JOHNS HOPKINS UNIVERSITY | Engineering for Professionals

# Module 6

SPIM - MIPS Assembler Workshop

## Module Six

- This week, we are going to talk about :
- SPIM - the Assembler / Translator / Simulator for the MIPS assembly language
- First, a tutorial
- Next, installation instructions
- Then, a small program to copy and execute

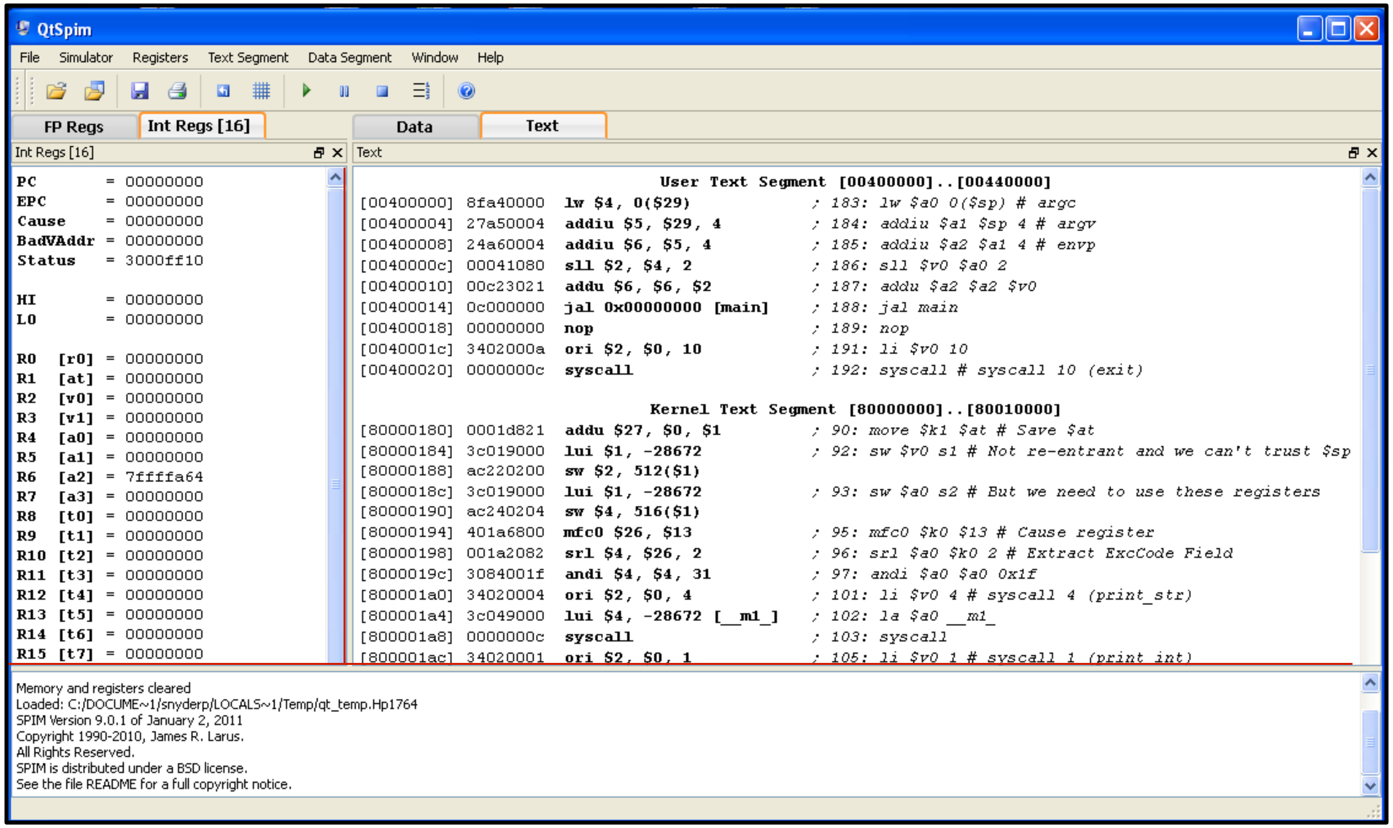
## QtSpim

- QtSpim is a self-contained simulator that will run a MIPS32 assembly program and display the processor's registers and memory.
- QtSpim reads and executes programs written in assembly language for a MIPS computer. QtSpim does not execute binary (compiled) programs.
- To simplify programming, QtSpim provides a simple debugger and small set of operating system services.
- QtSpim implements most of the MIPS32 assembler-extended instruction set. (It omits the floating point comparisons and rounding modes and the memory system page tables.)

