

MIPS Simulator - SPIM

Lab1 – MIPS Programming

Outline

- Introduction
- Download & Install
- How to use SPIM?
- Debug
- Demo

Introduction

- SPIM是一種MIPS processor simulator, 其主要用來執行assembly language for MIPS.而此次作業希望同學能熟悉MIPS assembly language的implement並透過SPIM來simulate MIPS assembly language.

Download & Install

- Download: <http://spimsimulator.sourceforge.net/>

SPIM: A MIPS32 Simulator

James Larus
spim@larusstone.org

Spim is a self-contained simulator that runs MIPS32 programs. It reads and executes assembly language programs written for this processor. *Spim* also provides a simple debugger and minimal set of operating system services. *Spim* does not execute binary (compiled) programs.

Spim implements almost the entire MIPS32 assembler-extended instruction set. (It omits most floating point comparisons and rounding modes and the memory system page tables.) The MIPS architecture has several variants that differ in various ways (e.g., the MIPS64 architecture supports 64-bit integers and addresses), which means that *Spim* will not run programs for all MIPS processors.


Spim comes with complete source code and documentation.























Spim implements both a terminal and windows interfaces. On Microsoft Windows, Linux, and Mac OS X, the *spim* program offers a simple terminal interface and the *QtSpim* program provides the windowing interface. The [older programs xspim and PCSpim](#) provide native window interfaces for these systems as well.

[Download SPIM](#)

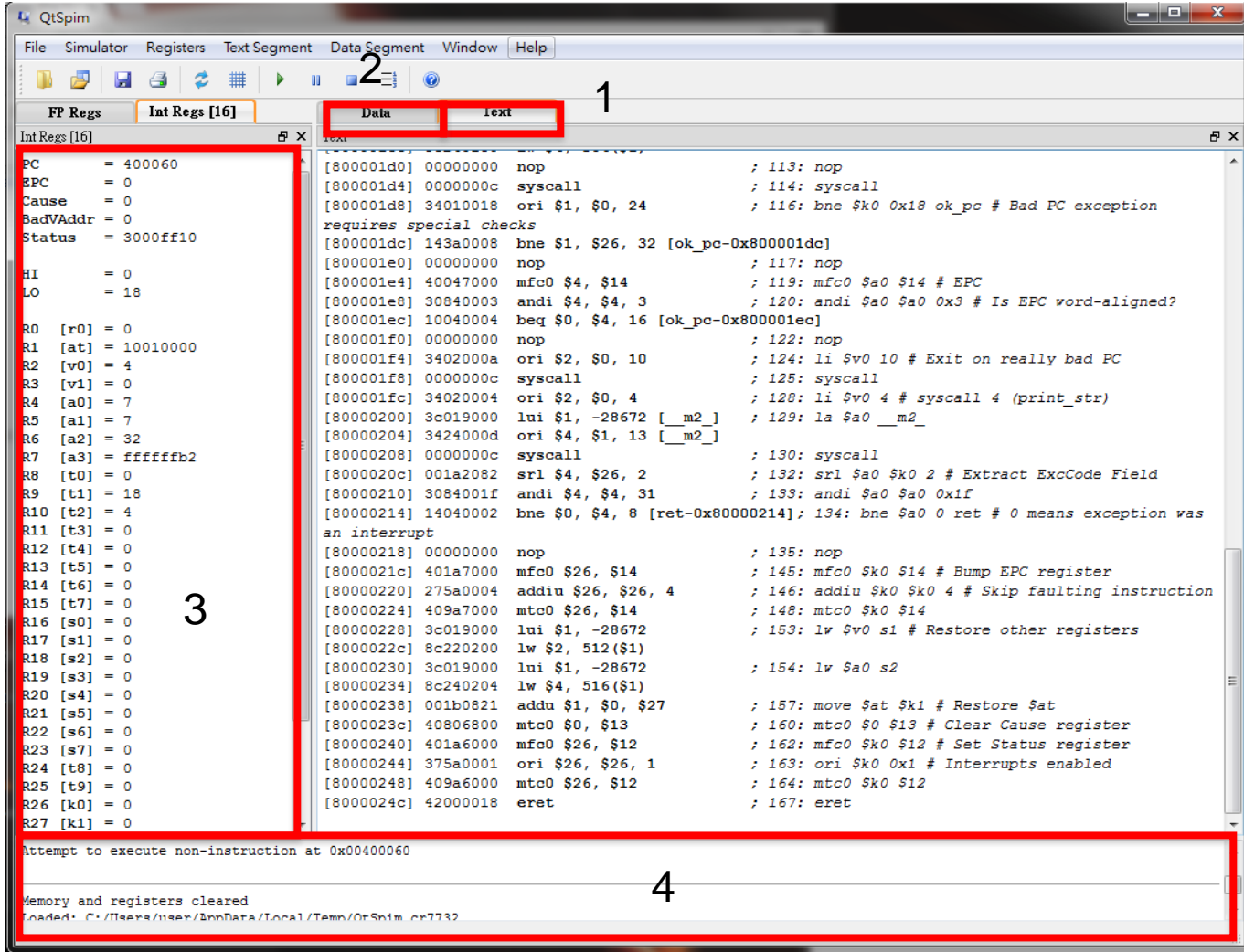
- Download the version you want, then install it.

