CS 305/341

Tutorial on Using QtSpim[1]

QtSpim is software that will help you to simulate the execution of MIPS assembly programs. It does a context and syntax check while loading an assembly program. In addition, it adds in necessary overhead instructions as needed, and updates register and memory content as each instruction is executed. Below, is a tutorial on how to use QtSpim.

Where to Get QtSpim?

Download the source from the SourceForge.org link at: http://pages.cs.wisc.edu/~larus/spim.html (See "New versions of spim" text in red at the top of the page.) Alternatively, you can go directly to: http://sourceforge.net/projects/spimsimulator/files/
- Note that versions for Windows machines, Linux machines, and Macs are all available

SPIM in action

When you open QtSpim, A window will open as shown in Figure 1. The window is divided into different sections:

1. The *Register* tabs display the content of all registers.

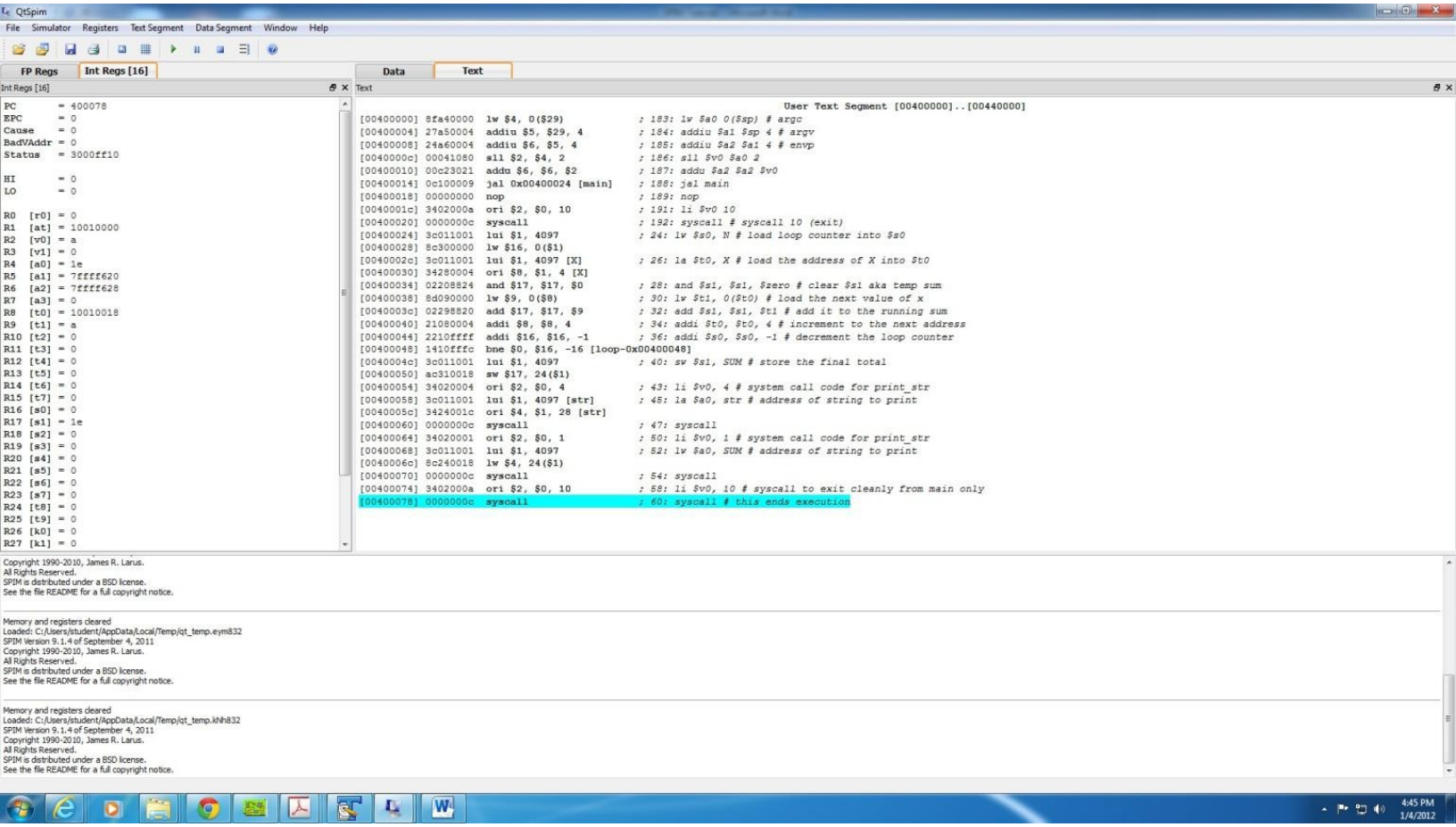
2. Buttons across the top are used to load and run a simulation