# Getting Started with QtSpim

- When QtSpim starts up, it opens a window containing that looks like the one below.

The main window has three parts:


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text

Int Regs [16]

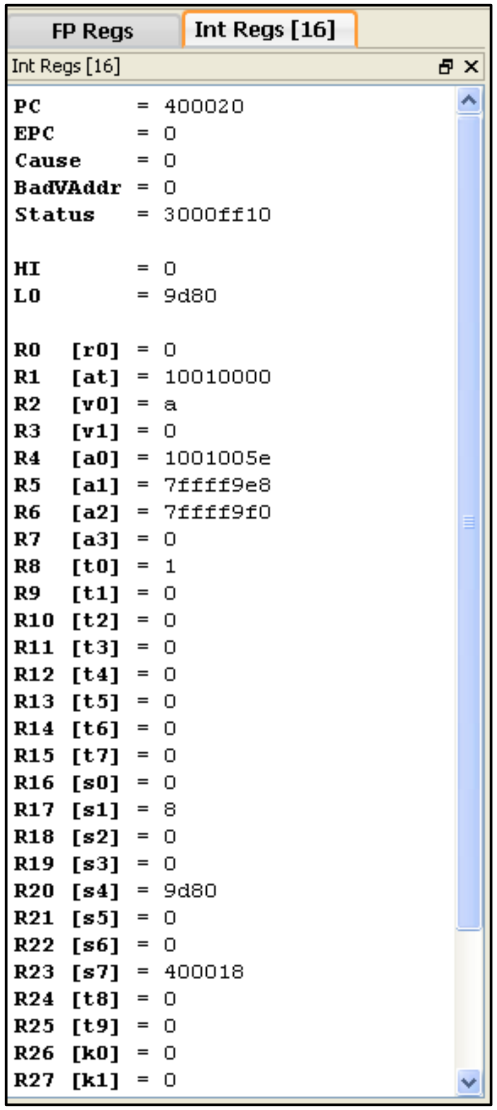
```
PC       = 00000000
EPC      = 00000000
Cause    = 00000000
BadVAddr = 00000000
Status   = 3000ff10

HI       = 00000000
LO       = 00000000

R0  [r0] = 00000000
R1  [at] = 00000000
R2  [v0] = 00000000
R3  [v1] = 00000000
R4  [a0] = 00000000
R5  [a1] = 00000000
R6  [a2] = 7ffffa64
R7  [a3] = 00000000
R8  [t0] = 00000000
R9  [t1] = 00000000
R10 [t2] = 00000000
R11 [t3] = 00000000
R12 [t4] = 00000000
R13 [t5] = 00000000
R14 [t6] = 00000000
R15 [t7] = 00000000
```

Text

```
                         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)
```

```
Memory and registers cleared
Loaded: C:/DOCUME~1/snyderp/LOCALS~1/Temp/qt_temp.Hp1764
SPIM Version 9.0.1 of January 2, 2011
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.
```