Looking for the latest version? [Download QtSpim_9.1.17_Windows.exe \(34.3 MB\)](#)

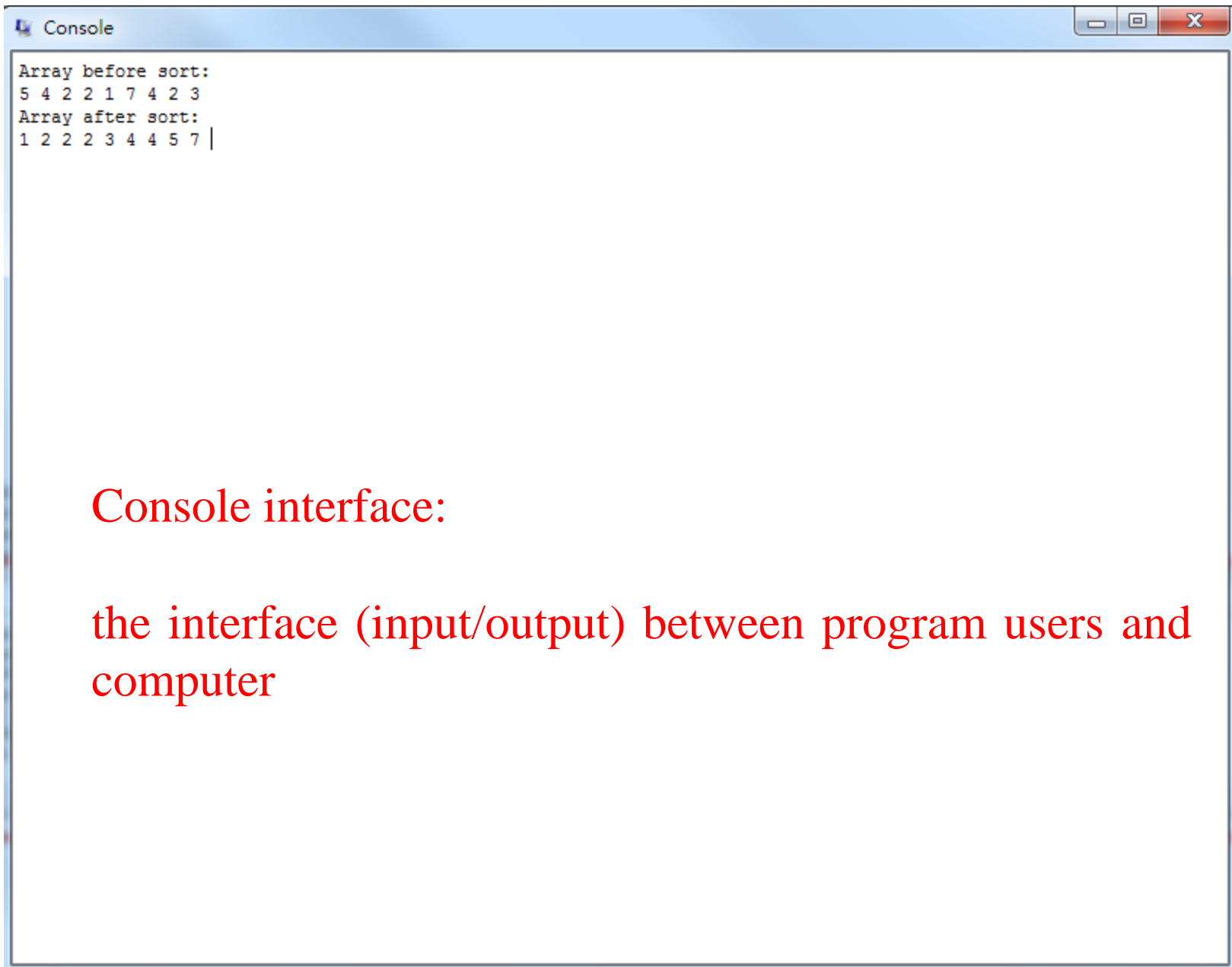
Home 

Name ↕	Modified ↕	Size ↕	Downloads / Week ↕	
QtSpim_9.1.18_mac.mpkg.zip	2017-01-20	28.8 MB	398 	
readme.md	2017-01-05	804 Bytes	3 	
readme.txt	2017-01-05	804 Bytes	50 	
QtSpim_9.1.18_Windows.exe	2017-01-03	34.3 MB	822 	
qtspim_9.1.18_linux64.deb	2017-01-03	19.8 MB	114 	
QtSpim_9.1.17_Windows.exe	2015-12-30	34.3 MB	824 	
