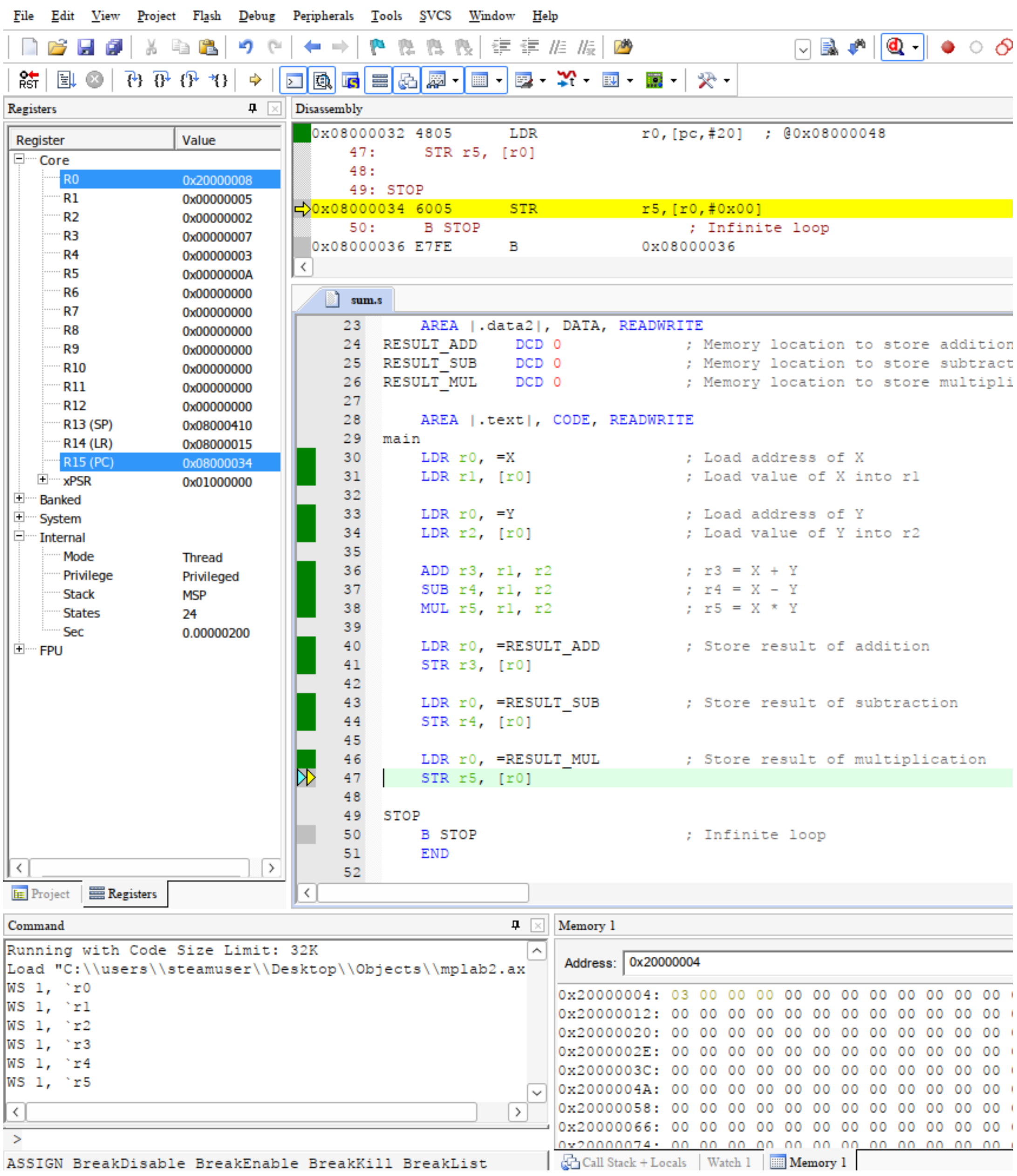
```
0x0800002E 4805      LDR        r0,[pc,#20]  ; @0x08000044
    44:     STR r4, [r0]
    45:
0x08000030 6004      STR        r4,[r0,#0x00]
    46:     LDR r0, =RESULT_MUL           ; Store result of multiplication
0x08000032 4805      LDR        r0,[pc,#20]  ; @0x08000048
    47:     STR r5, [r0]
```

```
    AREA |.data2|, DATA, READWRITE
RESULT_ADD     DCD 0                ; Memory location to store addition
RESULT_SUB     DCD 0                ; Memory location to store subtract
RESULT_MUL     DCD 0                ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                      ; Load address of X
    LDR r1, [r0]                    ; Load value of X into r1

    LDR r0, =Y                      ; Load address of Y
    LDR r2, [r0]                    ; Load value of Y into r2

    ADD r3, r1, r2                  ; r3 = X + Y
    SUB r4, r1, r2                  ; r4 = X - Y
    MUL r5, r1, r2                  ; r5 = X * Y

    LDR r0, =RESULT_ADD             ; Store result of addition
    STR r3, [r0]

    LDR r0, =RESULT_SUB             ; Store result of subtraction
    STR r4, [r0]

    LDR r0, =RESULT_MUL             ; Store result of multiplication
    STR r5, [r0]

STOP
    B STOP                          ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x20000004

0x20000004: 03 00 00 00 00 00 00 00 00 00 00 00
0x20000012: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000020: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000002E: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000003C: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000004A: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000058: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000066: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 22: Store the value of r4 in Result_sub address location



Register	Value
Core	
R0	0x20000008
R1	0x00000005
R2	0x00000002
R3	0x00000007
R4	0x00000003
R5	0x0000000A
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000034
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	24
Sec	0.00000200
FPU	

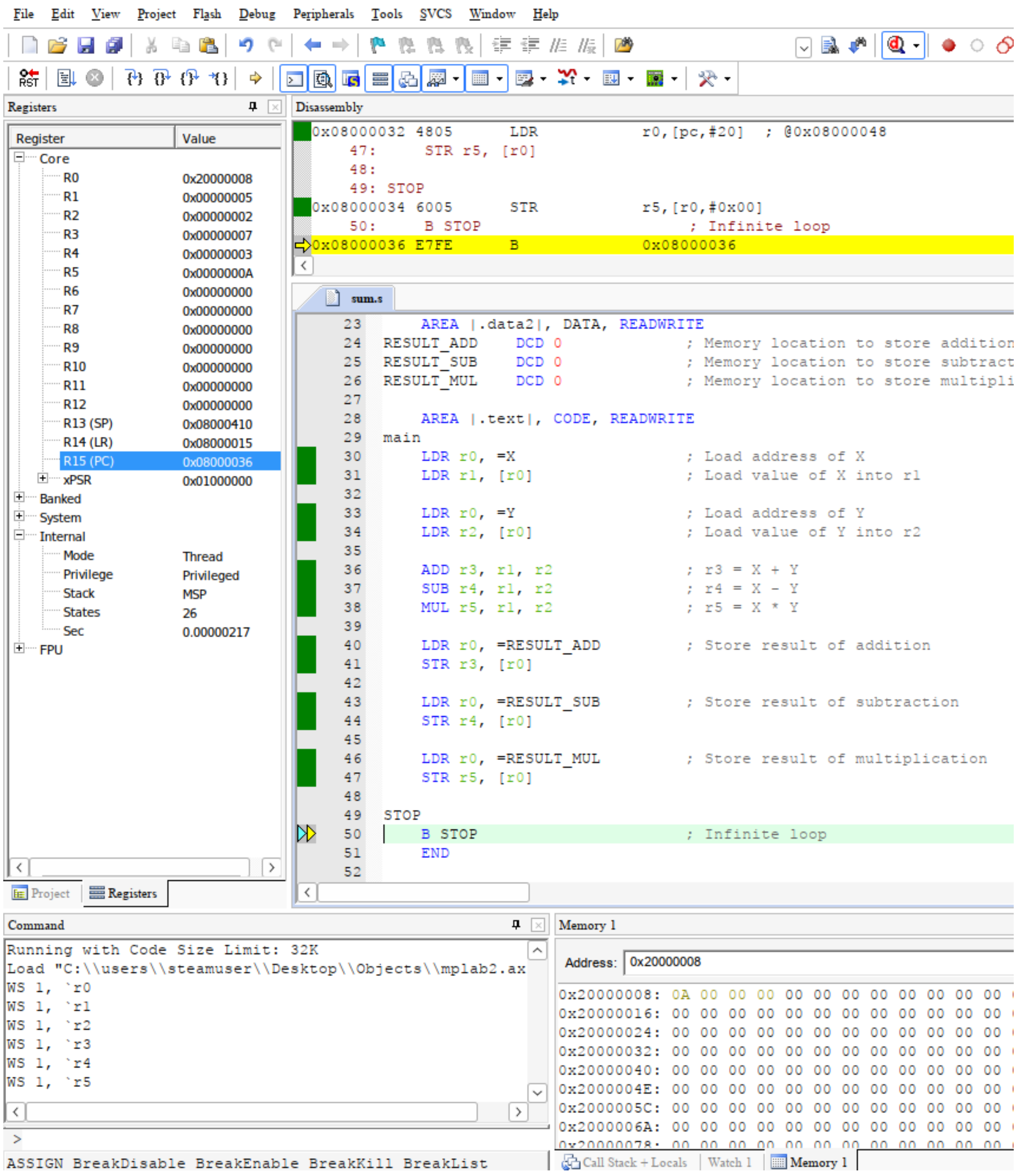
```
0x08000032 4805      LDR        r0,[pc,#20]  ; @0x08000048
    47:      STR r5, [r0]
    48:
    49: STOP
0x08000034 6005      STR        r5,[r0,#0x00]
    50:       B STOP                      ; Infinite loop
0x08000036 E7FE      B          0x08000036
```

