計算機組織與結構 – HOMEWORK1

撰寫MIPS程式 (共題，100分，滿分100分)

請於2022/10/25前上傳至eschool作業區繳交

請安裝QtSpim (http://spimsimulator.sourceforge.net/) 模擬器，並請詳細參考課本第二章及附錄A的介紹，於QtSpim模擬器環境下，撰寫一完整的MIPS核心指令集版本的程式。(需**貼完整程式碼，截圖呈現結果並文字說明**。)

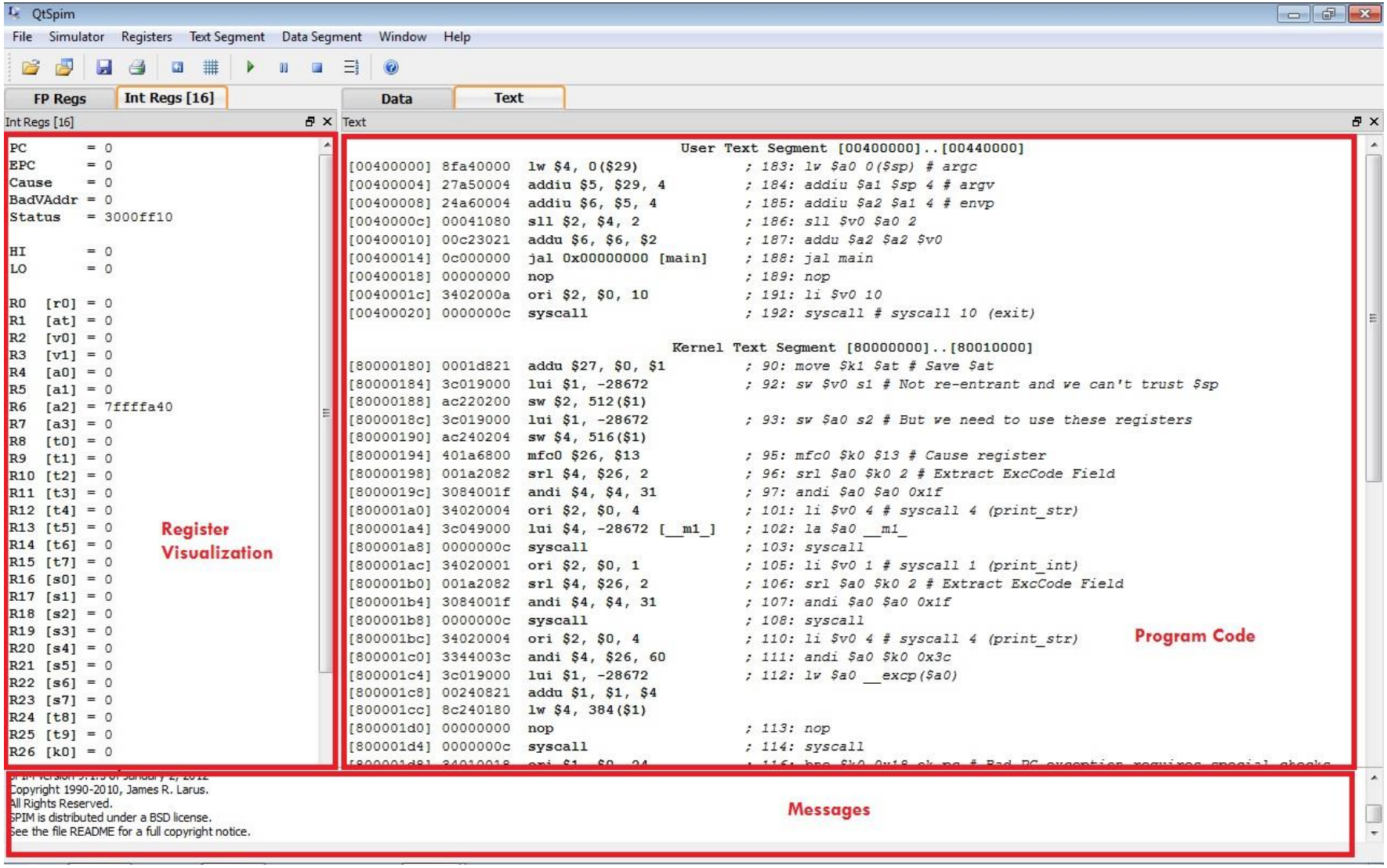
(1)實作第二章2.7小節範例if-then-else (中英文版90頁)，請自行完成變數設定，觀察暫存器及記憶體狀態並說明程式之運作。(50分)

