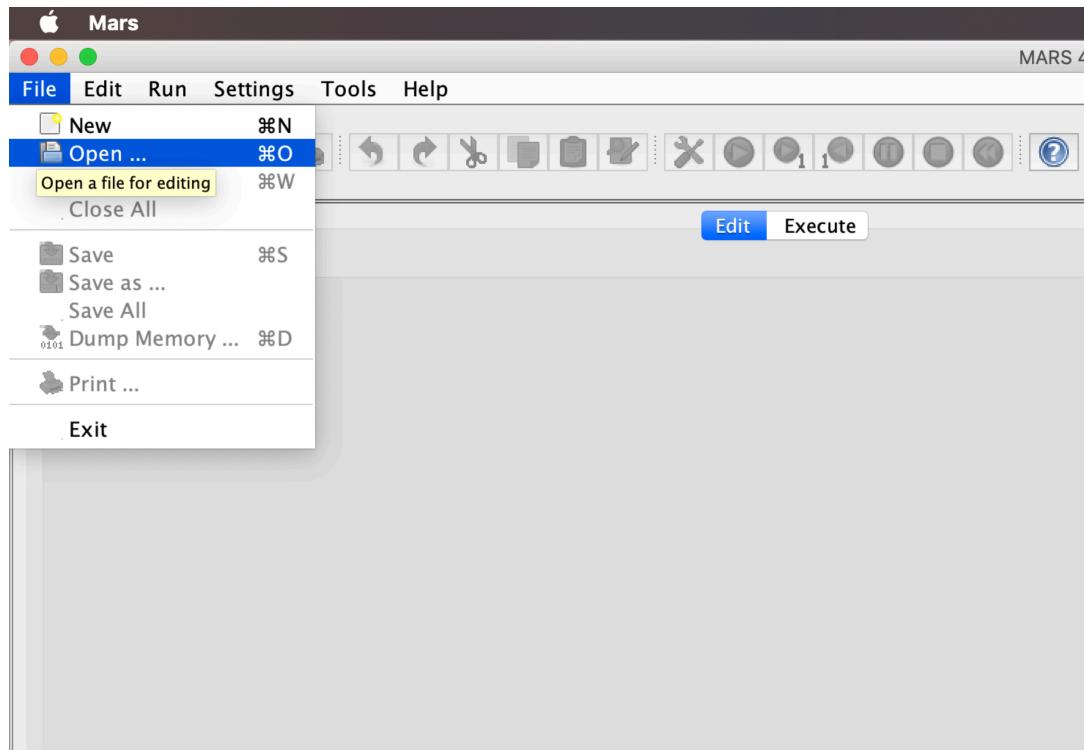


Assignment Directions – Mars Version

Please make sure to read the general assignment instruction. This document only explains the steps for setting a breakpoint in mars.

- Load the code.s file you into Mars. This can be done via the "Open" button from the File menu.



You will get the following screen:

The screenshot shows the Mars 4.5 assembly debugger interface. The assembly code in the editor window is:

```

1 | .data
2 |     endl: .asciz "\n"    # used for cout << endl;
3 |
4 |     .text
5 | # x --> $s1
6 | # y --> $s2
7 | # z --> $s3
8 | main:
9 |     li $s0, -1
10 |    li $s1, 0
11 |    li $s2, 10
12 |    li $s3, 5
13 |
14 |    loop: add $s3, $s3, $s0
15 |            add $s0, $s1, $s2
16 |
17 |    inc:   addi $s1, $s1, 1
18 |            blt $s1, $s2, loop
19 |
20 |    exit: move $s0, $s0      # puts w into arg0 (a0 register) for cout
21 |            addi $v0, $s0, 1    # puts 1 in v0 which denotes we are printing an int
22 |            syscall
23 |
24 |            la $a0, endl    # puts the address of the string endl into a0
25 |            addi $v0, $s0, 4    # puts 4 into v0 saying we are printing a string
26 |            syscall
27 |
28 |            move $s0, $s1      # puts x into arg0 (a0 register) for cout
29 |            addi $v0, $s0, 1    # puts 1 in v0 which denotes we are printing an int
30 |            syscall

```

The Registers window shows the following values:

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$s1	5	0
\$s2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194304
hi		0
lo		0

- From the Run menu select “Assemble”. This will compile the assembly code into machine code.