```
  23        AREA |.data2|, DATA, READWRITE
  24    RESULT_ADD      DCD 0             ; Memory location to store addition
  25    RESULT_SUB      DCD 0             ; Memory location to store subtract
  26    RESULT_MUL      DCD 0             ; Memory location to store multipli
  27
  28        AREA |.text|, CODE, READWRITE
  29    main
  30        LDR r0, =X                    ; Load address of X
  31        LDR r1, [r0]                  ; Load value of X into r1
  32
  33        LDR r0, =Y                    ; Load address of Y
  34        LDR r2, [r0]                  ; Load value of Y into r2
  35
  36        ADD r3, r1, r2                ; r3 = X + Y
  37        SUB r4, r1, r2                ; r4 = X - Y
  38        MUL r5, r1, r2                ; r5 = X * Y
  39
  40        LDR r0, =RESULT_ADD           ; Store result of addition
  41        STR r3, [r0]
  42
  43        LDR r0, =RESULT_SUB           ; Store result of subtraction
  44        STR r4, [r0]
  45
  46        LDR r0, =RESULT_MUL           ; Store result of multiplication
  47        STR r5, [r0]
  48
  49    STOP
  50        B STOP                        ; Infinite loop
  51        END
  52
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x20000004

0x20000004: 03 00 00 00 00 00 00 00 00 00 00 00
0x20000012: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000020: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000002E: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000003C: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000004A: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000058: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000066: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 23: Loaded the address of Result_mul in r0 register



Register	Value
Core	
R0	0x20000008
R1	0x00000005
R2	0x00000002
R3	0x00000007
R4	0x00000003
R5	0x0000000A
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000036
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	26
Sec	0.00000217
FPU	

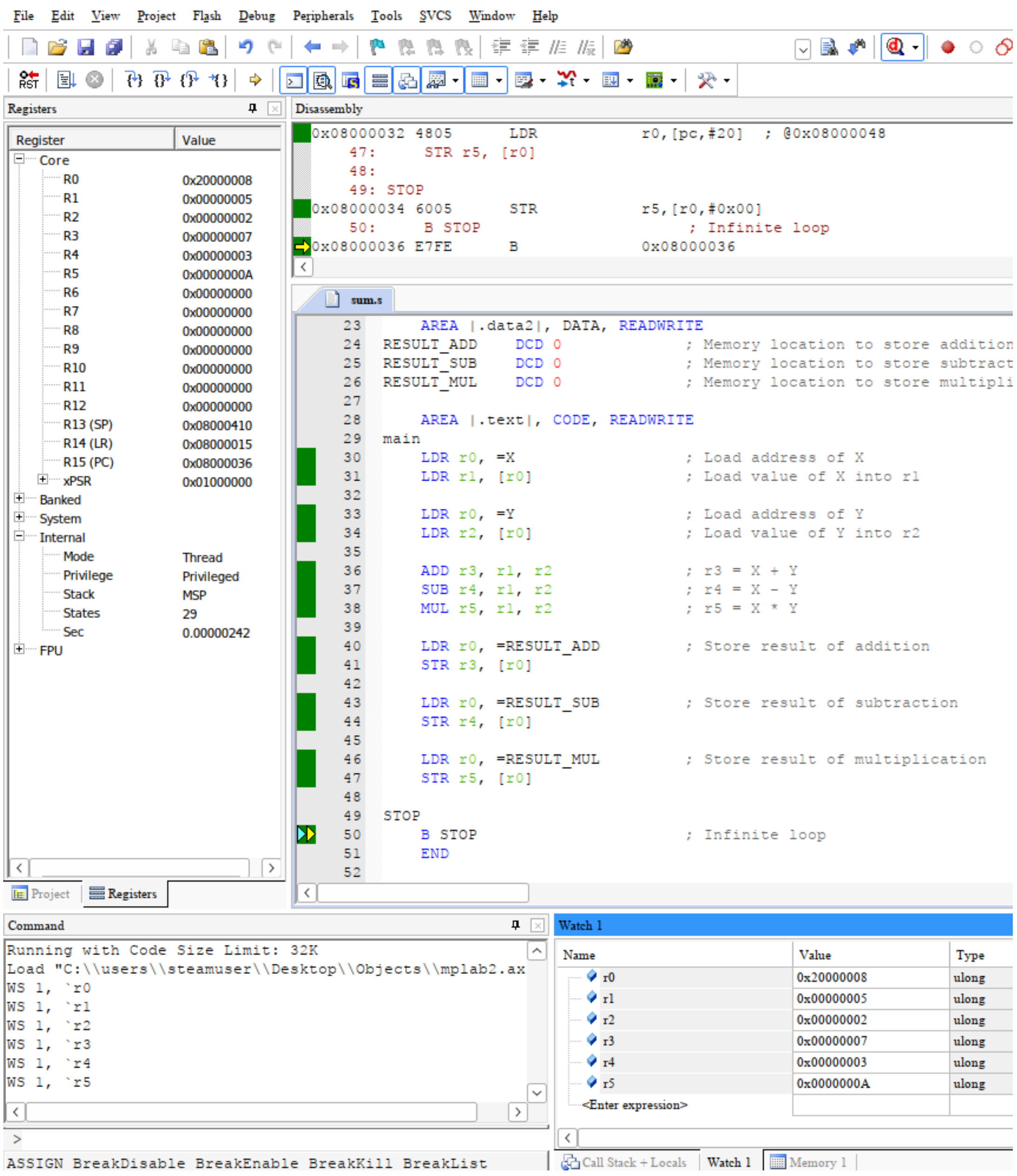
```
0x08000032 4805      LDR          r0,[pc,#20]  ; @0x08000048
    47:      STR r5, [r0]
    48:
    49: STOP
0x08000034 6005      STR          r5,[r0,#0x00]
    50:      B STOP                    ; Infinite loop
0x08000036 E7FE      B            0x08000036
```

```
    AREA |.data2|, DATA, READWRITE
RESULT_ADD     DCD 0                 ; Memory location to store addition
RESULT_SUB     DCD 0                 ; Memory location to store subtract
RESULT_MUL     DCD 0                 ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                       ; Load address of X
    LDR r1, [r0]                     ; Load value of X into r1

    LDR r0, =Y                       ; Load address of Y
    LDR r2, [r0]                     ; Load value of Y into r2

    ADD r3, r1, r2                   ; r3 = X + Y
    SUB r4, r1, r2                   ; r4 = X - Y
    MUL r5, r1, r2                   ; r5 = X * Y

    LDR r0, =RESULT_ADD              ; Store result of addition
    STR r3, [r0]

    LDR r0, =RESULT_SUB              ; Store result of subtraction
    STR r4, [r0]

    LDR r0, =RESULT_MUL              ; Store result of multiplication
    STR r5, [r0]

STOP
    B STOP                           ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Address: 0x20000008
0x20000008: 0A 00 00 00 00 00 00 00 00 00 00 00
0x20000016: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000024: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000032: 00 00 00 00 00 00 00 00 00 00 00 00
0x20000040: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000004E: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000005C: 00 00 00 00 00 00 00 00 00 00 00 00
0x2000006A: 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
```