(2)實作第二章2.7小節範例while迴圈 (中英文版92頁)，請自行完成變數設定，觀察暫存器及記憶體狀態並說明程式之運作。(50分)

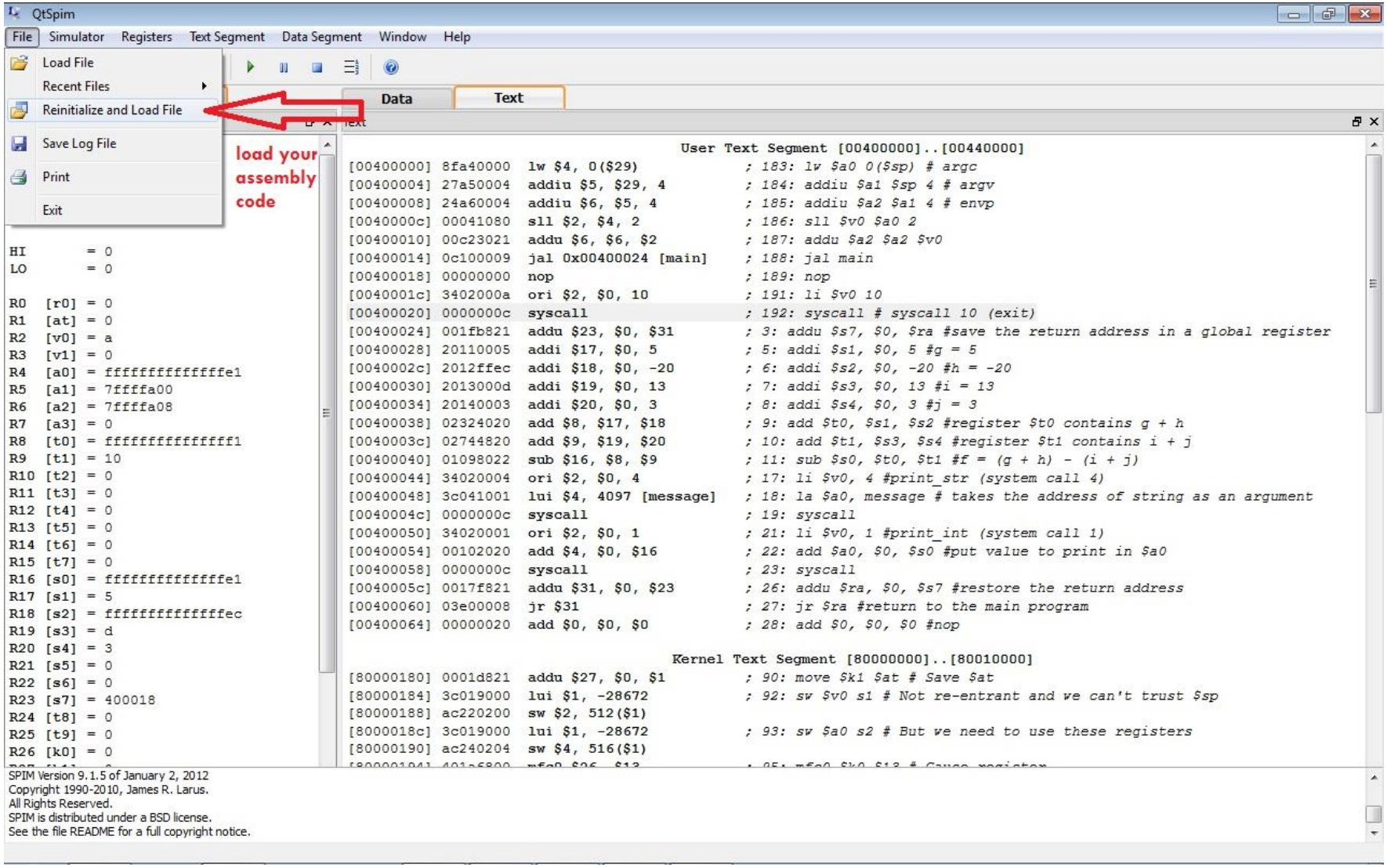
QtSpim

- *spim* is a simulator that runs MIPS32 programs
- It's been around for more than 20 years (improving over time).
- QtSpim is a new interface for *spim* built on the Qt UI framework which supports various platforms (Windows, Mac, Linux)
- It reads and executes assembly language programs.
- It contains a simple debugger

