



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY



# Introduction to QTSpim

Name: Zhang Siming

Email: [jasmine.qw825@gmail.com](mailto:jasmine.qw825@gmail.com)

Tel: 138-1892-5464





# Agenda

---

- ① Introduction
  - ① Download and Installation
  - ① User Interface
  - ① Program Layout
  - ① Data Type
  - ① System Calls
  - ① Registers
  - ① Usage of QTSpim
-



## QTSpim

- A self-contained simulator that runs MIPS32 programs
- Reads and executes assembly language programs
- Provides a simple debugger and minimal set of operating system services
- Does not execute binary (compiled) programs



## Features

- Newest version of spim
  - Cross platform: Windows, Unix, Mac OS X
  - Open source
-



## Download

- <http://sourceforge.net/projects/spimsimulator/files/>  
(Latest Version)
- [http://sourceforge.net/projects/spimsimulator/files/PCSpim\\_9.0.zip/download](http://sourceforge.net/projects/spimsimulator/files/PCSpim_9.0.zip/download) (Older Version, known as PCSpim)



## Installation

- Double click the installation file
-



# User Interface

The image displays the QtSpim user interface, which is divided into several panes. The 'Regs' pane on the left shows the state of 32 MIPS registers, with the 'PC' (Program Counter) highlighted at 0. The 'Text' pane in the center displays the assembly code for the 'User Text Segment' and 'Kernel Text Segment'. The 'Console' pane on the right is currently empty. A blue oval labeled 'Register' points to the 'R0' register in the 'Regs' pane. Another blue oval labeled 'Text' points to the assembly code in the 'Text' pane. A third blue oval labeled 'Message' points to the copyright notice at the bottom of the window.

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

Regs Int Regs [16]

Int Regs [16]

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] = 0  
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  
R27 [k1] = 0

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]..[80040000]

```
[80000180] 0001d821 addu $27, $0, $1 ; 90: move $k1 $0
[80000184] 3c019000 lui $1, -28672 ; 92: sw $v0 $1
[80000188] ac220200 sw $2, 512($1) ; 93: sw $a0 $2
[8000018c] 3c019000 lui $1, -28672 ; 94: sw $a1 $1
[80000190] ac240204 sw $4, 516($1) ; 95: mfc0 $k0 $1
[80000194] 401a6800 mfc0 $26, $13 ; 96: srl $a0 $26
[80000198] 001a2082 srl $4, $26, 2 ; 97: andi $a0 $26, 0x1a2082
[8000019c] 3084001f andi $4, $4, 31 ; 98: andi $a0 $4, 0x3084001f
[800001a0] 34020004 ori $2, $0, 4 ; 101: li $v0 4
[800001a4] 3c049000 lui $4, -28672 [__m1_] ; 102: la $a0 $4
[800001a8] 0000000c syscall ; 103: syscall
[800001ac] 34020004 ori $2, $0, 4 ; 105: li $v0 4
[800001b0] 0000000c syscall ; 106: srl $a0 $26
[800001b4] 0000000c syscall ; 107: andi $a0 $26, 0x1a2082
[800001b8] 0000000c syscall ; 108: syscall
[800001bc] 34020004 ori $2, $0, 4 ; 110: li $v0 4
[800001c0] 3344003c andi $v0, $26, 0x3344003c ; 111: andi $a0 $26, 0x3344003c
[800001c4] 3c019000 lui $1, -28672 ; 112: lv $a0 $1
[800001c8] 00240821 addu $1, $1, $4 ; 113: nop
[800001cc] 8c240180 lw $4, 384($1) ; 113: nop
[800001d0] 00000000 nop ; 113: nop
```