1This material was created by the Oregon State University EECS Department

3. The *Text* tab displays the MIPS instructions loaded into memory to be executed. (From left-to-right, the memory address of an instruction, the contents of the address in hex, the actual MIPS instructions – where register numbers are used, the MIPS assembly that you wrote, and any comments you made in your code are displayed.)

4. The *Data* tab displays memory addresses and their values in the data and stack segments of the memory.

5. The *Information Console* lists the actions performed by the simulator.



To run the program in QtSpim:

1. Use a text editor to create your program yyyyyy.s

2. Click on the “load” button and open yyyyyy.s

3. You can then run the program by simply pressing the “run” (play) button – all instructions will be executed, and the final contents of memory and the register file will be reflected in the QtSpim window.

Debugging

Suppose your program does not do what you expect. What can you do? QtSpim has two features that help debug your program.

The first, and perhaps the most useful, is single-stepping, which allows you to run your program an instruction at a time. The single stepping icon can be found in the toolbar. Every time you do single stepping, QtSpim will execute one instruction and update its display, so that you can see what the instruction changed in the registers or memory.

What do you do if your program runs for a long time before the bug arises? You could single-step until you get to the bug, but that can take a long time. A better alternative is to use a *breakpoint* , which tells QtSpim to stop your program immediately before it executes a particular instruction. When QtSpim is about to execute the instruction where there is a breakpoint, it asks for continue, single stepping or abort.

Single-stepping and setting breakpoints will probably help you find a bug in your program quickly. How do you fix it? Go back to the editor that you used to create your program and change it. Click on the Riinitialize simulator tab in the toolsbar and load the sourcefile again.

Generally Useful Information

When using QtSpim, you may find the following information to be useful:

You can access all of the commands via the “File” and “Simulator” menus as well.

When examining register or memory data, you can view the data in binary, hex, or decimal format. Just use the “Register” pull down menu to select.

Kernel Text and Kernel Data may not be necessary to be viewed all the times, you can unselect them by unselecting “Kernel Text” in the “Text Segment” pull down menu and unselecting “Kernel Data” in the “Data Segment” pull down menu.

You can set breakpoints in your code simply by right clicking on an instruction in the Text tab.

To view memory data, simply click on the Data tab.