qtspim_9.1.17_linux32.deb	2015-12-30	30.8 MB	67 	
qtspim_9.1.17_linux64.deb	2015-12-30	29.1 MB	14 	
QtSpim_9.1.17_mac.mpkg.zip	2015-12-30	28.9 MB	133 	
qtspim_9.1.16_linux32.deb	2015-08-26	29.7 MB	1 	
QtSpim_9.1.16_mac.mpkg.zip	2015-08-26	28.7 MB	8 	

How to use SPIM?



- 1:顯示instruction的地址
- 2:顯示data,例如：load進來的or stack裡的
- 3:MIPS中所有register的value(\$s1,\$t1.....)
- 4:顯示一些資訊，例如error message



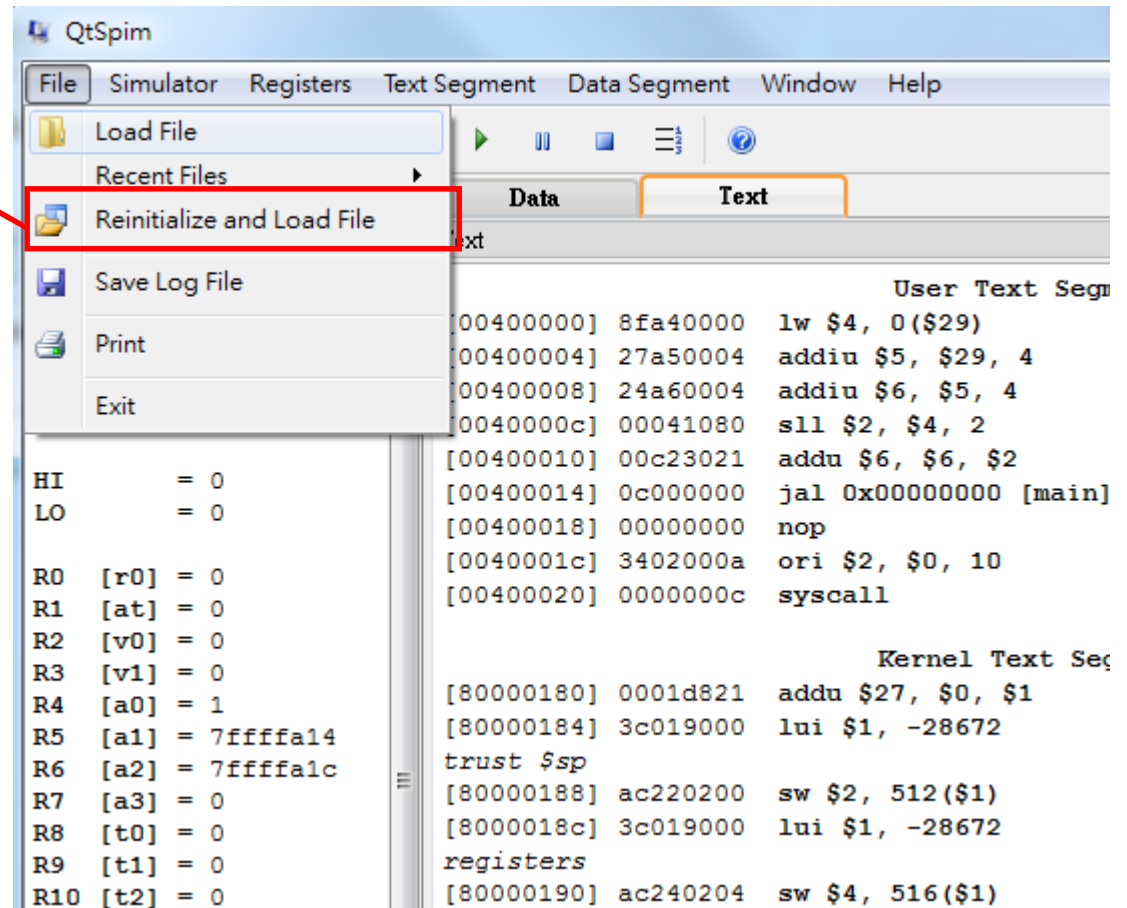
Console interface:

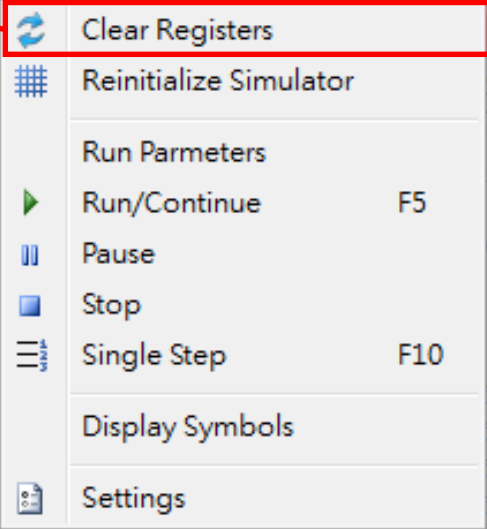
the interface (input/output) between program users and computer

Simple Steps

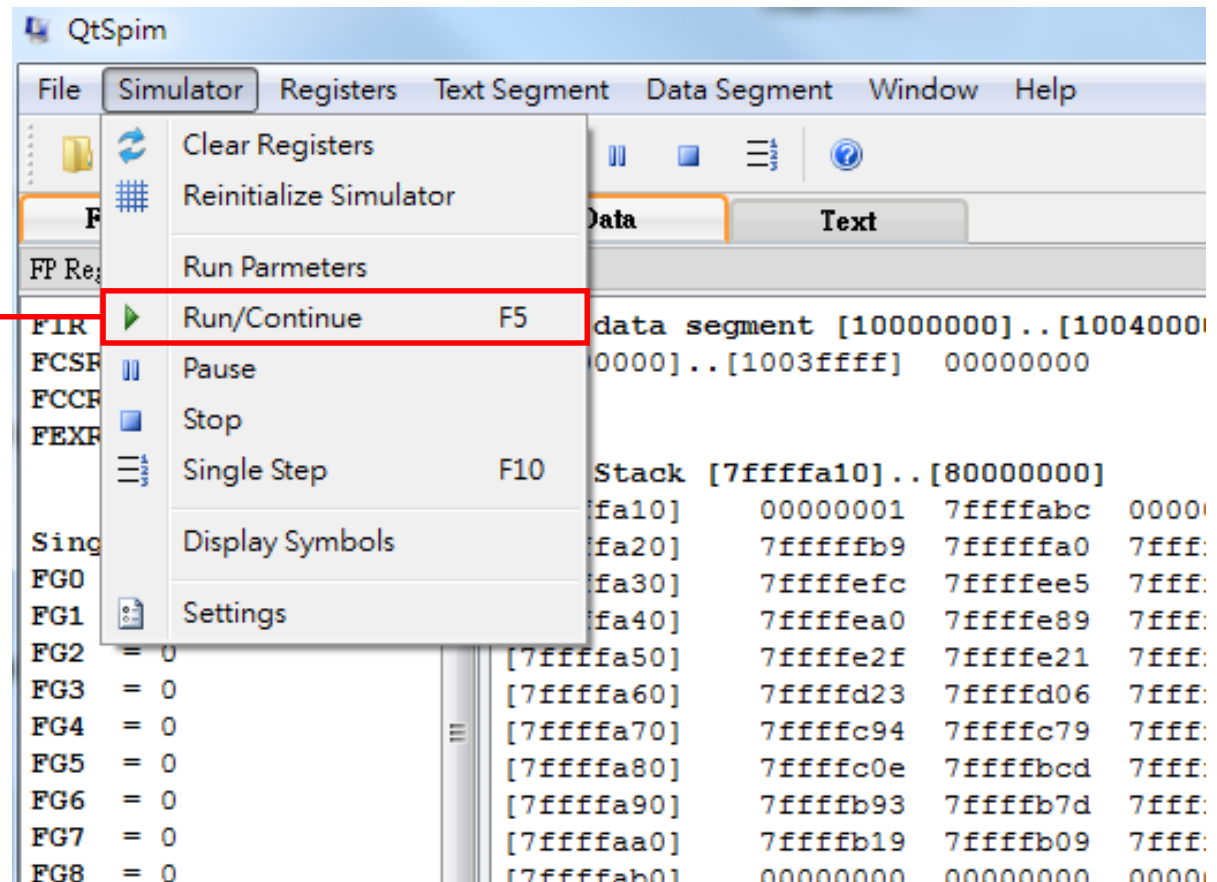
1. 新增一個.txt檔，並在上面撰寫MIPS code
 2. 把附檔名改成.s
 3. [File]->Reinitialize and Load File
 4. [Simulator]->Clear Registers
 5. [Simulator]->[Run]
- 有時候鍵盤右邊的數字鍵會讀不到數字，如果有同學遇到這個問題，請改用字母上方的數字鍵。

Load file "XXX.s"