Figure 24: Store the value of r3 in Result_mul address location



Register	Value
Core	
R0	0x20000008
R1	0x00000005
R2	0x00000002
R3	0x00000007
R4	0x00000003
R5	0x0000000A
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000410
R14 (LR)	0x08000015
R15 (PC)	0x08000036
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	29
Sec	0.00000242
FPU	

```
0x08000032 4805      LDR          r0,[pc,#20]  ; @0x08000048
    47:      STR r5, [r0]
    48:
    49: STOP
0x08000034 6005      STR          r5,[r0,#0x00]
    50:      B STOP                   ; Infinite loop
0x08000036 E7FE      B            0x08000036
```

```
    AREA |.data2|, DATA, READWRITE
RESULT_ADD     DCD 0                    ; Memory location to store addition
RESULT_SUB     DCD 0                    ; Memory location to store subtract
RESULT_MUL     DCD 0                    ; Memory location to store multipli

    AREA |.text|, CODE, READWRITE
main
    LDR r0, =X                          ; Load address of X
    LDR r1, [r0]                        ; Load value of X into r1

    LDR r0, =Y                          ; Load address of Y
    LDR r2, [r0]                        ; Load value of Y into r2

    ADD r3, r1, r2                      ; r3 = X + Y
    SUB r4, r1, r2                      ; r4 = X - Y
    MUL r5, r1, r2                      ; r5 = X * Y

    LDR r0, =RESULT_ADD                 ; Store result of addition
    STR r3, [r0]

    LDR r0, =RESULT_SUB                 ; Store result of subtraction
    STR r4, [r0]

    LDR r0, =RESULT_MUL                 ; Store result of multiplication
    STR r5, [r0]

STOP
    B STOP                              ; Infinite loop
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r1
WS 1, `r2
WS 1, `r3
WS 1, `r4
WS 1, `r5
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

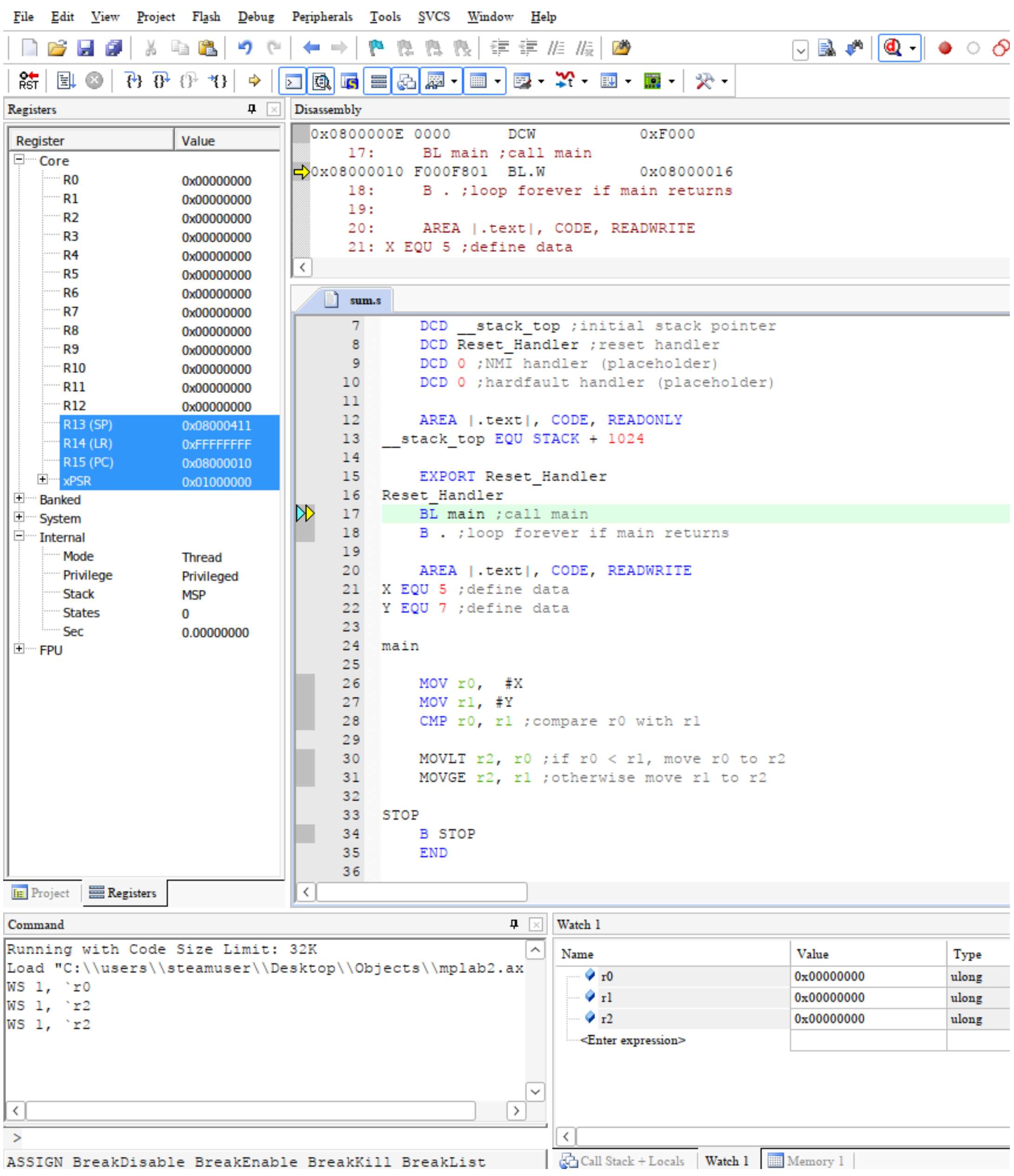
Name	Value	Type
r0	0x20000008	ulong
r1	0x00000005	ulong
r2	0x00000002	ulong
r3	0x00000007	ulong
r4	0x00000003	ulong
r5	0x0000000A	ulong
<Enter expression>		

Figure 25: Stop position

1.3 Task 3

Write an assembly language to find the smaller of two integer numbers.

1.3.1 Screenshot that shows the state of the system after the code has been loaded.



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000411
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000010
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000
FPU	