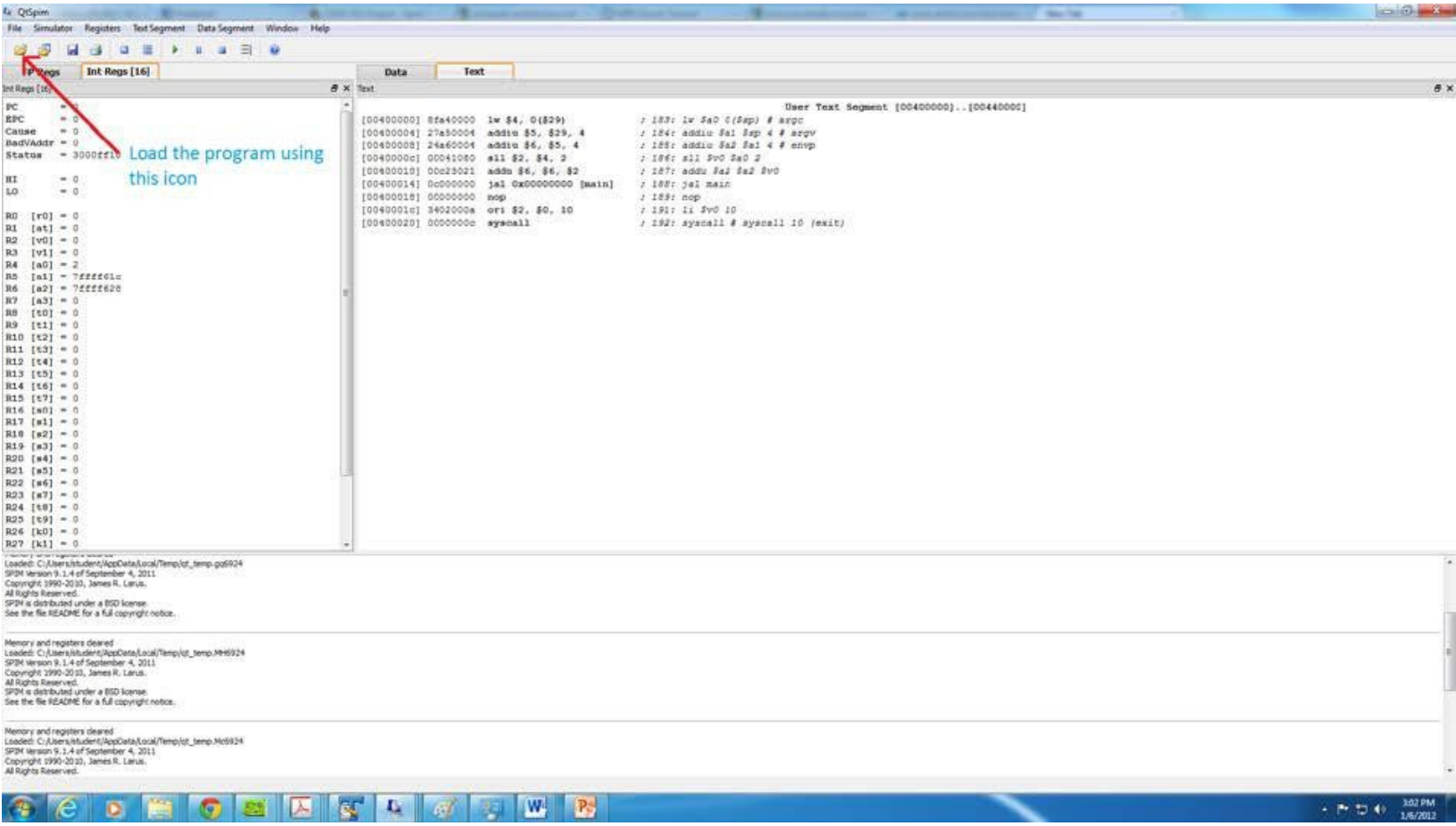
By right clicking on a register file value or memory address value, you can change its contents dynamically.