## Main Window



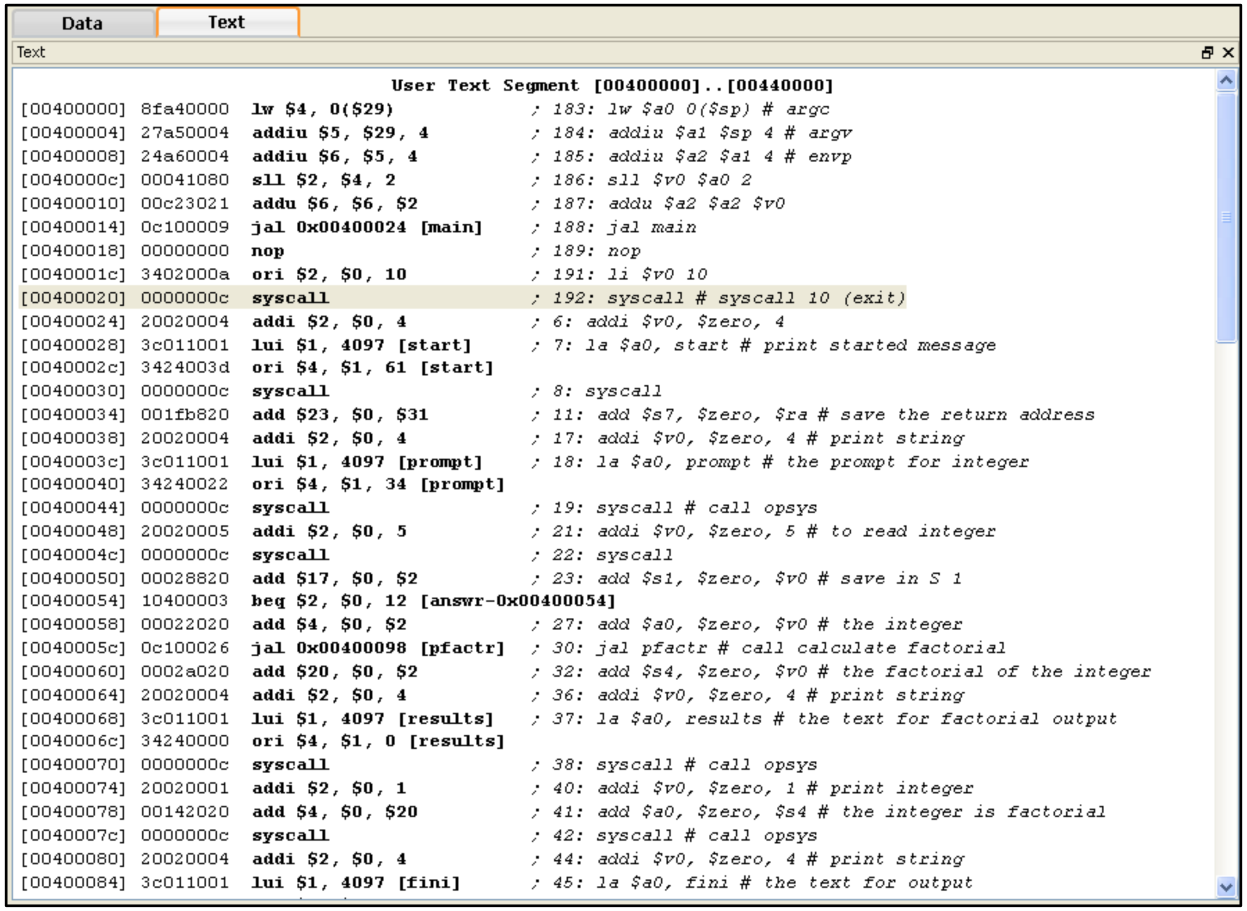
```
PC         = 400020
EPC        = 0
Cause      = 0
BadVAddr   = 0
Status     = 3000ff10

HI         = 0
LO         = 9d80

R0   [r0]  = 0
R1   [at]  = 10010000
R2   [v0]  = a
R3   [v1]  = 0
R4   [a0]  = 1001005e
R5   [a1]  = 7ffff9e8
R6   [a2]  = 7ffff9f0
R7   [a3]  = 0
R8   [t0]  = 1
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]  = 8
R18  [s2]  = 0
R19  [s3]  = 0
R20  [s4]  = 9d80
R21  [s5]  = 0
R22  [s6]  = 0
R23  [s7]  = 400018
R24  [t8]  = 0
R25  [t9]  = 0
R26  [k0]  = 0
R27  [k1]  = 0
```

- The main window has three parts:
- The narrow pane on the left can display integer or floating-point registers. Select the set of registers by clicking the tab at the top of the pane.

