

**CSC 256 - Machine Structures**  
**Project 2**

**Assigned : February 26th, 2017**  
**Due : March 3rd, 2017 @ midnight**  
**Total Points: 20 Points**

**Description** For project two, the goal is to see if you have qtSpim installed and running. It is ok if you chose a different emulator other than QtSPIM. You are still expected to complete the project as outlined below. Note another notable emulator is the following:

- MARS from MSU

You will take the given program already written in MIPS assembly and run it through your emulator. You will be asked to set breakpoints at certain spots in the code to pause its execution. While execution is paused, you will be asked to list the values of certain registers. These values will be listed in a table format as shown below and submitted to ilearn as a .txt file (not a word processing document). **NOTHING ELSE WILL BE ACCEPTED.**

Sample Table format:

iteration	\$s0	\$s1	\$s2	\$s3
0	???	???	???	???
1	???	???	???	???
2	???	???	???	???
...				

Note that each value in this table is separated by spaces, including the header row(row one). PLEASE KEEP this format. Deviating from this format will cause a grade penalty to occur. ??? will be replaced with register values for each iteration of the while loop.

## Assignment Directions

- Load the code.s file you downloaded from ilearn into QtSPIM. This can be done via the "Reinitialize and Load File" button under the File tab.
- Set a breakpoint at the loop label. This at line 14 in the assembly source code file. It contains the instruction add \$s3, \$s3, \$s0. A breakpoint can be set in QtSpim by right-clicking on the instruction or address of instruction and clicking the "Set Breakpoint" tab.

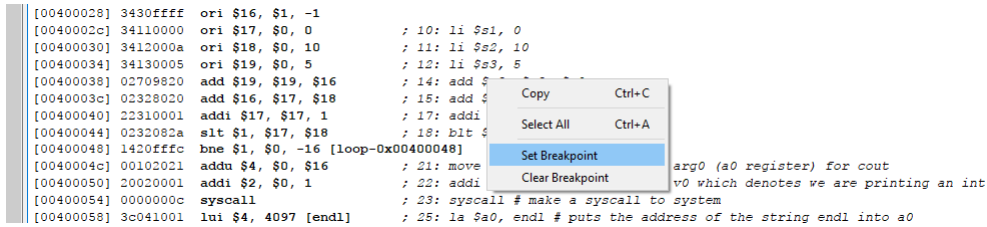


Figure 1: Setting a breakpoint a line 14

- Click the play button in the top toolbar to run the MIPS program. The execution should pause saying execution has stopped at a breakpoint. Figure below

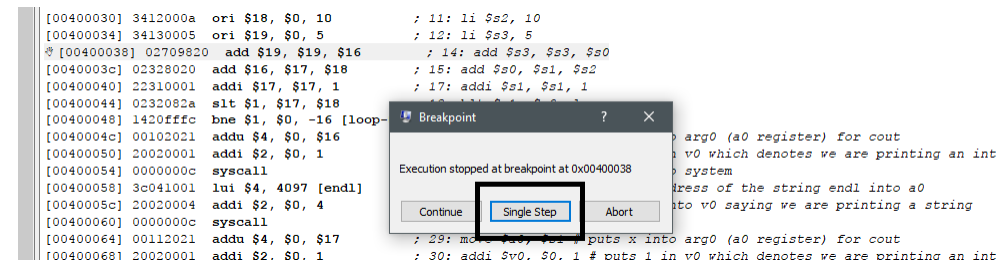


Figure 2: Execution paused when breakpoint is encountered.

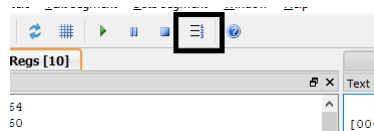
- Looking to your left on the QtSPIM GUI, you will see a list of registers. Find the registers being used in the program. These are \$s0 to \$s3. Here you can view the values. To make things easier, you can right-click on this window and set the value's base system into decimal. Before continuing execution, you will want to write down the values of the four registers. make sure to use the format described earlier in this document.

```

R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = -1
R17 [s1] = 0
R18 [s2] = 10
R19 [s3] = 4
R20 [s4] = 0
R21 [s5] = 0
---
```

Figure 3: List of registers.

- After recording the values of the registers, click on the single step button in the window. This button is shown in Figure 2. This will move your program to the next instruction. To continue your programs execution step-by-step slick the button in the top toolbar that has lines with numbers next to them. It's next to the stop button. Continue to click this button until the breakpoint dialog pops up again.



- When the window pop ups, record the values at the registers as done before. You will keep doing this until the program finishes. Please make sure to follow the table format explained earlier.

## Submission

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.**

## GIVEN MIPS Code

```
1      .data
2      endl:      .asciiz  "\n"      # used for cout << endl;
3      .text
4      # w —> $s0
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   $a0, $s0      # puts w into arg0 (a0 register) for cout
21                 addi   $v0, $0, 1    # puts 1 in v0 which denotes we are printing
22                 an     int
23                 syscall              # make a syscall to system
24
25                 la     $a0, endl     # puts the address of the string endl into a0
26                 addi   $v0, $0, 4    # puts 4 into v0 saying we are printing a
27                 an     string
28                 syscall
29
30                 move   $a0, $s1     # puts x into arg0 (a0 register) for cout
31                 addi   $v0, $0, 1    # puts 1 in v0 which denotes we are printing
32                 an     int
33                 syscall              # make a syscall to system
34
35                 la     $a0, endl     # puts the address of the string endl into a0
36                 addi   $v0, $0, 4    # puts 4 into v0 saying we are printing a
37                 an     string
38                 syscall
39
40                 move   $a0, $s2     # puts y into arg0 (a0 register) for cout
41                 addi   $v0, $0, 1    # puts 1 in v0 which denotes we are printing
42                 an     int
43                 syscall              # make a syscall to system
44
45                 la     $a0, endl     # puts the address of the string endl into a0
46                 addi   $v0, $0, 4    # puts 4 into v0 saying we are printing a
47                 an     string
48                 syscall
49
50                 move   $a0, $s3     # puts z into arg0 (a0 register) for cout
```

```

45      addi $v0, $0, 1      # puts 1 in v0 which denotes we are printing
                        an int
46      syscall            # make a syscall to system
47
48      la   $a0, endl      # puts the address of the string endl into a0
49      addi $v0, $0, 4      # puts 4 into v0 saying we are printing a
                        string
50      syscall
51
52      addi $v0,$0, 10
53      syscall

```

code.s

## C++ Equivalent

```
1 #include <iostream>
2
3 using namespace std;
4
5
6
7 int main(void)
8 {
9     int w = -1;
10    int x = 0;
11    int y = 10;
12    int z = 5;
13
14    while(x < y)
15    {
16        cout << x << endl;
17        z = z + w;
18        w = x + y;
19        x++;
20    }
21    cout << endl;
22    cout << w << endl;
23    cout << x << endl;
24    cout << y << endl;
25    cout << z << endl;
26    return 0;
27 }
```

code.cpp