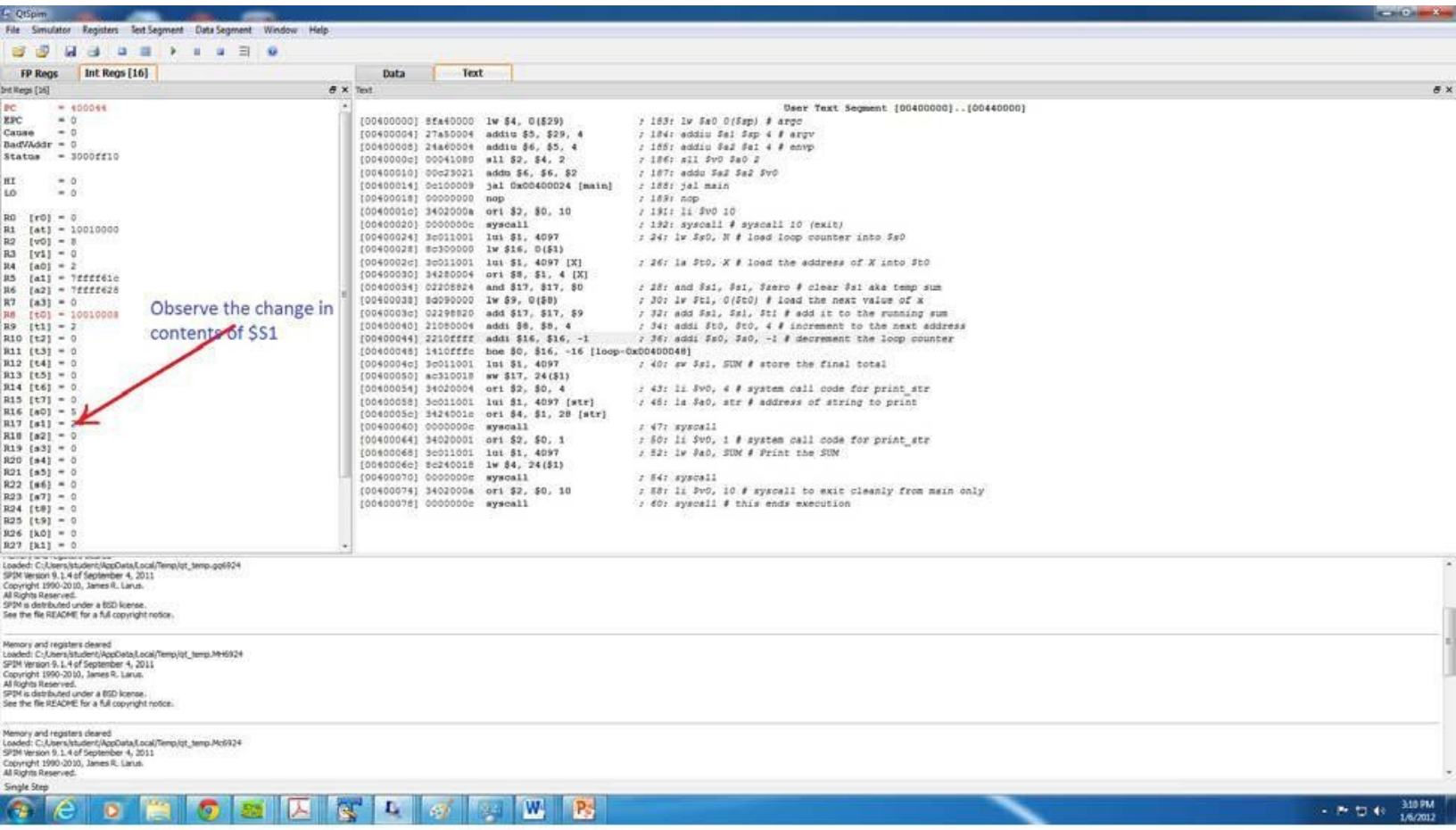
Example Program

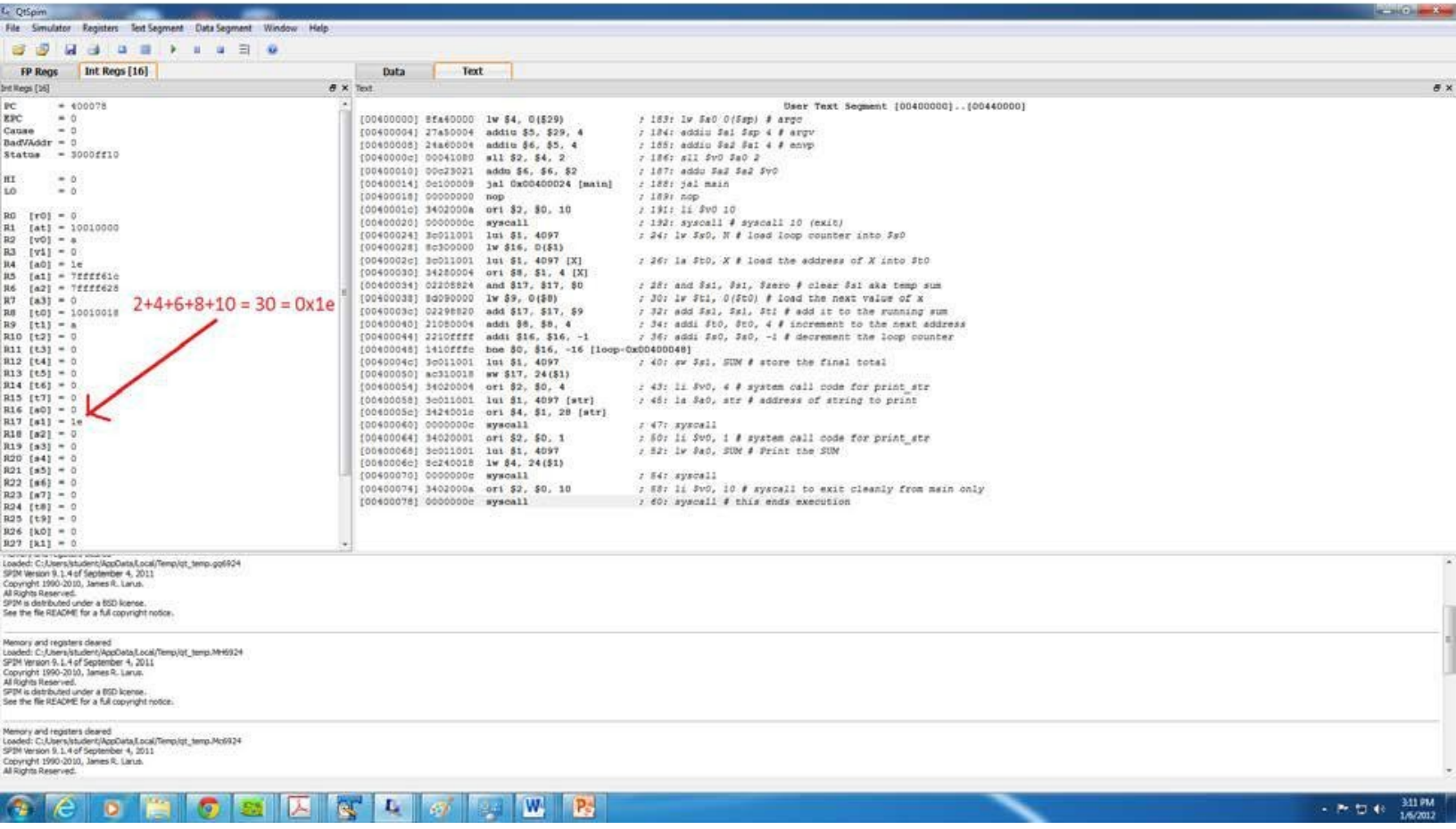
Consider an example program to find the sum of an array.Copy this into a text editor and save it as a **.s** file, say **mysum.s** and open it in QtSpim by loading the file. You can directly run it or do single stepping and observe the change in the Register file.

Steps:

1. Load the program




QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
PC = 400044
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = 8
R3 [v1] = 0
R4 [a0] = 2
R5 [a1] = 7fffff61c
R6 [a2] = 7fffff628
R7 [a3] = 0
R8 [t0] = 10010008
R9 [t1] = 2
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 5
R17 [s1] = 2
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
Observe the change in contents of $S1
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] 3c011001 lui $1, 4097 ; 24: lw $s0, N # load loop counter into $s0
[00400028] 8c300000 lw $16, 0($1)
[0040002c] 3c011001 lui $1, 4097 [X] ; 26: la $t0, X # load the address of X into $t0
[00400030] 34280004 ori $8, $1, 4 [X]
[00400034] 02208824 and $17, $17, $0 ; 28: and $s1, $s1, $zero # clear $s1 aka temp sum
[00400038] 8d090000 lw $9, 0($8) ; 30: lw $t1, 0($t0) # load the next value of x
[0040003c] 02298820 add $17, $17, $9 ; 32: add $s1, $s1, $t1 # add it to the running sum
[00400040] 21080004 addi $8, $8, 4 ; 34: addi $t0, $t0, 4 # increment to the next address