# Main Window

- The main window has three parts:

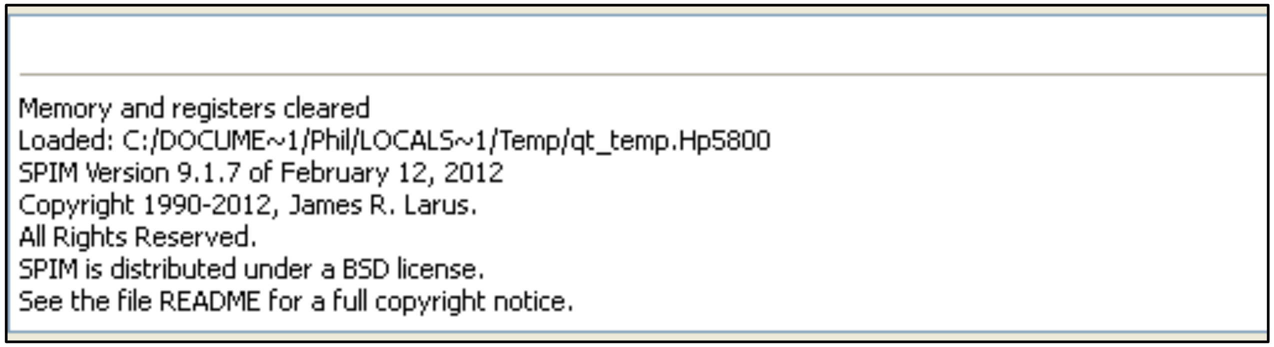
- The wide pane on the right can display the text segment, which contains instructions, and the data segments. Choose between text and data by clicking the tab at the top of the pane.



```
                        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] 0c100009  jal 0x00400024 [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)
[00400024] 20020004  addi $2, $0, 4         ; 6: addi $v0, $zero, 4
[00400028] 3c011001  lui $1, 4097 [start]   ; 7: la $a0, start # print started message
[0040002c] 3424003d  ori $4, $1, 61 [start]
[00400030] 0000000c  syscall                ; 8: syscall
[00400034] 001fb820  add $23, $0, $31       ; 11: add $s7, $zero, $ra # save the return address
[00400038] 20020004  addi $2, $0, 4         ; 17: addi $v0, $zero, 4 # print string
[0040003c] 3c011001  lui $1, 4097 [prompt]  ; 18: la $a0, prompt # the prompt for integer
[00400040] 34240022  ori $4, $1, 34 [prompt]
[00400044] 0000000c  syscall                ; 19: syscall # call opsys
[00400048] 20020005  addi $2, $0, 5         ; 21: addi $v0, $zero, 5 # to read integer
[0040004c] 0000000c  syscall                ; 22: syscall
[00400050] 00028820  add $17, $0, $2        ; 23: add $s1, $zero, $v0 # save in S 1
[00400054] 10400003  beq $2, $0, 12 [answr-0x00400054]
[00400058] 00022020  add $4, $0, $2         ; 27: add $a0, $zero, $v0 # the integer
[0040005c] 0c100026  jal 0x00400098 [pfactr] ; 30: jal pfactr # call calculate factorial
[00400060] 0002a020  add $20, $0, $2        ; 32: add $s4, $zero, $v0 # the factorial of the integer
[00400064] 20020004  addi $2, $0, 4         ; 36: addi $v0, $zero, 4 # print string
[00400068] 3c011001  lui $1, 4097 [results] ; 37: la $a0, results # the text for factorial output
[0040006c] 34240000  ori $4, $1, 0 [results]
[00400070] 0000000c  syscall                ; 38: syscall # call opsys
[00400074] 20020001  addi $2, $0, 1         ; 40: addi $v0, $zero, 1 # print integer
[00400078] 00142020  add $4, $0, $20        ; 41: add $a0, $zero, $s4 # the integer is factorial
[0040007c] 0000000c  syscall                ; 42: syscall # call opsys
[00400080] 20020004  addi $2, $0, 4         ; 44: addi $v0, $zero, 4 # print string
[00400084] 3c011001  lui $1, 4097 [fini]    ; 45: la $a0, fini # the text for output
```