```
0x0800000E 0000      DCW          0xF000
    17:      BL main ;call main
0x08000010 F000F801  BL.W         0x08000016
    18:      B . ;loop forever if main returns
    19:
    20:      AREA |.text|, CODE, READWRITE
    21: X EQU 5 ;define data
```

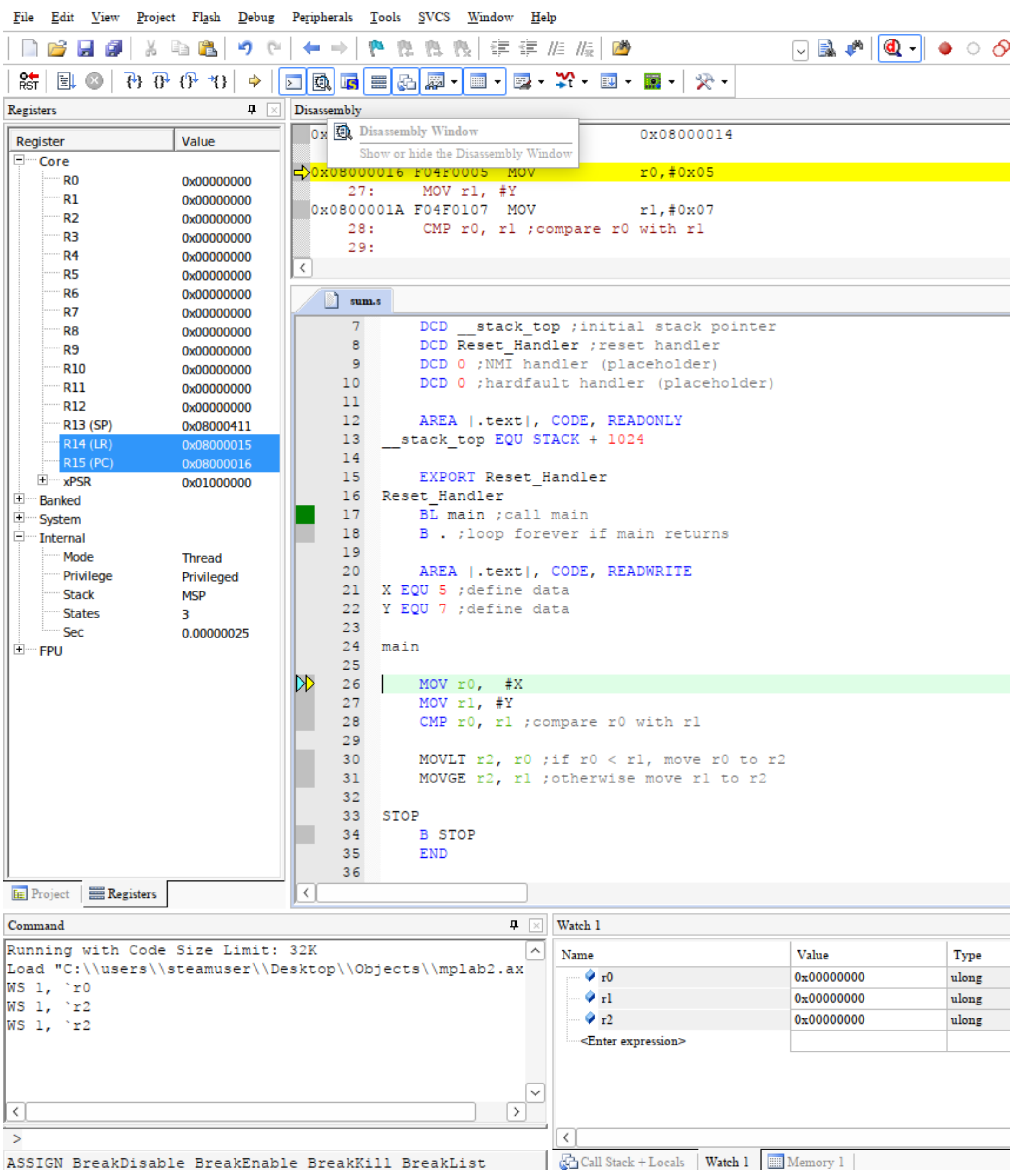
```
    DCD __stack_top ;initial stack pointer
    DCD Reset_Handler ;reset handler
    DCD 0 ;NMI handler (placeholder)
    DCD 0 ;hardfault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main ;call main
    B . ;loop forever if main returns

    AREA |.text|, CODE, READWRITE
X EQU 5 ;define data
Y EQU 7 ;define data

main

    MOV r0,  #X
    MOV r1, #Y
    CMP r0, r1 ;compare r0 with r1

    MOVLT r2, r0 ;if r0 < r1, move r0 to r2
    MOVGE r2, r1 ;otherwise move r1 to r2

STOP
    B STOP
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r2
WS 1, `r2
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000000	ulong
r1	0x00000000	ulong
r2	0x00000000	ulong
<Enter expression>		

Figure 26: After build and debugging

1.3.2 Screenshot that shows the situation after the code has been executed.


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
Disassembly Window
Show or hide the Disassembly Window
sum.s
Project
Registers
Command
Watch 1
Call Stack + Locals
Memory 1

```
Register        Value
Core
  R0            0x00000000
  R1            0x00000000
  R2            0x00000000
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x08000411
  R14 (LR)      0x08000015
  R15 (PC)      0x08000016
  xPSR          0x01000000
Banked
System
Internal
  Mode          Thread
  Privilege     Privileged
  Stack         MSP
  States        3
  Sec           0.00000025
FPU
```

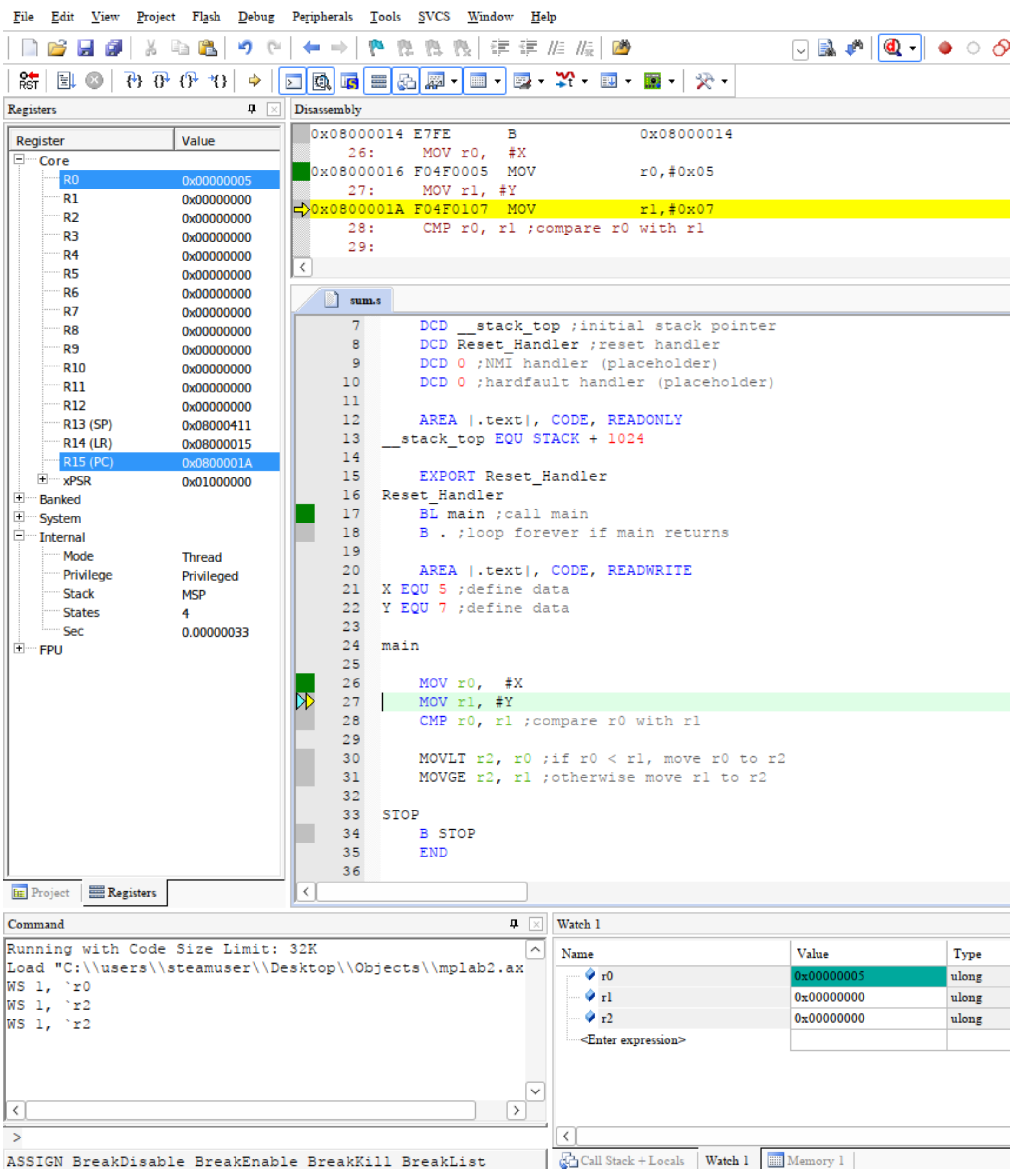
```
0x                                      0x08000014
0x08000016 F04F0005  MOV                r0,#0x05
    27:      MOV r1, #Y
0x0800001A F04F0107  MOV                r1,#0x07
    28:      CMP r0, r1 ;compare r0 with r1
    29:
```

```
    DCD __stack_top ;initial stack pointer
    DCD Reset_Handler ;reset handler
    DCD 0 ;NMI handler (placeholder)
    DCD 0 ;hardfault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main ;call main
    B . ;loop forever if main returns

    AREA |.text|, CODE, READWRITE
X EQU 5 ;define data
Y EQU 7 ;define data

main

    MOV r0,   #X
    MOV r1, #Y
    CMP r0, r1 ;compare r0 with r1

    MOVLT r2, r0 ;if r0 < r1, move r0 to r2
    MOVGE r2, r1 ;otherwise move r1 to r2

STOP
    B STOP
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r2
WS 1, `r2
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000000	ulong
r1	0x00000000	ulong
r2	0x00000000	ulong
<Enter expression>		

Figure 27: Call of main