[00400044] 2210ffff addi $16, $16, -1 ; 36: addi $s0, $s0, -1 # decrement the loop counter
[00400048] 1410fffc bne $0, $16, -16 [loop-0x00400048]
[0040004c] 3c011001 lui $1, 4097 ; 40: sw $s1, SUM # store the final total
[00400050] ac310018 sw $17, 24($1)
[00400054] 34020004 ori $2, $0, 4 ; 43: li $v0, 4 # system call code for print_str
[00400058] 3c011001 lui $1, 4097 [str] ; 45: la $a0, str # address of string to print
[0040005c] 3424001c ori $4, $1, 28 [str]
[00400060] 0000000c syscall ; 47: syscall
[00400064] 34020001 ori $2, $0, 1 ; 50: li $v0, 1 # system call code for print_str
[00400068] 3c011001 lui $1, 4097 ; 52: lw $a0, SUM # Print the SUM
[0040006c] 8c240018 lw $4, 24($1)
[00400070] 0000000c syscall ; 54: syscall
[00400074] 3402000a ori $2, $0, 10 ; 58: li $v0, 10 # syscall to exit cleanly from main only
[00400078] 0000000c syscall ; 60: syscall # this ends execution
Loaded: C:/Users/student/AppData/Local/Temp/qt_temp.qq6924
SPIM Version 9.1.4 of September 4, 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.
Memory and registers cleared
Loaded: C:/Users/student/AppData/Local/Temp/qt_temp.MH6924
SPIM Version 9.1.4 of September 4, 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.
Memory and registers cleared
Loaded: C:/Users/student/AppData/Local/Temp/qt_temp.Mc6924
SPIM Version 9.1.4 of September 4, 2011
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
Single Step
3:10 PM
1/6/2012