Console

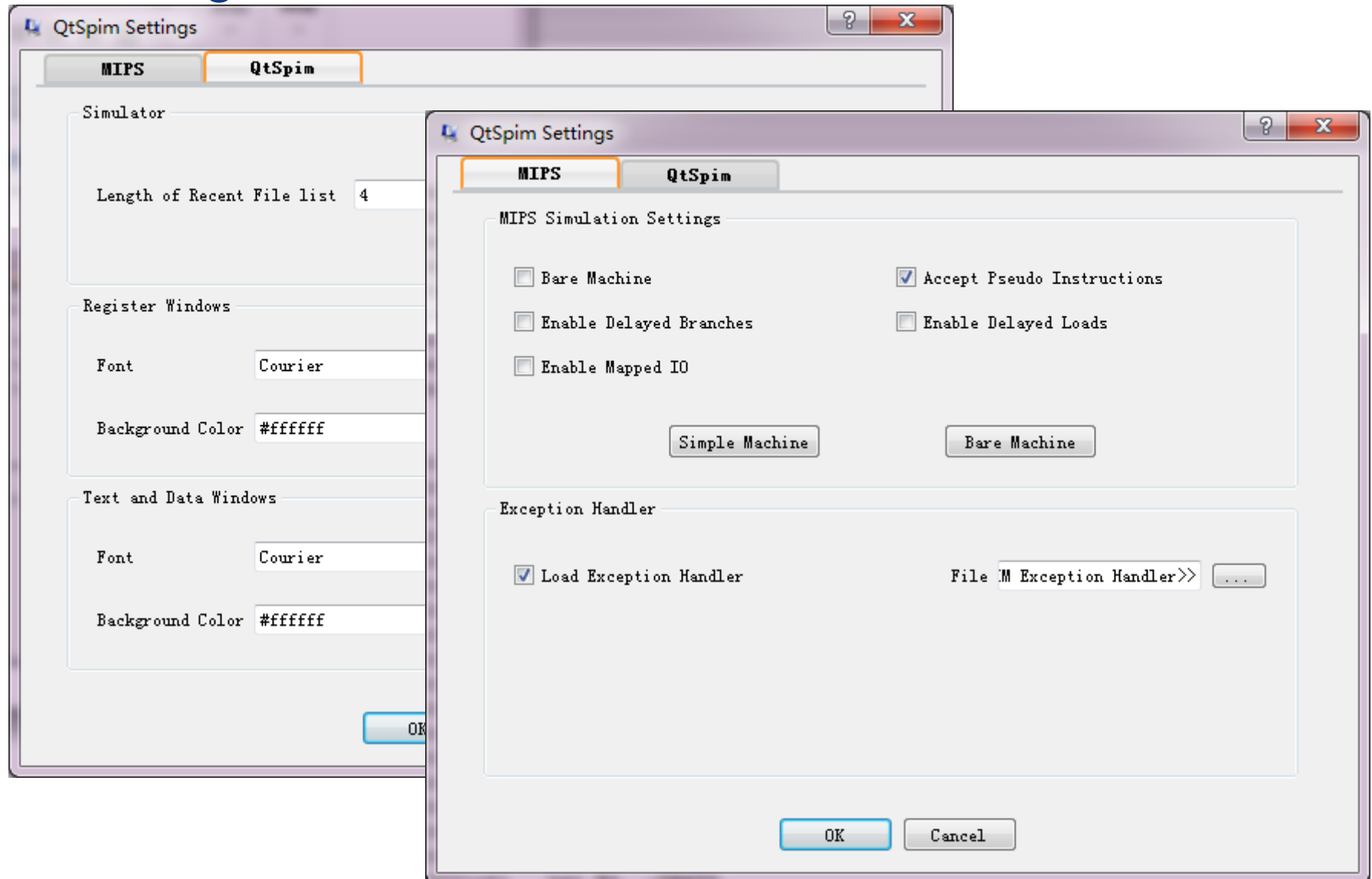
Copyright 1990-2012, James R. Larus.  
All Rights Reserved.  
SPIM is distributed under a BSD license.  
See the file README for a full copyright notice.