## Main Window

- The main window has three parts:
- The small pane on the bottom is where QtSpim writes its messages.



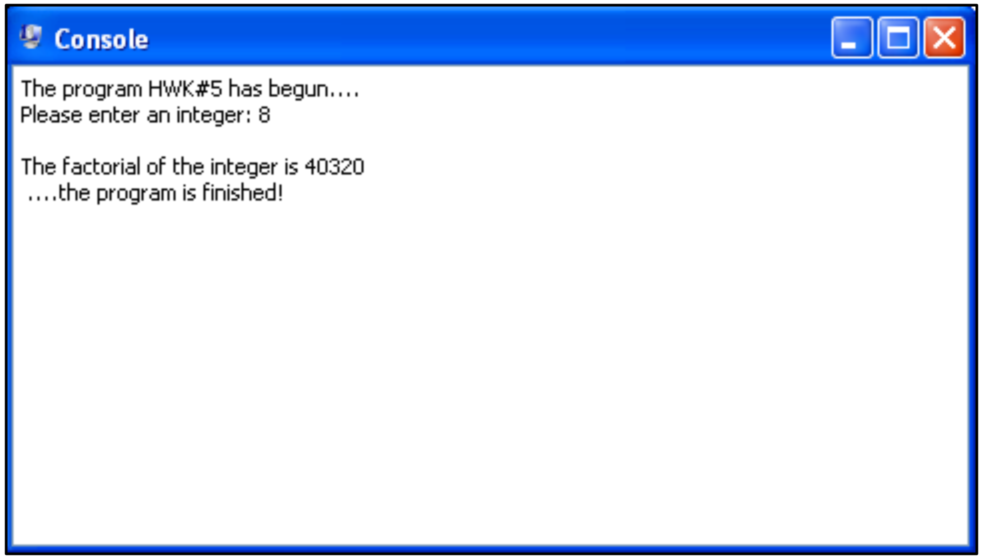
```
Memory and registers cleared
Loaded: C:/DOCUME~1/Phil/LOCALS~1/Temp/qt_temp.Hp5800
SPIM Version 9.1.7 of February 12, 2012
Copyright 1990-2012, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
```

- All of the panes are dockable, which means that you can grab a pane by its top bar and drag it out of QtSpim's main window, to put on some other part of your screen.

JOHNS HOPKINS UNIVERSITY | Engineering for Professionals

## Console

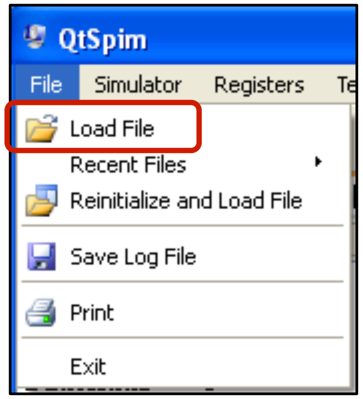
- QtSpim also opens another window called Console that displays output from your program.



```
The program HWK#5 has begun....
Please enter an integer: 8

The factorial of the integer is 40320
 ....the program is finished!
```