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
PC = 400078
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = a
R3 [v1] = 0
R4 [a0] = 1e
R5 [a1] = 7fffff61c
R6 [a2] = 7fffff628
R7 [a3] = 0
R8 [t0] = 10010018
R9 [t1] = a
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 1e
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
2+4+6+8+10 = 30 = 0x1e
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] 3c011001 lui $1, 4097 ; 24: lw $s0, N # load loop counter into $s0
[00400028] 8c300000 lw $16, 0($1)
[0040002c] 3c011001 lui $1, 4097 [X] ; 26: la $t0, X # load the address of X into $t0
[00400030] 34280004 ori $8, $1, 4 [X]
[00400034] 02208824 and $17, $17, $0 ; 28: and $s1, $s1, $zero # clear $s1 aka temp sum
[00400038] 8d090000 lw $9, 0($8) ; 30: lw $t1, 0($t0) # load the next value of x
[0040003c] 02298820 add $17, $17, $9 ; 32: add $s1, $s1, $t1 # add it to the running sum
[00400040] 21080004 addi $8, $8, 4 ; 34: addi $t0, $t0, 4 # increment to the next address
[00400044] 2210ffff addi $16, $16, -1 ; 36: addi $s0, $s0, -1 # decrement the loop counter
[00400048] 1410fffc bne $0, $16, -16 [loop-0x00400048]
[0040004c] 3c011001 lui $1, 4097 ; 40: sw $s1, SUM # store the final total
[00400050] ac310018 sw $17, 24($1)
[00400054] 34020004 ori $2, $0, 4 ; 43: li $v0, 4 # system call code for print_str
[00400058] 3c011001 lui $1, 4097 [str] ; 45: la $a0, str # address of string to print
[0040005c] 3424001c ori $4, $1, 28 [str]
[00400060] 0000000c syscall ; 47: syscall
[00400064] 34020001 ori $2, $0, 1 ; 50: li $v0, 1 # system call code for print_str
[00400068] 3c011001 lui $1, 4097 ; 52: lw $a0, SUM # Print the SUM
[0040006c] 8c240018 lw $4, 24($1)
[00400070] 0000000c syscall ; 54: syscall
[00400074] 3402000a ori $2, $0, 10 ; 58: li $v0, 10 # syscall to exit cleanly from main only
[00400078] 0000000c syscall ; 60: syscall # this ends execution
Loaded: C:/Users/student/AppData/Local/Temp/qt_temp.qq6924
SPIM Version 9.1.4 of September 4, 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.
Memory and registers cleared
Loaded: C:/Users/student/AppData/Local/Temp/qt_temp.MH6924
SPIM Version 9.1.4 of September 4, 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.
Memory and registers cleared
Loaded: C:/Users/student/AppData/Local/Temp/qt_temp.Mc6924
SPIM Version 9.1.4 of September 4, 2011
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
3:11 PM
1/6/2012