# User Interface



## Settings





# Program Layout

---

.data	#data section
data label	#data setting
.text	#code section
.globl main	#start point, must be globl
main:	
your code	

---



# Program Layout

.data            #data section  
data label      #data setting

.text            #code section  
.globl main      #start point:  
main:            your code

.data  
msg:    .asciiz "Hello World"

.text  
.globl main

main:

li \$v0, 4  
la \$a0, msg  
syscall  
li \$v0, 10  
syscall

System Call

Data Type





# Data Type

---

- ④ .data # set the data
  - ④ .text # code section
  - ④ .globl # where the program starts
  - ④ .word # 32 bit integer
  - ④ .half # 16 bit integer
  - ④ .byte: # 8 bit integer
  - ④ .ascii / .asciiz : # string
  - ④ .double / .float
-



# System Calls

Service	System Call Code	Args	Return
print_int	1	\$a0 = integer	
print_float	2	\$f12 = float	
print_double	3	\$f12 = double	
print_string	4	\$a0 = string	
read_int	5		integer(in \$v0)
read_float	6		float(in \$f0)
read_double	7		double(in \$f0)
read_string	8	\$a0 = buffer, \$a1 = length	
allocate	9	\$a0 = size to allocate	address(in \$v0)
exit	10		
print_char	11	\$a0 = char	
read_char	12		char(in \$a0)



# Registers

Register name	Number	Usage
\$zero	0	constant 0
\$at	1	reserved for assembler
\$v0	2	expression evaluation and results of a function
\$v1	3	expression evaluation and results of a function
\$a0	4	argument 1
\$a1	5	argument 2
\$a2	6	argument 3
\$a3	7	argument 4
\$t0	8	temporary (not preserved across call)
\$t1	9	temporary (not preserved across call)
\$t2	10	temporary (not preserved across call)
\$t3	11	temporary (not preserved across call)
\$t4	12	temporary (not preserved across call)
\$t5	13	temporary (not preserved across call)
\$t6	14	temporary (not preserved across call)
\$t7	15	temporary (not preserved across call)



# Registers

---

\$s0	16	saved temporary (preserved across call)
\$s1	17	saved temporary (preserved across call)
\$s2	18	saved temporary (preserved across call)
\$s3	19	saved temporary (preserved across call)
\$s4	20	saved temporary (preserved across call)
\$s5	21	saved temporary (preserved across call)
\$s6	22	saved temporary (preserved across call)
\$s7	23	saved temporary (preserved across call)
\$t8	24	temporary (not preserved across call)
\$t9	25	temporary (not preserved across call)
\$k0	26	reserved for OS kernel
\$k1	27	reserved for OS kernel
\$gp	28	pointer to global area
\$sp	29	stack pointer
\$fp	30	frame pointer
\$ra	31	return address (used by function call)

---



## Example

### HelloWorld:

.data            #data section  
data label      #data setting

.text            #code section  
.globl main      #start point:  
main:

your code

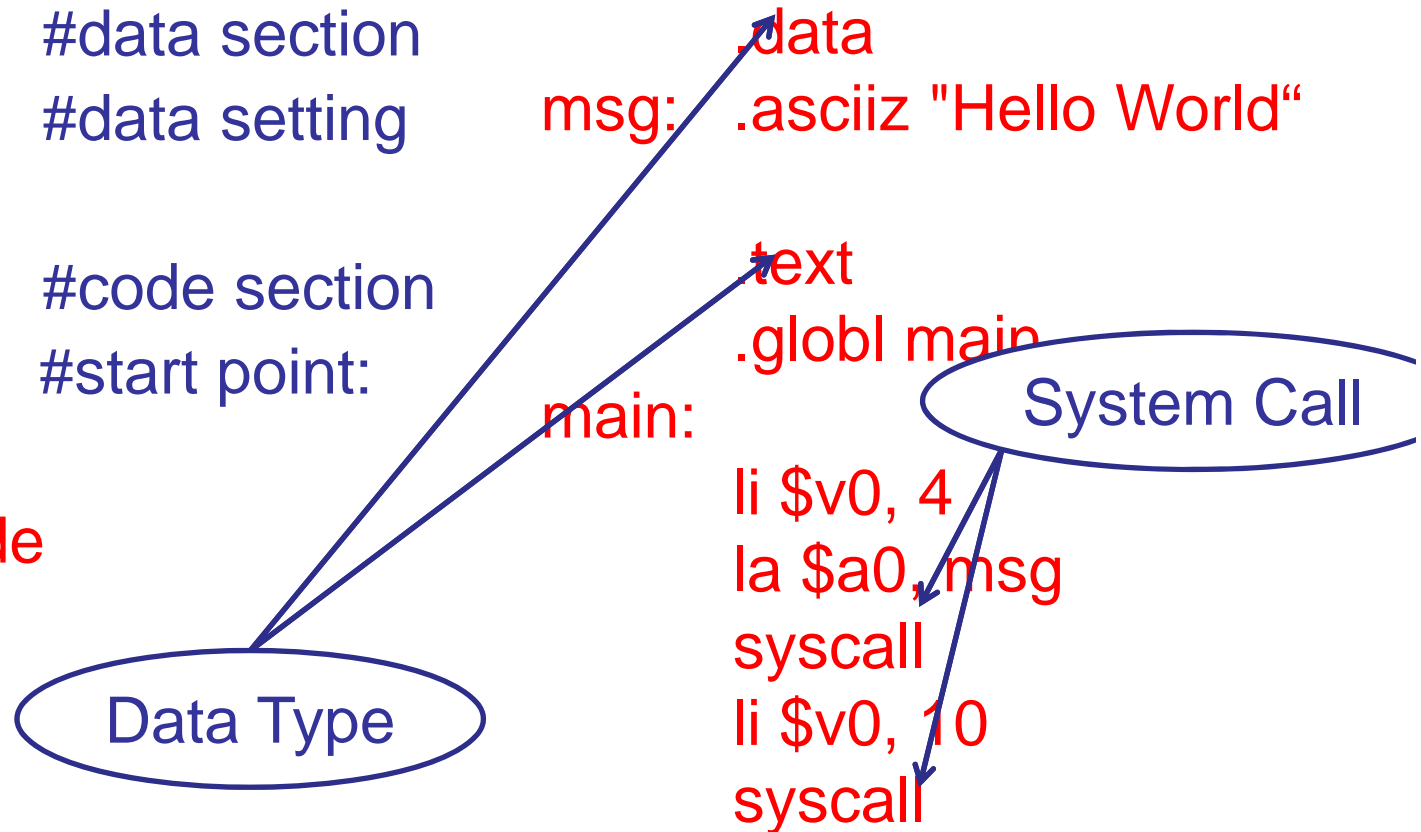
Data Type

msg:            .data  
                 .ascii "Hello World"

main:           .text  
                 .globl main

li \$v0, 4  
la \$a0, msg  
syscall  
li \$v0, 10  
syscall

System Call



## Example

### Addition:

```
.data  
value: .word 0,0,0  
msg: .asciiz "result = "
```

```
.text  
.globl main  
main: la $t0, value  
      li $v0, 5  
      syscall  
      sw $v0, 0($t0)
```

```
li $v0, 5  
syscall  
sw $v0, 4($t0)
```

```
lw $t1, 0($t0)  
lw $t2, 4($t0)  
add $t3, $t1, $t2  
sw $t3, 8($t0)
```

```
li $v0, 4  
la $a0, msg  
syscall
```

```
li $v0, 1  
move $a0, $t3  
syscall
```

```
li $v0, 10  
syscall
```