```
Register        Value
Core
  R0            0x00000005
  R1            0x00000000
  R2            0x00000000
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x08000411
  R14 (LR)      0x08000015
  R15 (PC)      0x0800001A
  xPSR          0x01000000
Banked
System
Internal
  Mode          Thread
  Privilege     Privileged
  Stack         MSP
  States        4
  Sec           0.00000033
FPU
```

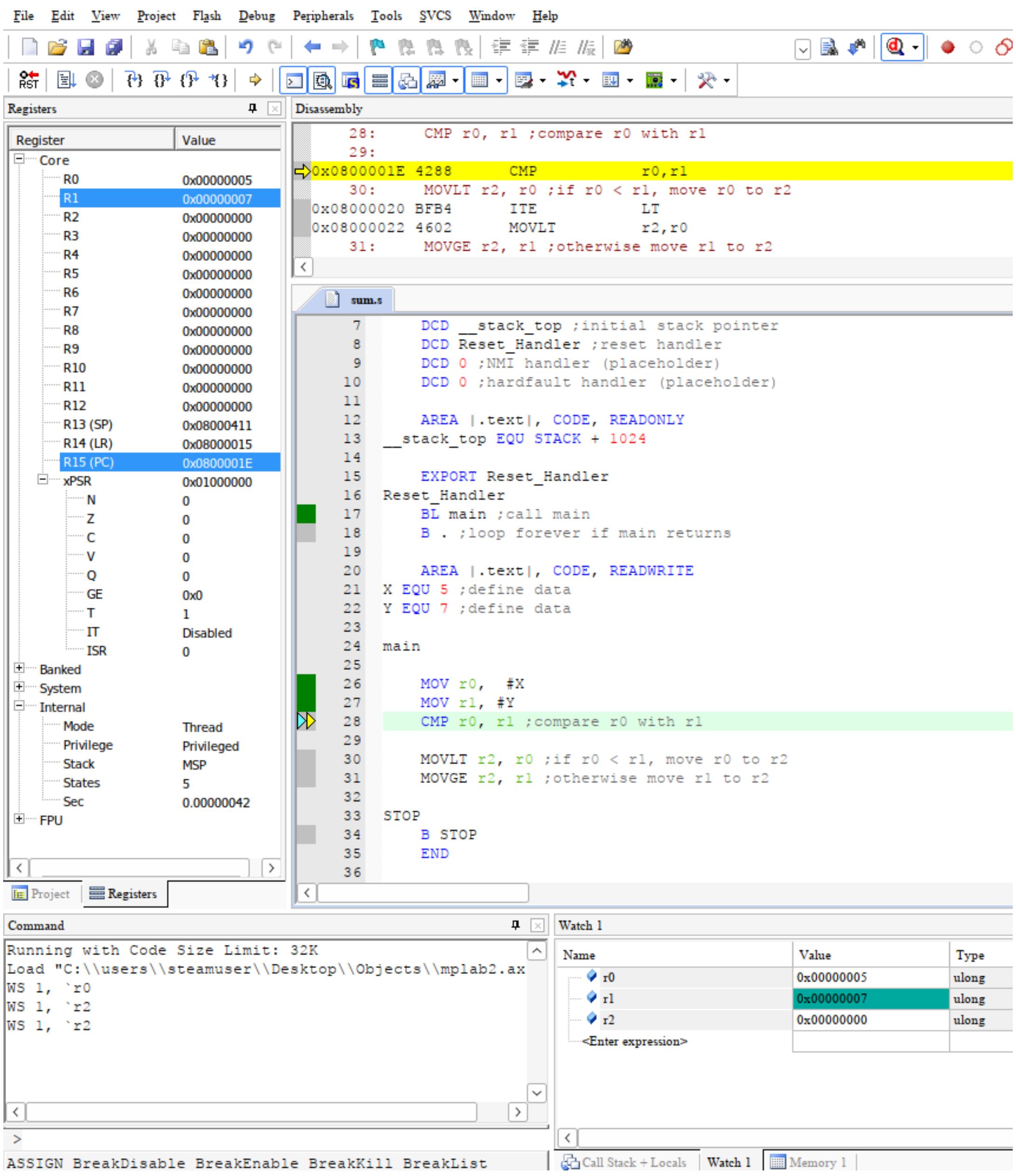
```
0x08000014 E7FE      B              0x08000014
    26:     MOV r0,  #X
0x08000016 F04F0005  MOV            r0,#0x05
    27:     MOV r1, #Y
0x0800001A F04F0107  MOV            r1,#0x07
    28:     CMP r0, r1 ;compare r0 with r1
    29:
```

```
    DCD __stack_top ;initial stack pointer
    DCD Reset_Handler ;reset handler
    DCD 0 ;NMI handler (placeholder)
    DCD 0 ;hardfault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main ;call main
    B . ;loop forever if main returns

    AREA |.text|, CODE, READWRITE
X EQU 5 ;define data
Y EQU 7 ;define data

main

    MOV r0,   #X
    MOV r1,  #Y
    CMP r0, r1 ;compare r0 with r1

    MOVLT r2, r0 ;if r0 < r1, move r0 to r2
    MOVGE r2, r1 ;otherwise move r1 to r2

STOP
    B STOP
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r2
WS 1, `r2

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000005	ulong
r1	0x00000000	ulong
r2	0x00000000	ulong
<Enter expression>		