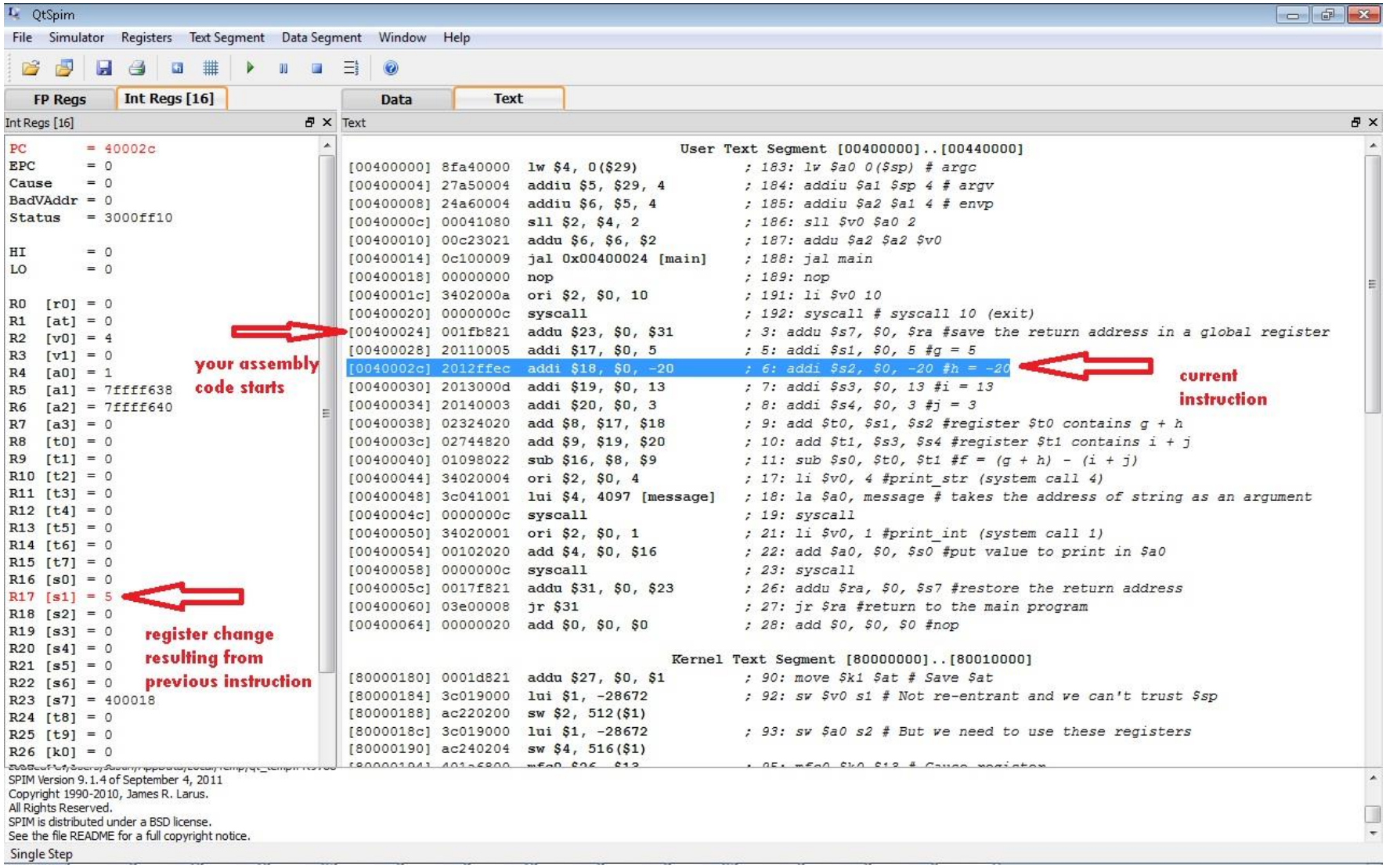
Start SPIM


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text
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] = 0
R5 [a1] = 0
R6 [a2] = 7ffffa40
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
R26 [k0] = 0
Register Visualization
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
[800001d4] 0000000c syscall ; 114: syscall
Program Code
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.
Messages