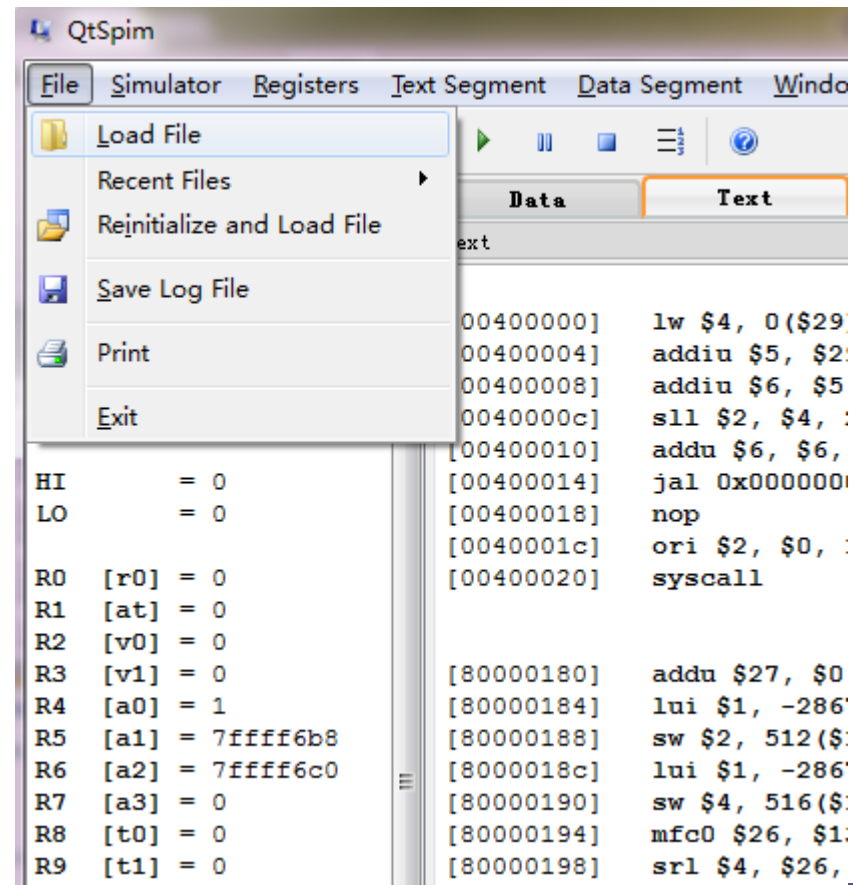
# Usage of QTSpim

---

- ④ Write your code in text editor, and save as “\*.s” file
  - ④ Load File from QTSpim
    - File -> Load File
  - ④ Run Your Program
    - Simulator -> Run/Continue or F5
    - ‘F10’ for single step
  - ④ Observe the Changes of Registers
-



- Load File or
- Reinitialize and Load File







- Run/Continue(F5) or
- Single Step(F10)

Observe the Registers

The image shows the QtSpim simulator interface. The 'Registers' tab is active, displaying the 'Int Regs [16]' window. The 'Text' tab is also visible, showing the 'User Text Segment' and 'Kernel Text Segment'. The 'Run/Continue' button (F5) is highlighted in the 'Registers' tab. The 'Single Step' button (F10) is also visible. The 'Status' register is highlighted with a blue arrow pointing to it from the 'Observe the Registers' text.

Reg	Value
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]	= 7ffff6b8
R6 [a2]	= 7ffff6c0
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

**User Text Segment**

```
[00400000] lw $4, 0($29)
[00400004] addiu $5, $29, 4
[00400008] addiu $6, $5, 4
[0040000c] sll $2, $4, 2
[00400010] addu $6, $6, $2
[00400014] jal 0x00000000 [main]
[00400018] nop
[0040001c] ori $2, $0, 10
[00400020] syscall
```

**Kernel Text Segment**

```
[80000180] addu $27, $0, $1
[80000184] lui $1, -28672
[80000188] sw $2, 512($1)
[8000018c] lui $1, -28672
[80000190] sw $4, 516($1)
[80000194] mfc0 $26, $13
[80000198] srl $4, $26, 2
[8000019c] andi $4, $4, 31
[800001a0] ori $2, $0, 4
[800001a4] lui $4, -28672 [__m1_]
[800001a8] syscall
[800001ac] ori $2, $0, 1
[800001b0] srl $4, $26, 2
[800001b4] andi $4, $4, 31
[800001b8] syscall
```



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY

# Exercise

---



Add from 1 to 20



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY

---

- Name: Zhang Siming
  - Email: [jasmine.qw825@gmail.com](mailto:jasmine.qw825@gmail.com)
  - Tel: 138-1892-5464
-



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY



# Thank You!