Figure 28: Move the value of X in r0



Register	Value
Core	
R0	0x00000005
R1	0x00000007
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000411
R14 (LR)	0x08000015
R15 (PC)	0x0800001E
xPSR	0x01000000
N	0
Z	0
C	0
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	5
Sec	0.00000042
FPU	

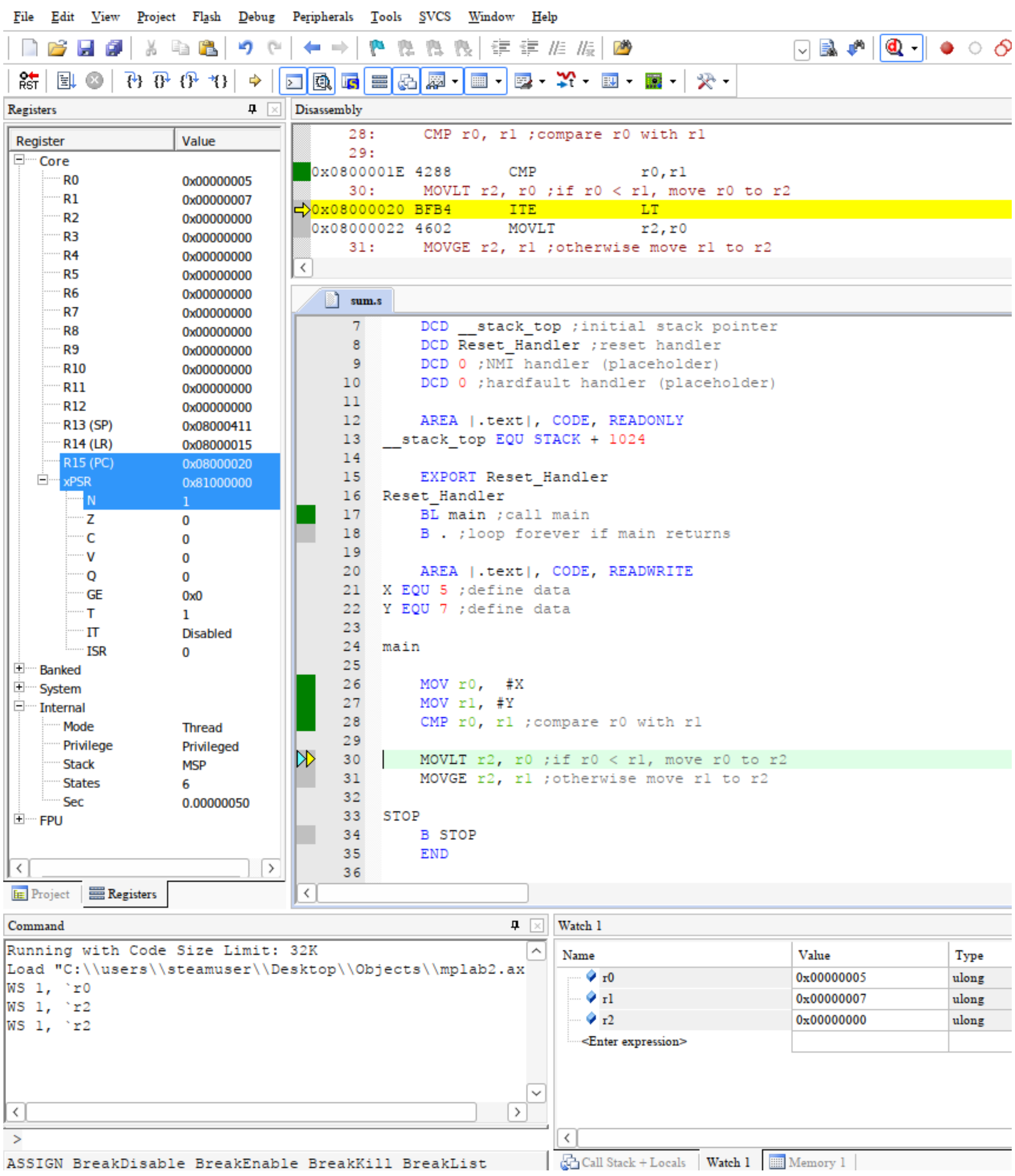
```
    28:     CMP r0, r1 ;compare r0 with r1
    29:
0x0800001E 4288       CMP          r0,r1
    30:     MOVLT r2, r0 ;if r0 < r1, move r0 to r2
0x08000020 BFB4       ITE          LT
0x08000022 4602       MOVLT        r2,r0
    31:     MOVGE r2, r1 ;otherwise move r1 to r2
```

```
    DCD __stack_top ;initial stack pointer
    DCD Reset_Handler ;reset handler
    DCD 0 ;NMI handler (placeholder)
    DCD 0 ;hardfault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main ;call main
    B . ;loop forever if main returns

    AREA |.text|, CODE, READWRITE
X EQU 5 ;define data
Y EQU 7 ;define data

main

    MOV r0,  #X
    MOV r1,  #Y
    CMP r0, r1 ;compare r0 with r1

    MOVLT r2, r0 ;if r0 < r1, move r0 to r2
    MOVGE r2, r1 ;otherwise move r1 to r2

STOP
    B STOP
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r2
WS 1, `r2
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000005	ulong
r1	0x00000007	ulong
r2	0x00000000	ulong
<Enter expression>		

Figure 29: Move the value of Y in r1



```
Register        Value
Core
  R0            0x00000005
  R1            0x00000007
  R2            0x00000000
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x08000411
  R14 (LR)      0x08000015
  R15 (PC)      0x08000020
  xPSR          0x81000000
    N           1
    Z           0
    C           0
    V           0
    Q           0
    GE          0x0
    T           1
    IT          Disabled
    ISR         0
Banked
System
Internal
  Mode          Thread
  Privilege     Privileged
  Stack         MSP
  States        6
  Sec           0.00000050
FPU
```

