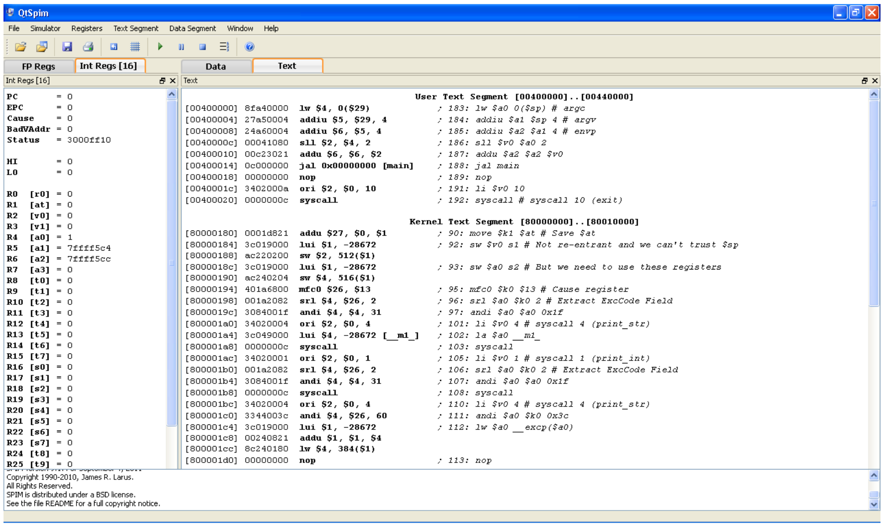
ECPE 173 – Spring 2013
QtSPIM Tutorial[1]

1. Read handout from book on QtSPIM (B-42 through B-49).
2. Start QtSpim from your desktop or Start Menu. The main window should open up as seen in Figure 1.



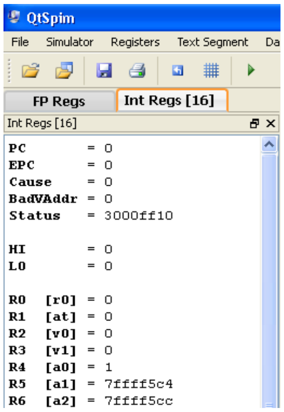
```
PC        = 0
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 3000ff10

HI        = 0
LO        = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 7ffff5c4
R6  [a2] = 7ffff5cc
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
```

```
                              User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)          ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4       ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4        ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2          ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2        ; 187: addu $a2 $a2 $v0
[00400014] 0c000000  jal 0x00000000 [main]  ; 188: jal main
[00400018] 00000000  nop                    ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10         ; 191: li $v0 10
[00400020] 0000000c  syscall                ; 192: syscall # syscall 10 (exit)

                             Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1       ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672         ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672         ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13          ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2         ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31        ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4          ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_] ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1          ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2         ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31        ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4          ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60       ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672         ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
[800001cc] 8c240180  lw $4, 384($1)
[800001d0] 00000000  nop                    ; 113: nop
```

```
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
```

Figure 1: Initial QtSPIM Window

3. There are three primary sections contained within this window. The Register panel (Figure 2) shows the contents of all the MIPS registers. There are two tabs in this panel: one for the floating point registers and one for the integer registers. The integer registers include the general purpose registers, the Program Counter, etc.

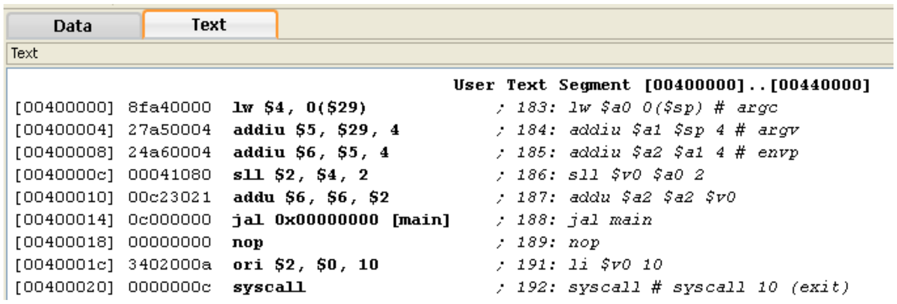


```
PC        = 0
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 3000ff10

HI        = 0
LO        = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 7ffff5c4
R6  [a2] = 7ffff5cc
```

Figure 2: Register Panel

[1] Hemmelman, B. "QT SPIM Tutorial." Accessed online at http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CCsQFjAB&url=http%3A%2F%2Fsdmines.sdsmt.edu%2Fupload%2Fdirectory%2Fmaterials%2F25478_20110916144307.doc&ei=lf8UT5iDFeWfiQL02KHNDQ&usg=AFQjCNFiXbyL302-dKeePVY3VH660GyS8g on January 16, 2012.