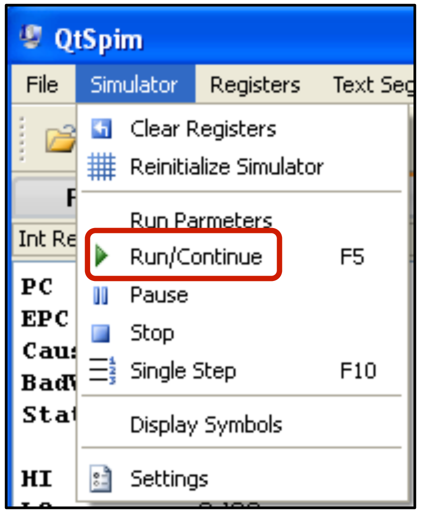
## Loading a Program

- Your program should be stored in a file. Assembly code files usually have the extension ".asm", as in file1.asm.
- To load a file, go to the File menu and select Load File.
- The screen will change as the file is loaded, to show the instructions and data in your program.
- Another very useful command on the File menu is Reinitialize and Load File.
- It first clears all changes made by a program, including deleting all of its instructions, and then reloads the last file.
- This command works well when debugging a program, as you can change your program and quickly test it in a fresh computer without closing and restarting QtSpim.



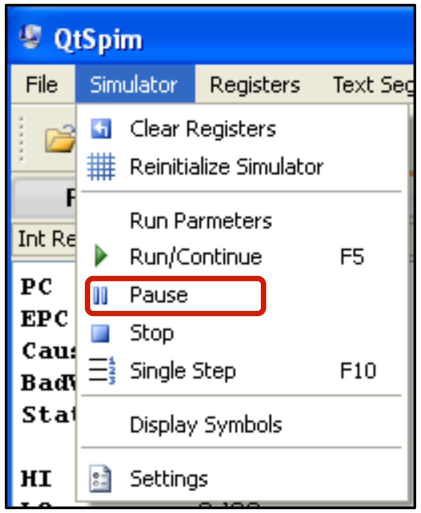
## Running a Program

- To start a program running after you have loaded it, go to the Simulator menu and click Run/Continue.
- Your program will run until it finishes or until an error occurs.
- Either way, you will see the changes that your program made to the MIPS registers and memory, and the output your program writes will appear in the Console window.
- If your program does not work correctly, there are several things you can do. The easiest is to single step between instructions, which lets you see the changes each instructions makes, one at a time. This command is also on the Simulator menu and is named Single Step.



## To STOP a program

- If you want to stop your program while it is running, go to the Simulator menu and click Pause.
- This command stops your program, let you look around, and continue execution if you want. If you do not want to continue running, click Stop instead.
- When QtSpim stops, either because of an error in your program, a breakpoint, after clicking Pause, or after single stepping, you can continue the program running by clicking on Run/Continue.
- If you click Stop, instead of Pause, then clicking Run/ Continue will restart your program from the beginning, instead of continuing from where it stopped.



## Display Options

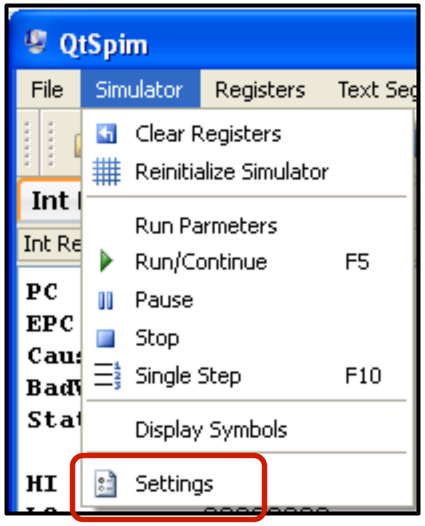
- The three other menus -- Registers, Text Segment, and Data Segment -- control QtSpim's displays.
- For example, the Register menu controls the way QtSpim displays the contents of registers, either in binary, base 8 (octal), base 10 (decimal), or base 16 (hexadecimal).
- It is often quite convenient to flip between these representations to understand your data.
- These menus also let you turn off the display of various parts of the machine, which can help reduce clutter on the screen and let you concentrate on the parts of the program or data that really matter.