```
    28:     CMP r0, r1 ;compare r0 with r1
    29:
0x0800001E 4288      CMP           r0,r1
    30:     MOVLT r2, r0 ;if r0 < r1, move r0 to r2
0x08000020 BFB4      ITE           LT
0x08000022 4602      MOVLT         r2,r0
    31:     MOVGE r2, r1 ;otherwise move r1 to r2
```

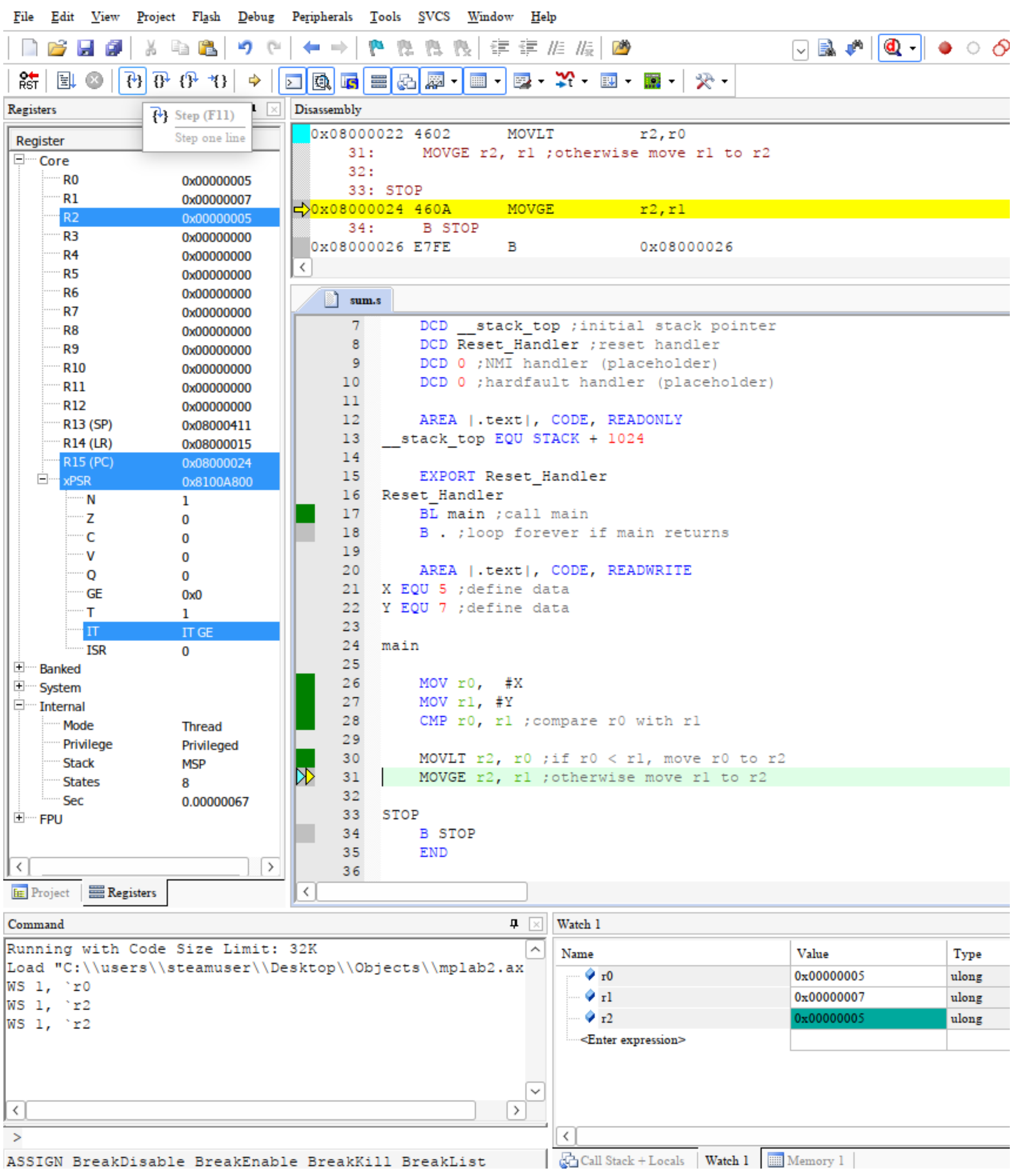
```
    DCD __stack_top ;initial stack pointer
    DCD Reset_Handler ;reset handler
    DCD 0 ;NMI handler (placeholder)
    DCD 0 ;hardfault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main ;call main
    B . ;loop forever if main returns

    AREA |.text|, CODE, READWRITE
X EQU 5 ;define data
Y EQU 7 ;define data

main

    MOV r0,  #X
    MOV r1, #Y
    CMP r0, r1 ;compare r0 with r1

    MOVLT r2, r0 ;if r0 < r1, move r0 to r2
    MOVGE r2, r1 ;otherwise move r1 to r2

STOP
    B STOP
    END
```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r2
WS 1, `r2

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000005	ulong
r1	0x00000007	ulong
r2	0x00000000	ulong
<Enter expression>		

Figure 30: Comparing the value of r0 with r1

Here in xPSR register field, N field is set to 1 which indicates negative.



Register	
Core	
R0	0x00000005
R1	0x00000007
R2	0x00000005
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000411
R14 (LR)	0x08000015
R15 (PC)	0x08000024
xPSR	0x8100A800
N	1
Z	0
C	0
V	0
Q	0
GE	0x0
T	1
IT	IT GE
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	8
Sec	0.00000067
FPU	

```
0x08000022 4602      MOVLT      r2,r0
    31:      MOVGE r2, r1 ;otherwise move r1 to r2
    32:
    33: STOP
0x08000024 460A      MOVGE      r2,r1
    34:      B STOP
0x08000026 E7FE      B          0x08000026
```

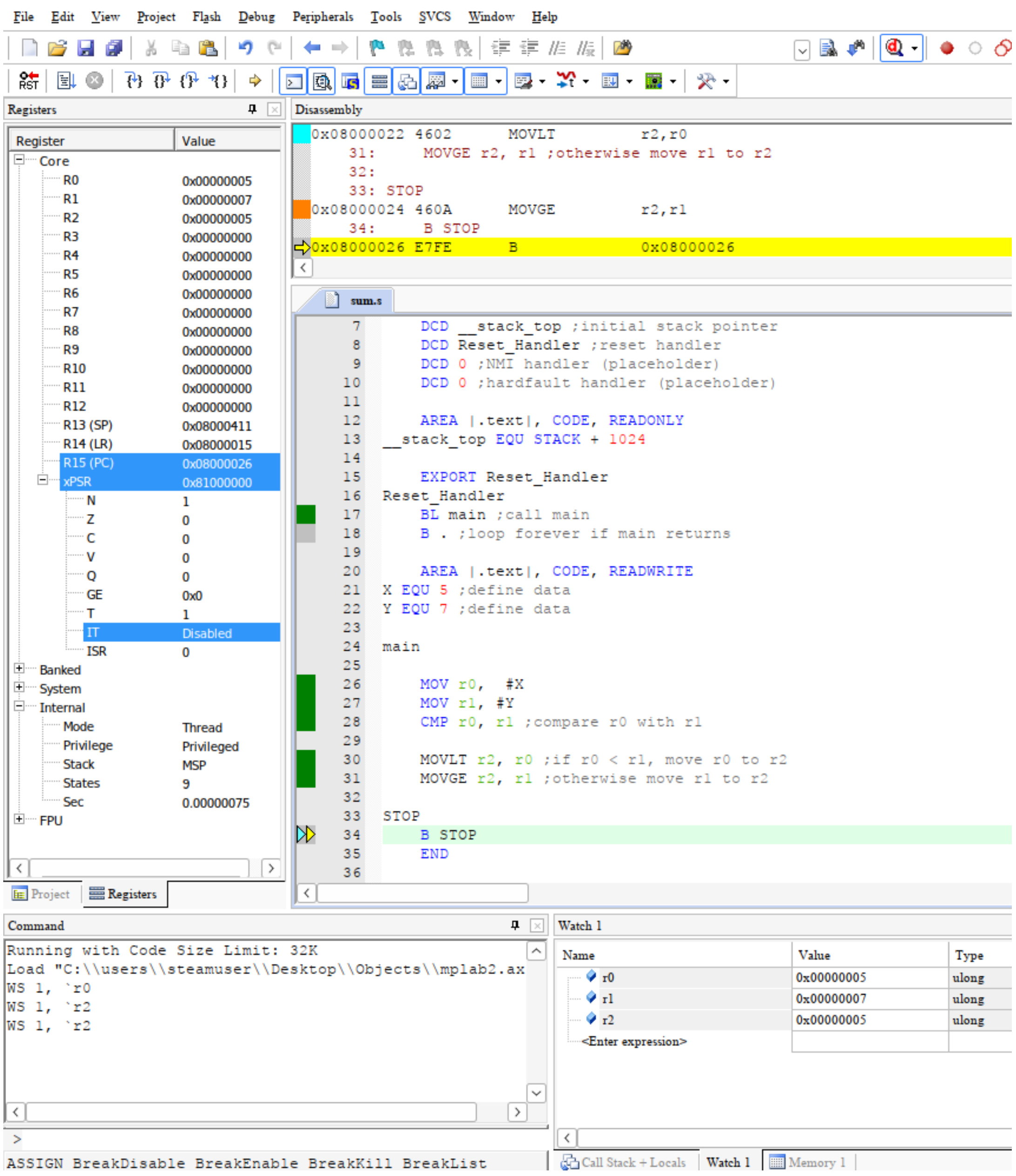
```
    DCD __stack_top ;initial stack pointer
    DCD Reset_Handler ;reset handler
    DCD 0 ;NMI handler (placeholder)
    DCD 0 ;hardfault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main ;call main
    B . ;loop forever if main returns