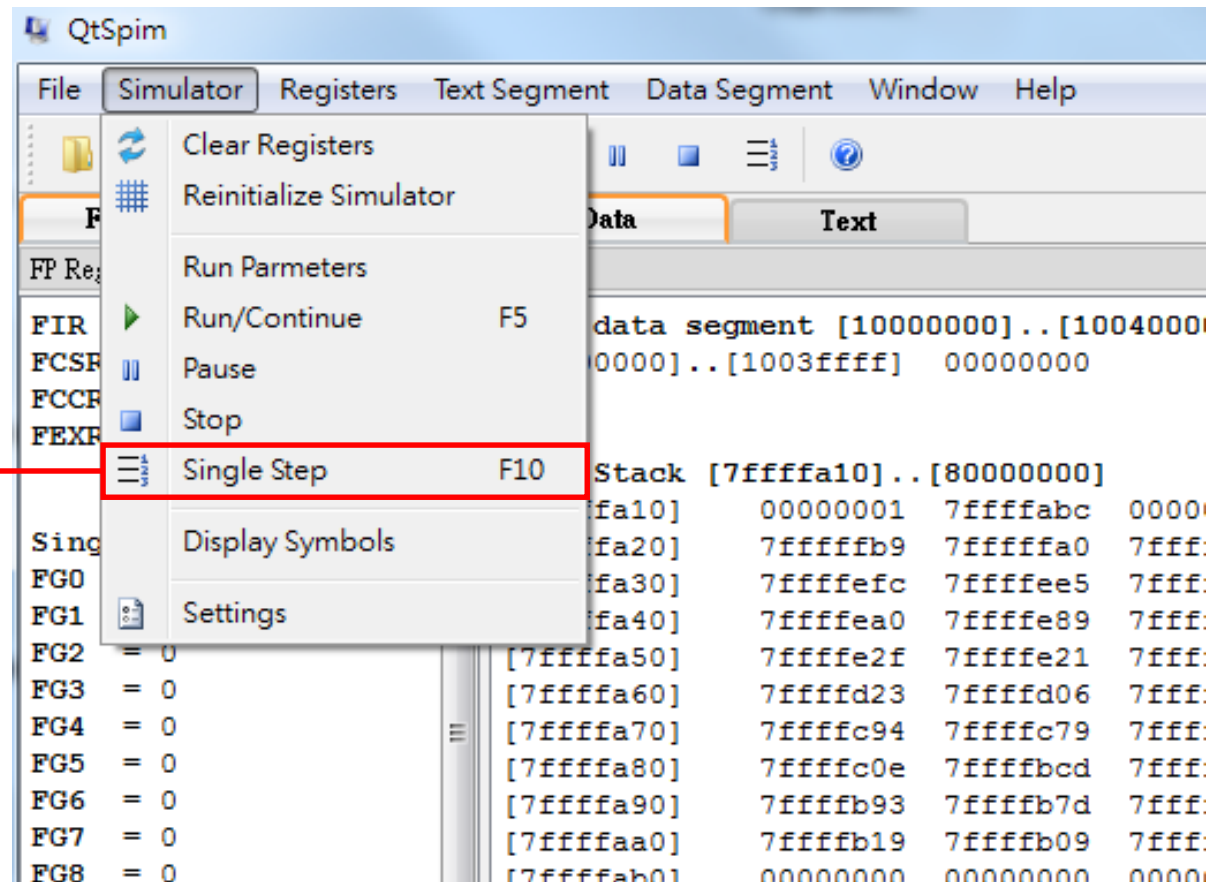
Run your program



Debugging

- Two methods:
 - Simulator -> single step(F10)
 - Set breakpoints

Run a single instruction
For debugging



Data

Text

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] 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] 34020004 ori \$2, \$0, 4 ; 5: li \$v0, 4 # system call: print string

[00400028] 3c011001 lui \$1, 4097 [str1] ; 6: la \$a0, str1 # address of string

[0040002c] 34240028 ori \$4, \$1, 40 [str1]

[00400030] 0000000c syscall ; 7: syscall

[00400034] 3c011001 lui \$1, 4097 [array_elements]; 9: la \$a1, array_elements

[00400038] 34250024 ori \$5, \$1, 36 [array_elements]

[0040003c] 8ca50000 lw \$5, 0(\$5) ; 10: lw \$a1, 0(\$a1)

[00400040] 3c011001 lui \$1, 4097 [array] ; 12: la \$t0, array # address of array

[00400044] 34280000 ori \$8, \$1, 0 [array]

[00400048] 00005021 addu \$10, \$0, \$0 ; 13: move \$t2, \$zero

[0040004c] 0145482a slt \$9, \$10, \$5 ; 14: slt \$t1, \$t2, \$a1