The screenshot shows the Mars 4.5 assembly debugger interface with the Run menu open. The Assemble option is selected, highlighted in blue. The assembly code in the editor window is identical to the one in the previous screenshot.

You should get the following screen:

The screenshot shows the MARS 4.5 assembly debugger interface. The assembly window displays the following code:

```

Text Segment
Bkpt Address Code Basic Source
0x00400000 0x2410ffff addiu $16,$0,-1
0x00400004 0x24110000 addiu $17,$0,0
0x00400008 0x24120000 addiu $18,$0,10
0x0040000c 0x24130005 addiu $19,$0,5
0x00400010 0x02709820 add $19,$19,$16
0x00400014 0x02328020 add $16,$17,$18
0x00400018 0x02231001 addi $17,$17,1
0x0040001c 0x0232082a slt $1,$17,$18
0x00400020 0x1420ffff bne $1,$0,-5
0x00400024 0x00102021 addi $4,$0,$16
0x00400028 0x20020001 addi $2,$0,1
0x0040002c 0x0000000c syscall

Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+C) Value (+10) Value (+14) Value (+18) Value (+1C)
0x10010000 10 0 0 0 0 0 0 0
0x10010020 0 0 0 0 0 0 0 0
0x10010040 0 0 0 0 0 0 0 0
0x10010060 0 0 0 0 0 0 0 0
0x10010080 0 0 0 0 0 0 0 0
0x100100A0 0 0 0 0 0 0 0 0
0x100100C0 0 0 0 0 0 0 0 0
0x100100E0 0 0 0 0 0 0 0 0
0x10010100 0 0 0 0 0 0 0 0

```

The Registers window shows the following register values:

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$s10	26	0
\$s11	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194364
hi		0
lo		0

The Mars Messages window shows:

```

Assemble: assembling /Users/soheil/mips-assembly/assignment/2-debugging/code.s
Assemble: operation completed successfully.

```

- Set a breakpoint at the loop label. This is at line 14 in the assembly source code file. It contains the instruction add \$s3, \$s3, \$s0. A breakpoint can be set in Mars by clicking on the Bkpt column of that line.