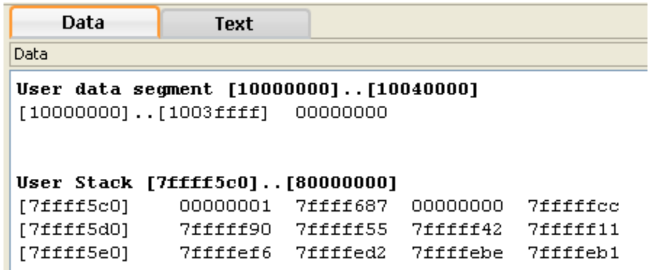
4. The Memory panel (Figure 3) has two tabs: Data and Text. The Text tab shows the contents of the Program memory space. This includes the hexadecimal memory addresses (contained in the brackets), the hexadecimal op codes, your assembly language instructions (on the right) including pseudoinstructions, and the actual assembly instructions that correspond to the op codes.



```
                                                  User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)                     ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4                  ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4                   ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2                     ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2                   ; 187: addu $a2 $a2 $v0
[00400014] 0c000000  jal 0x00000000 [main]             ; 188: jal main
[00400018] 00000000  nop                               ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10                    ; 191: li $v0 10
[00400020] 0000000c  syscall                           ; 192: syscall # syscall 10 (exit)
```

Figure 3: Memory Panel – Test Tab

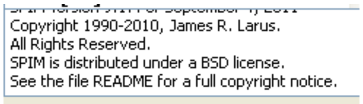
5. The Data tab (Figure 4) shows the contents of the Data memory space. This includes the variables and array data you create, along with the stack content.



```
User data segment [10000000]..[10040000]
[10000000]..[1003ffff]  00000000


User Stack [7ffff5c0]..[80000000]
[7ffff5c0]    00000001  7ffff687  00000000  7fffffcc
[7ffff5d0]    7fffff90  7fffff55  7fffff42  7fffff11
[7ffff5e0]    7ffffef6  7ffffed2  7ffffebe  7ffffeb1
```

Figure 4: Memory Panel - Data Tab

6. Finally, the Messages panel (Figure 5) is where dialogue and messages from QtSPIM are displayed to the user.



```
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
```

Figure 5: Messages Panel

7. Clear the register space by selecting Simulator → Clear Registers.

8. We will use the provided example.asm code to explore the space. Download this file from Sakai and load the assembly program by selecting File → Load File. If you receive a parser error when loading, you may need to select: File → Reinitialize and Load File.

   You can scroll down in the Text pane to see that the assembly code has been loaded into Program memory space. In this case, the first instruction is at memory location 0x00400024.

9. Select Simulator → Run Parameters. Enter 0x00400024 into the box entitled "Address or label to start running program". This simply tells QtSPIM on which line **your** code starts, since there is other initialization code that appears whenever you start QtSPIM. That code ends with a syscall at address 0x00400020.

10. You can then run a simulation of the assembly instructions, set breakpoints, or single step through your instructions. Single stepping through the instructions by pressing the single step button or F10 will show how each of the assembly instructions affects the registers. Step through the complete program, figure out what it does, and make sure you understand how the QtSPIM environment works. Once you understand, demonstrate for check-off.

11. Complete the following exercises. Submit assembly code plus screenshot of final register space via Sakai. Only one person needs to submit.

    a. Put the following bit pattern into register $s1: 0x1234ABCD

        i. Solve this using just three instructions.

        ii. Now solve this one letter at a time, using *ori* to load each letter (each nibble) into a register, and then shifting it into position.

    b. Write a program to reorder the last four bytes of register $s1 in a specific pattern and place them into $s2. If you have 0x00002143 in $s1, after running your program, you should see 0x00001234 in $s2. Try it with 0x0000AFEC.