[00400050] 1120000b beq \$9, \$0, 44 [end_print1-0x00400005]

[00400054] 8d040000 lw \$4, 0(\$8) ; 17: lw \$a1, 0(\$a1)

[00400058] 34020001 ori \$2, \$0, 1 ; 18: li \$v0, 1

[0040005c] 0000000c syscall ; 19: syscall

[00400060] 3c011001 lui \$1, 4097 [space] ; 21: la \$a1, space

[00400064] 34240052 ori \$4, \$1, 82 [space]

[00400068] 34020004 ori \$2, \$0, 4 ; 22: li \$v0, 4

[0040006c] 0000000c syscall ; 23: syscall

[00400070] 214a0001 addi \$10, \$10, 1 ; 25: addi \$t2, \$t2, 1 # i ++

[00400074] 21080004 addi \$8, \$8, 4 ; 26: addi \$t0, \$t0, 4

[00400078] 0c100013 jal 0x0040004c [print1] ; 27: jal print1

[0040007c] 3c011001 lui \$1, 4097 [array] ; 29: la \$a0, array

[00400080] 34240000 ori \$4, \$1, 0 [array]

[00400084] 0c100041 jal 0x00400104 [sort] ; 31: jal sort

- Click the right button
- Set Breakpoints

Copy Ctrl+C

Select All Ctrl+A

Set Breakpoint

Clear Breakpoint

Run your program, then it will stop at the breakpoints you set.