The screenshot shows the MARS 4.5 assembly debugger with a breakpoint set at line 14. The assembly window displays the following code:

```

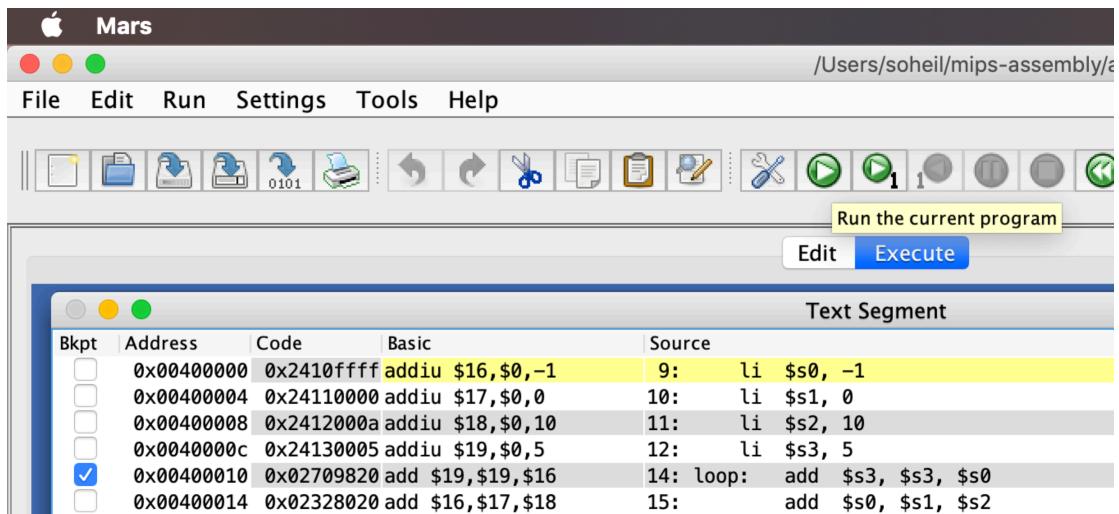
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x2410ffff addiu $16,$0,-1
0x00400004 0x24110000 addiu $17,$0,0
0x00400008 0x24120000 addiu $18,$0,10
0x0040000c 0x24130005 addiu $19,$0,5
0x00400010 0x02709820 add $19,$19,$16
0x00400014 0x02328020 add $16,$17,$18
0x00400018 0x02231001 addi $17,$17,1
0x0040001c 0x0232082a slt $1,$17,$18
0x00400020 0x1420ffff bne $1,$0,-5
0x00400024 0x00102021 addi $4,$0,$16
0x00400028 0x20020001 addi $2,$0,1
0x0040002c 0x0000000c syscall

Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+C) Value (+10) Value (+14) Value (+18) Value (+1C)
0x10010000 10 0 0 0 0 0 0 0
0x10010020 0 0 0 0 0 0 0 0
0x10010040 0 0 0 0 0 0 0 0
0x10010060 0 0 0 0 0 0 0 0
0x10010080 0 0 0 0 0 0 0 0
0x100100A0 0 0 0 0 0 0 0 0
0x100100C0 0 0 0 0 0 0 0 0
0x100100E0 0 0 0 0 0 0 0 0
0x10010100 0 0 0 0 0 0 0 0

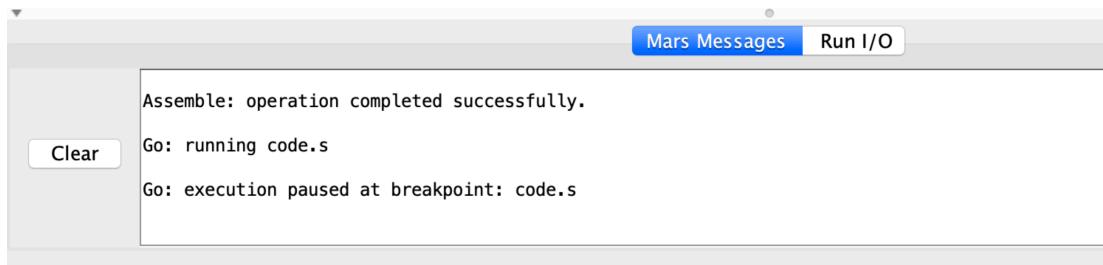
```

The Registers window shows the same register values as the previous screenshot.

- Click the play button in the top toolbar to run the MIPS program.



- The execution should pause saying "Execution has paused at breakpoint".
Figure below



- Looking to your right on the GUI, you will see a list of registers. Find the registers being used in the program. These are \$s0 to \$s3. Here you can view the values. To make things easier, you can see the values in decimal by unchecking the option "values displayed in hexadecimal" from the settings menu. Before continuing execution, you will want to write down the values of the four registers. Make sure to use the format described earlier in this document (QTSPIM Version).

\$s0	16	-1
\$s1	17	0
\$s2	18	10
\$s3	19	5

- Repeat this step. Press the play button and write the values for every time the program execution stops on the loop label. Repeat this until the program finishes.

When you have completed the assignment please upload your properly formatted .txt file to ilearn under the Project 2 section. PLEASE MAKE SURE YOUR FILE IS A .txt FILE AND NOTHING ELSE. ANY OTHER FILE TYPE SUBMITTED WILL BE IGNORED.