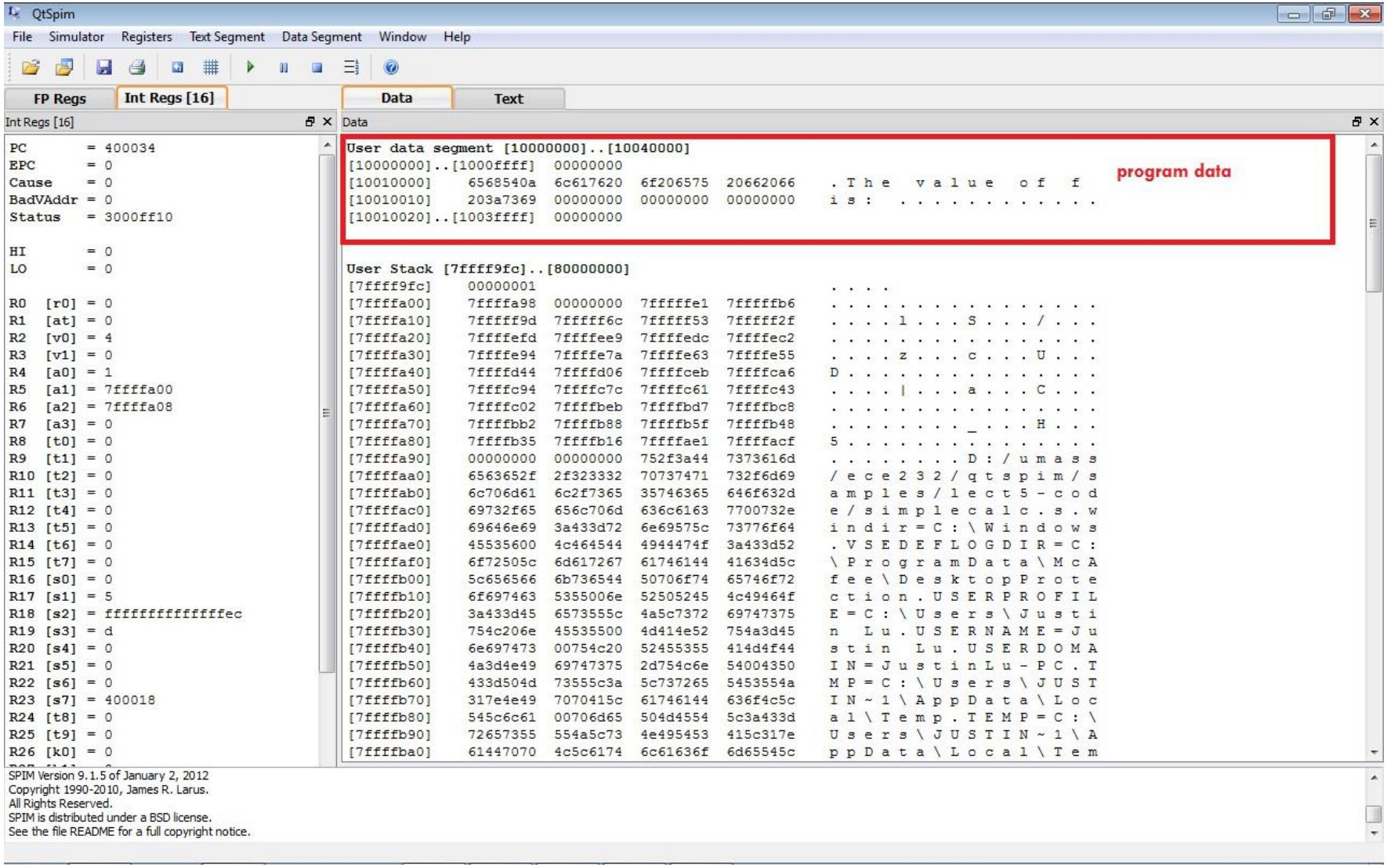
Load Program



Execute Program

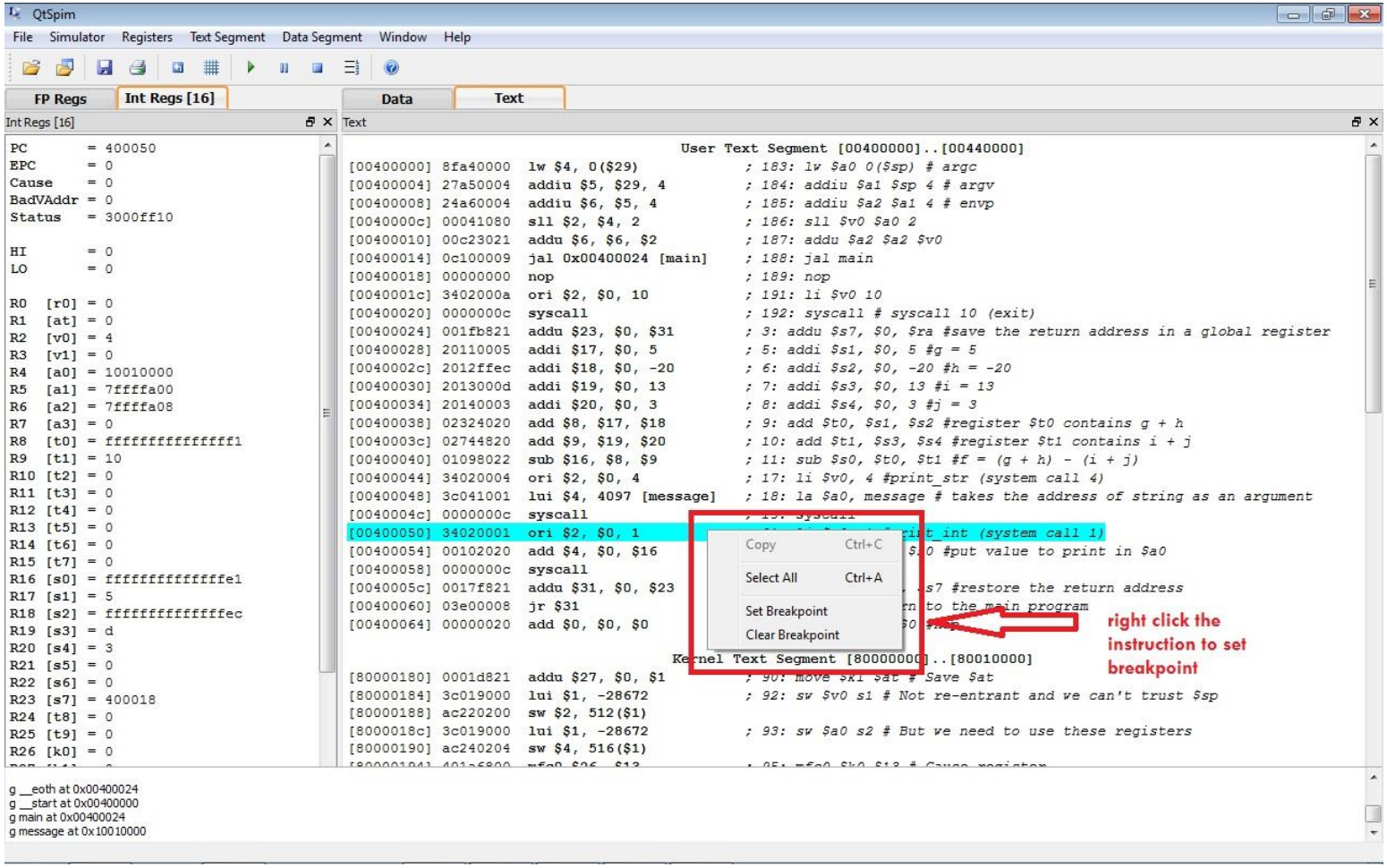


Program data

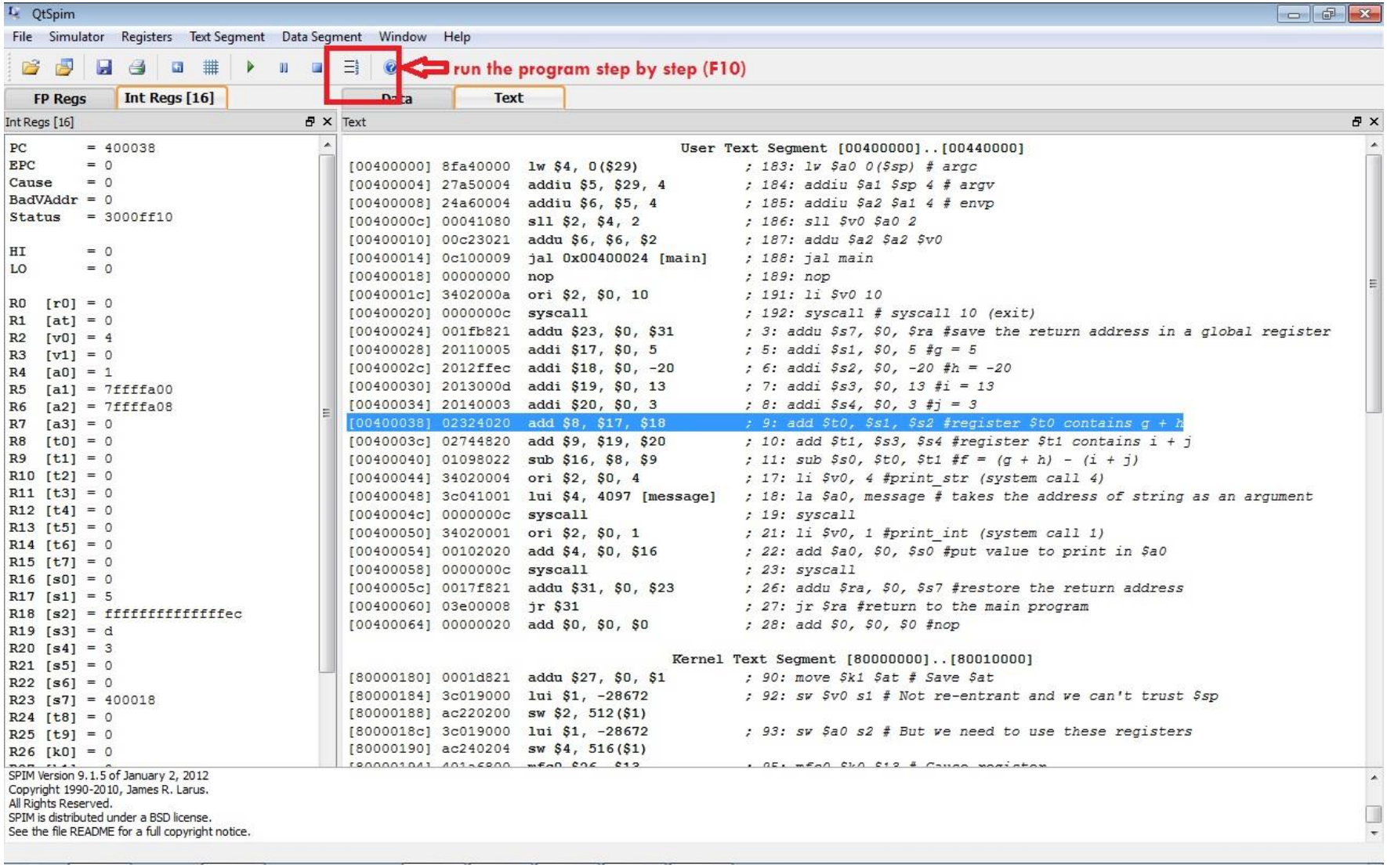


Set a break point

- Set a break point at the conditional instruction



Debug by stepping your code line by line



MIPS Hello World

```
# Hello, World!
        .data           ## Data declaration section
## String to be printed:
out_string: .asciiz "\nHello, World!\n"
        .text           ## Assembly language instructions go in text segment
main:                   ## Start of code section
        li $v0, 4               # system call code for printing string = 4
        la $a0, out_string      # load address of string to be printed into $a0
        syscall                 # call operating system to perform operation
                                # specified in $v0
                                # syscall takes its arguments from $a0, $a1, ...
        li $v0, 10              # terminate program
        syscall
```