CSC 256 - Machine Structures
Project 2

Total Points: 40 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. Another possible 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:

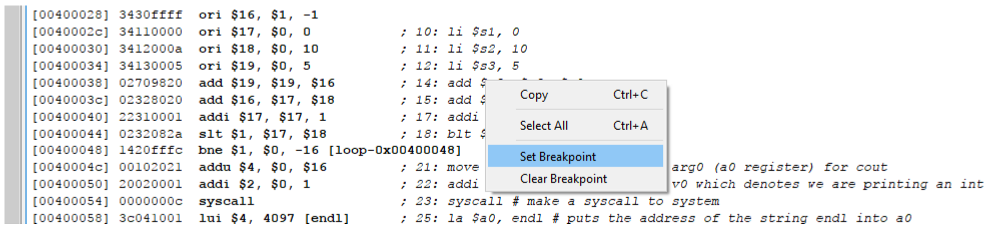
```
iteration       $s0        $s1        $s2        $s3
0               ???        ???        ???        ???
1               ???        ???        ???        ???
2               ???        ???        ???        ???
...
```

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.

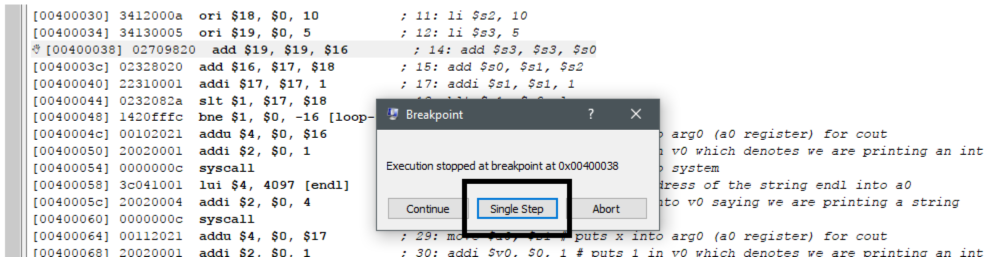
– Set a breakpoint at the loop label. This at line 14 in the assembly source code file. It contains the instruction add \$s3, \$s3, \$s0. A breakpoint can be set in QtSpim by right-clicking on the instruction or address of instruction and clicking the ”Set Breakpoint” tab.



```
[00400028] 3430ffff  ori $16, $1, -1
[0040002c] 34110000  ori $17, $0, 0           ; 10: li $s1, 0
[00400030] 3412000a  ori $18, $0, 10          ; 11: li $s2, 10
[00400034] 34130005  ori $19, $0, 5           ; 12: li $s3, 5
[00400038] 02709820  add $19, $19, $16        ; 14: add $
[0040003c] 02328020  add $16, $17, $18        ; 15: add $
[00400040] 22310001  addi $17, $17, 1         ; 17: addi
[00400044] 0232082a  slt $1, $17, $18         ; 18: blt $
[00400048] 1420fffc  bne $1, $0, -16 [loop-0x00400048]
[0040004c] 00102021  addu $4, $0, $16         ; 21: move                      arg0 (a0 register) for cout
[00400050] 20020001  addi $2, $0, 1           ; 22: addi                      v0 which denotes we are printing an int
[00400054] 0000000c  syscall                  ; 23: syscall # make a syscall to system
[00400058] 3c041001  lui $4, 4097 [endl]      ; 25: la $a0, endl # puts the address of the string endl into a0
```

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 breakpoint at some address”. Figure below