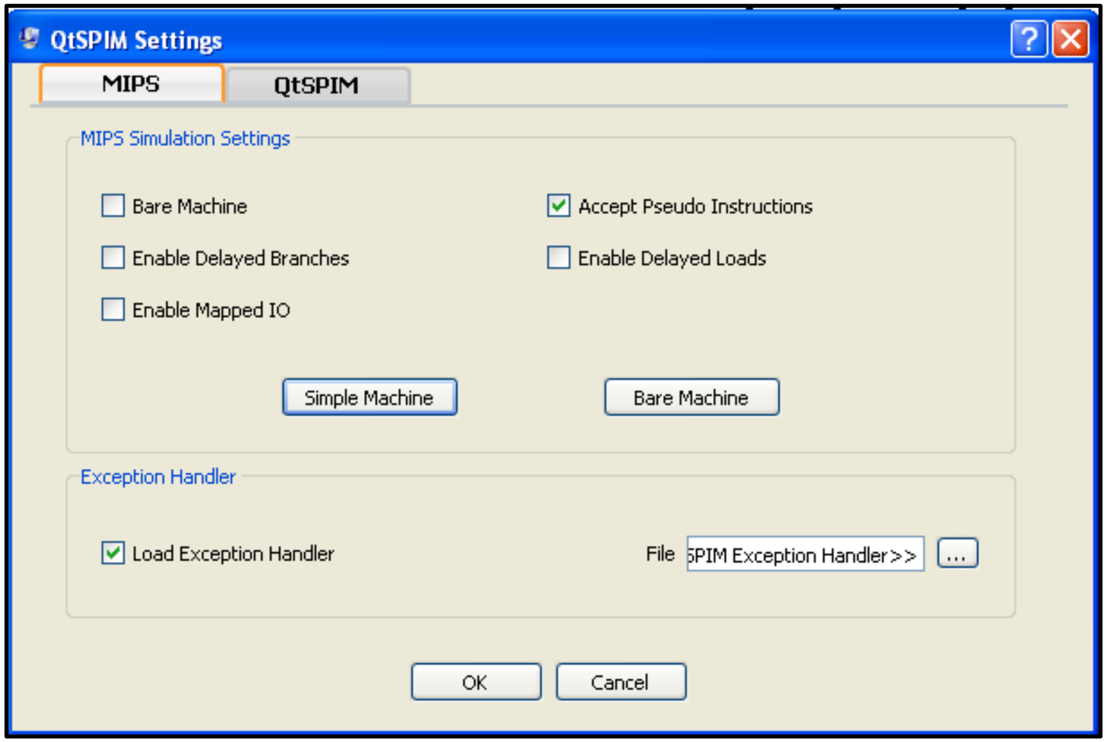
## Changing Registers and Memory

- You can change the contents of either a register or memory location by right-clicking on it and selecting Change Register Contents or Change Memory Contents, respectively.

## Settings

- The Simulator menu contains the Settings command, which brings up a dialog like this:





QtSPIM Settings

MIPS Simulation Settings

- [ ] Bare Machine
- [✓] Accept Pseudo Instructions
- [ ] Enable Delayed Branches
- [ ] Enable Delayed Loads
- [ ] Enable Mapped IO

Exception Handler

- [✓] Load Exception Handler

File: SPIM Exception Handler>>

## Settings (continued)

- It changes the way that *QtSpim* operates:

- **Bare machine** make *QtSpim* simulate a bare MIPS processor.
- **Accept pseudo instructions** enables *QtSpim* to accept assembly instructions that MIPS does not actually execute, to make programming easier.
- **Enable delayed branches** causes *QtSpim* to execute the instruction immediateyafter a branch instruction before transfering control and to calculate the new PC from the address of this next instruction.
- **Enable delayed loads** causes *QtSpim* to delay the value loaded from memory for one instruction after the load instructions.
- **Enable mapped IO** turns on memory-mapped IO.

## Summary

- SPIM Tutorial

Next: SPIM Download from Internet