    AREA |.text|, CODE, READWRITE
X EQU 5 ;define data
Y EQU 7 ;define data

main

    MOV r0,  #X
    MOV r1, #Y
    CMP r0, r1 ;compare r0 with r1

    MOVLT r2, r0 ;if r0 < r1, move r0 to r2
    MOVGE r2, r1 ;otherwise move r1 to r2

STOP
    B STOP
    END

```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r2
WS 1, `r2
```

```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000005	ulong
r1	0x00000007	ulong
r2	0x00000005	ulong
<Enter expression>		

Figure 31: Moved r0 value in r2. IT (if-then) is set to GE



Register	Value
Core	
R0	0x00000005
R1	0x00000007
R2	0x00000005
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000411
R14 (LR)	0x08000015
R15 (PC)	0x08000026
xPSR	0x81000000
N	1
Z	0
C	0
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	9
Sec	0.00000075
FPU	

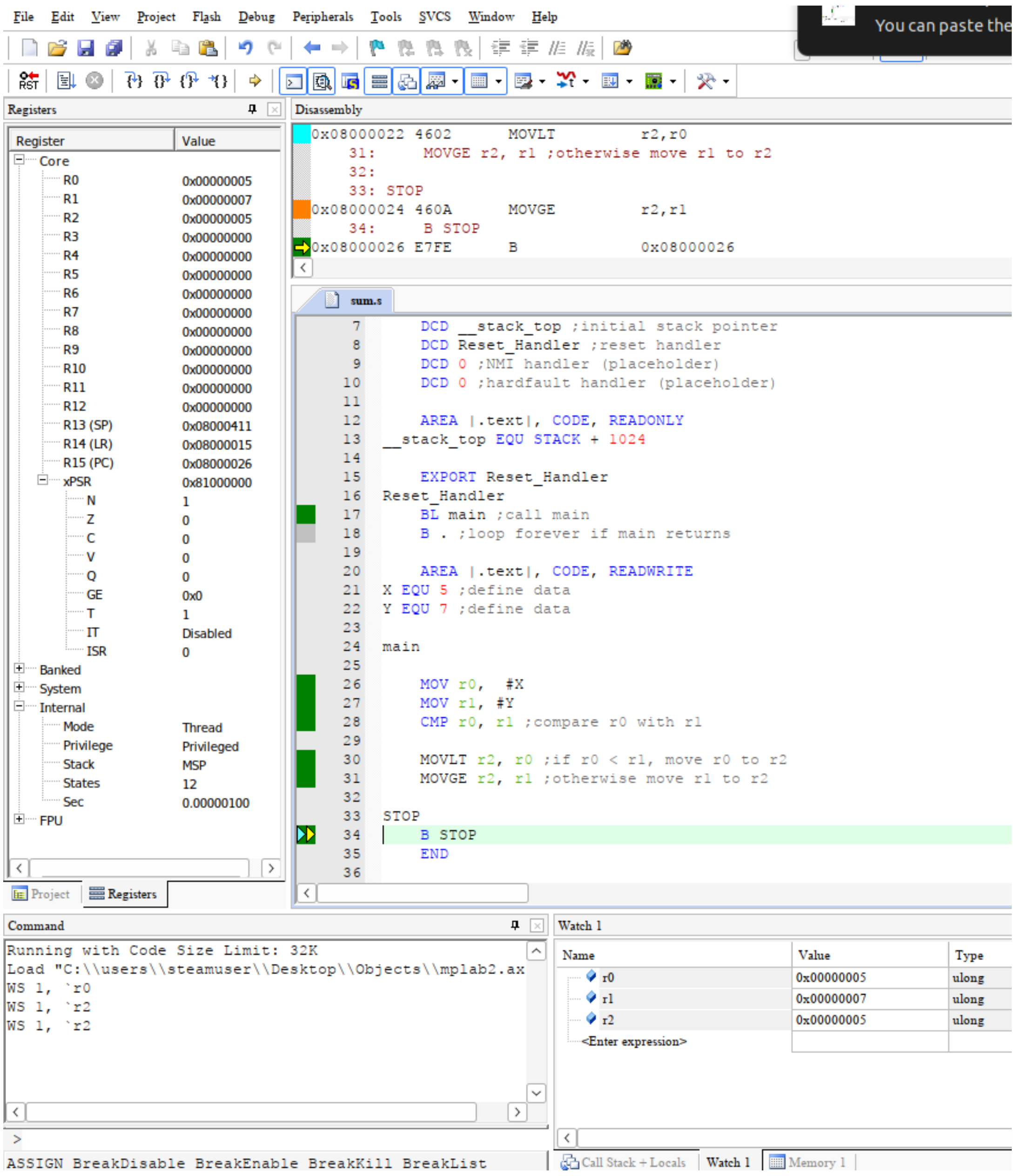
```
0x08000022 4602      MOVLT        r2,r0
    31:      MOVGE r2, r1 ;otherwise move r1 to r2
    32:
    33: STOP
0x08000024 460A      MOVGE        r2,r1
    34:      B STOP
0x08000026 E7FE      B            0x08000026
```

```
    DCD __stack_top ;initial stack pointer
    DCD Reset_Handler ;reset handler
    DCD 0 ;NMI handler (placeholder)
    DCD 0 ;hardfault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main ;call main
    B . ;loop forever if main returns

    AREA |.text|, CODE, READWRITE
X EQU 5 ;define data
Y EQU 7 ;define data

main

    MOV r0,  #X
    MOV r1, #Y
    CMP r0, r1 ;compare r0 with r1

    MOVLT r2, r0 ;if r0 < r1, move r0 to r2
    MOVGE r2, r1 ;otherwise move r1 to r2

STOP
    B STOP
    END

```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r2
WS 1, `r2

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000005	ulong
r1	0x00000007	ulong
r2	0x00000005	ulong
<Enter expression>		

Figure 32: MOVGE is not transfering data as condition is not met. IT is set to disabled



Register	Value
Core	
R0	0x00000005
R1	0x00000007
R2	0x00000005
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x08000411
R14 (LR)	0x08000015
R15 (PC)	0x08000026
xPSR	0x81000000
N	1
Z	0
C	0
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	12
Sec	0.00000100
FPU	

```
0x08000022 4602      MOVLT        r2,r0
    31:      MOVGE r2, r1 ;otherwise move r1 to r2
    32:
    33: STOP
0x08000024 460A      MOVGE        r2,r1
    34:      B STOP
0x08000026 E7FE      B            0x08000026
```

```
    DCD __stack_top ;initial stack pointer
    DCD Reset_Handler ;reset handler
    DCD 0 ;NMI handler (placeholder)
    DCD 0 ;hardfault handler (placeholder)

    AREA |.text|, CODE, READONLY
__stack_top EQU STACK + 1024

    EXPORT Reset_Handler
Reset_Handler
    BL main ;call main
    B . ;loop forever if main returns

    AREA |.text|, CODE, READWRITE
X EQU 5 ;define data
Y EQU 7 ;define data

main

    MOV r0,  #X
    MOV r1,  #Y
    CMP r0, r1 ;compare r0 with r1

    MOVLT r2, r0 ;if r0 < r1, move r0 to r2
    MOVGE r2, r1 ;otherwise move r1 to r2

STOP
    B STOP
    END

```

```
Running with Code Size Limit: 32K
Load "C:\\users\\steamuser\\Desktop\\Objects\\mplab2.ax
WS 1, `r0
WS 1, `r2
WS 1, `r2

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Name	Value	Type
r0	0x00000005	ulong
r1	0x00000007	ulong
r2	0x00000005	ulong
<Enter expression>		

Figure 33: Stop state