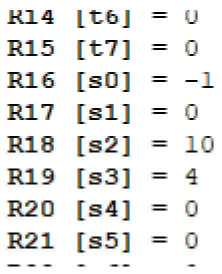


```
[00400030] 3412000a  ori $18, $0, 10          ; 11: li $s2, 10
[00400034] 34130005  ori $19, $0, 5           ; 12: li $s3, 5
[00400038] 02709820  add $19, $19, $16        ; 14: add $s3, $s3, $s0
[0040003c] 02328020  add $16, $17, $18        ; 15: add $s0, $s1, $s2
[00400040] 22310001  addi $17, $17, 1         ; 17: addi $s1, $s1, 1
[00400044] 0232082a  slt $1, $17, $18
[00400048] 1420fffc  bne $1, $0, -16 [loop-
[0040004c] 00102021  addu $4, $0, $16                                     arg0 (a0 register) for cout
[00400050] 20020001  addi $2, $0, 1                                       v0 which denotes we are printing an int
[00400054] 0000000c  syscall                                              system
[00400058] 3c041001  lui $4, 4097 [endl]                                  ress of the string endl into a0
[0040005c] 20020004  addi $2, $0, 4                                       to v0 saying we are printing a string
[00400060] 0000000c  syscall
[00400064] 00112021  addu $4, $0, $17         ; 29: mo                    puts x into arg0 (a0 register) for cout
[00400068] 20020001  addi $2, $0, 1           ; 30: addi $v0, $0, 1 # puts 1 in v0 which denotes we are printing an int
```

Execution stopped at breakpoint at 0x00400038

Figure 2: Execution paused when breakpoint is encountered.

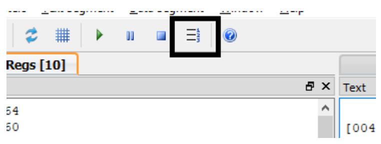
– Looking to your left on the QtSPIM GUI, you will see a list of registers. Find the registers being used in the program. These are $s0 to $s3. Here you can view the values. To make things easier, you can right-click on this window and set the value's base system into decimal. Before continuing execution, 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 click 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 in the registers as done before. You will keep doing this until the program finishes. You will do this EVERY TIME you come to the breakpoint, not every time you hit the single step button. Points will be deducted for too many or too few rows. Stop when the program finishes its execution.

– Please make sure to follow the table format explained earlier. Points will be deducted for deviating from the format outlined above.

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.**

Grading Rubric

-1 for each incorrect value recorded.

-2 for each additional row.

-2 for each missing row.

-10 for breaking file format.

-ALL POINTS for submitting anything other than a .txt file.

GIVEN MIPS Code

```
    .data
        endl:    .asciiz  "\n"   # used for cout << endl;
    .text
# w --> $s0
# x --> $s1
# y --> $s2
# z --> $s3
main:
    li  $s0, -1
    li  $s1, 0
    li  $s2, 10
    li  $s3, 5

loop:   add  $s3, $s3, $s0
        add  $s0, $s1, $s2

inc:    addi $s1, $s1, 1
        blt  $s1, $s2, loop

exit:   move $a0, $s0       # puts w into arg0 (a0 register) for cout
        addi $v0, $0, 1     # puts 1 in v0 which denotes we are printing an
           int
        syscall             # make a syscall to system

        la   $a0, endl      # puts the address of the string endl into a0
        addi $v0, $0, 4     # puts 4 into v0 saying we are printing a string
        syscall

        move $a0, $s1       # puts x into arg0 (a0 register) for cout
        addi $v0, $0, 1     # puts 1 in v0 which denotes we are printing an
           int
        syscall             # make a syscall to system

        la   $a0, endl      # puts the address of the string endl into a0
        addi $v0, $0, 4     # puts 4 into v0 saying we are printing a string
        syscall

        move $a0, $s2       # puts y into arg0 (a0 register) for cout
        addi $v0, $0, 1     # puts 1 in v0 which denotes we are printing an
           int
        syscall             # make a syscall to system

        la   $a0, endl      # puts the address of the string endl into a0
        addi $v0, $0, 4     # puts 4 into v0 saying we are printing a string
        syscall

        move $a0, $s3       # puts z into arg0 (a0 register) for cout
        addi $v0, $0, 1     # puts 1 in v0 which denotes we are printing an
           int
        syscall                 # make a syscall to system

        la   $a0, endl          # puts the address of the string endl into a0
        addi $v0, $0, 4         # puts 4 into v0 saying we are printing a string
        syscall

        addi $v0,$0, 10
        syscall
```

C++ Equivalent

```
#include <iostream>

using namespace std;



int main(void)
{
    int w = -1;
    int x = 0;
    int y = 10;
    int z = 5;

    while(x < y)
    {
        cout << x << endl;
        z = z + w;
        w = x + y;
        x++;
    }
    cout << endl;
    cout << w << endl;
    cout << x << endl;
    cout << y << endl;
    cout << z << endl;
    return 0;
}
```