上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Introduction to QTSpim

Name: Zhang Siming

Email: jasmine.qw825@gmail.com

Tel: 138-1892-5464

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Agenda

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

Introduction

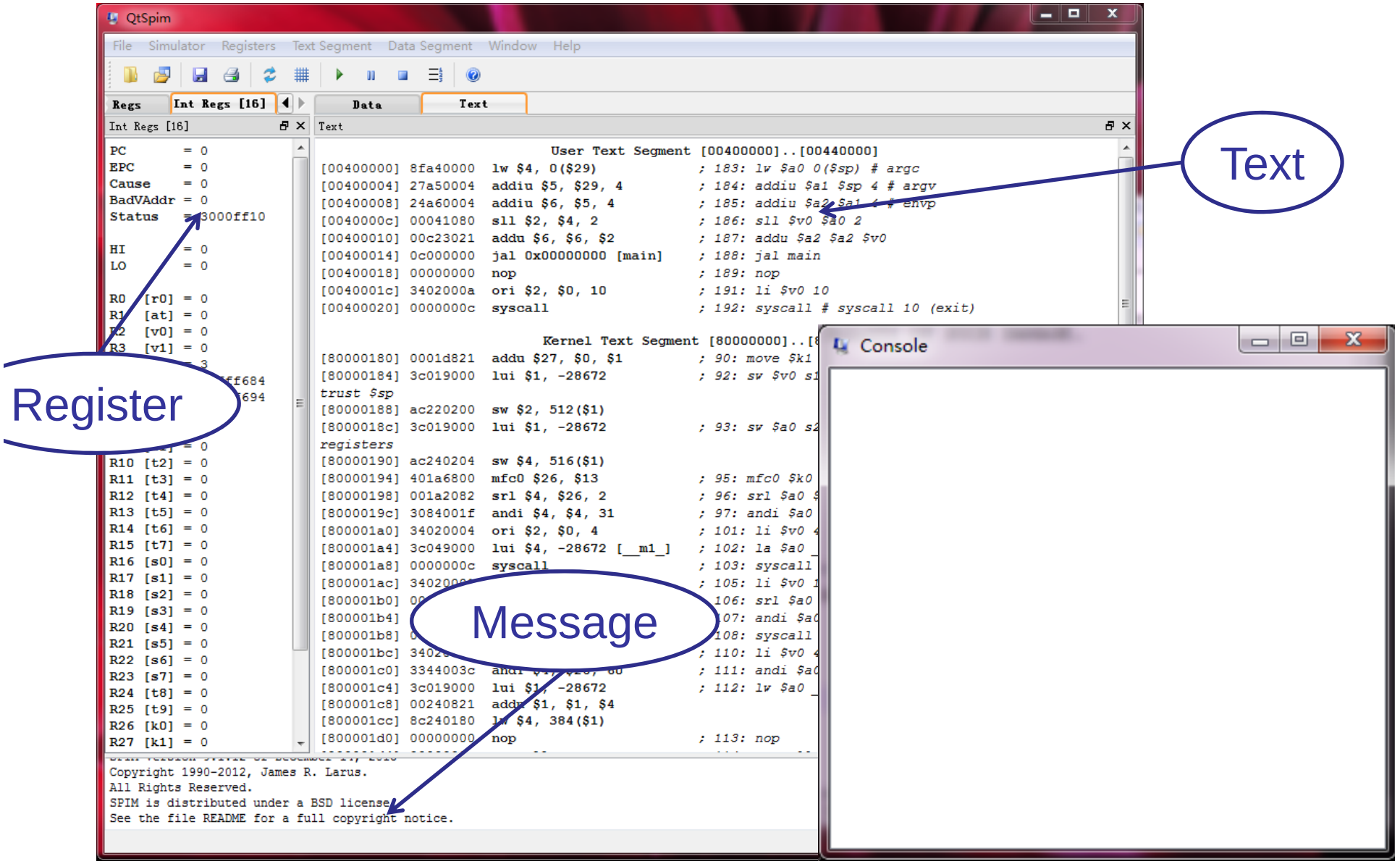
- QTSpim
  - A self-contained simulator that runs MIPS32 programs
  - Reads and executes assembly language programs
  - Provides a simple debugger and minimal set of operating system services
  - Does not execute binary (compiled) programs
- Features
  - Newest version of spim
  - Cross platform: Windows, Unix, Mac OS X
  - Open source

Download and Installation

- Download
  - http://sourceforge.net/projects/spimsimulator/files/ (Latest Version)
  - http://sourceforge.net/projects/spimsimulator/files/PCSpim_9.0.zip/download (Older Version, known as PCSpim)

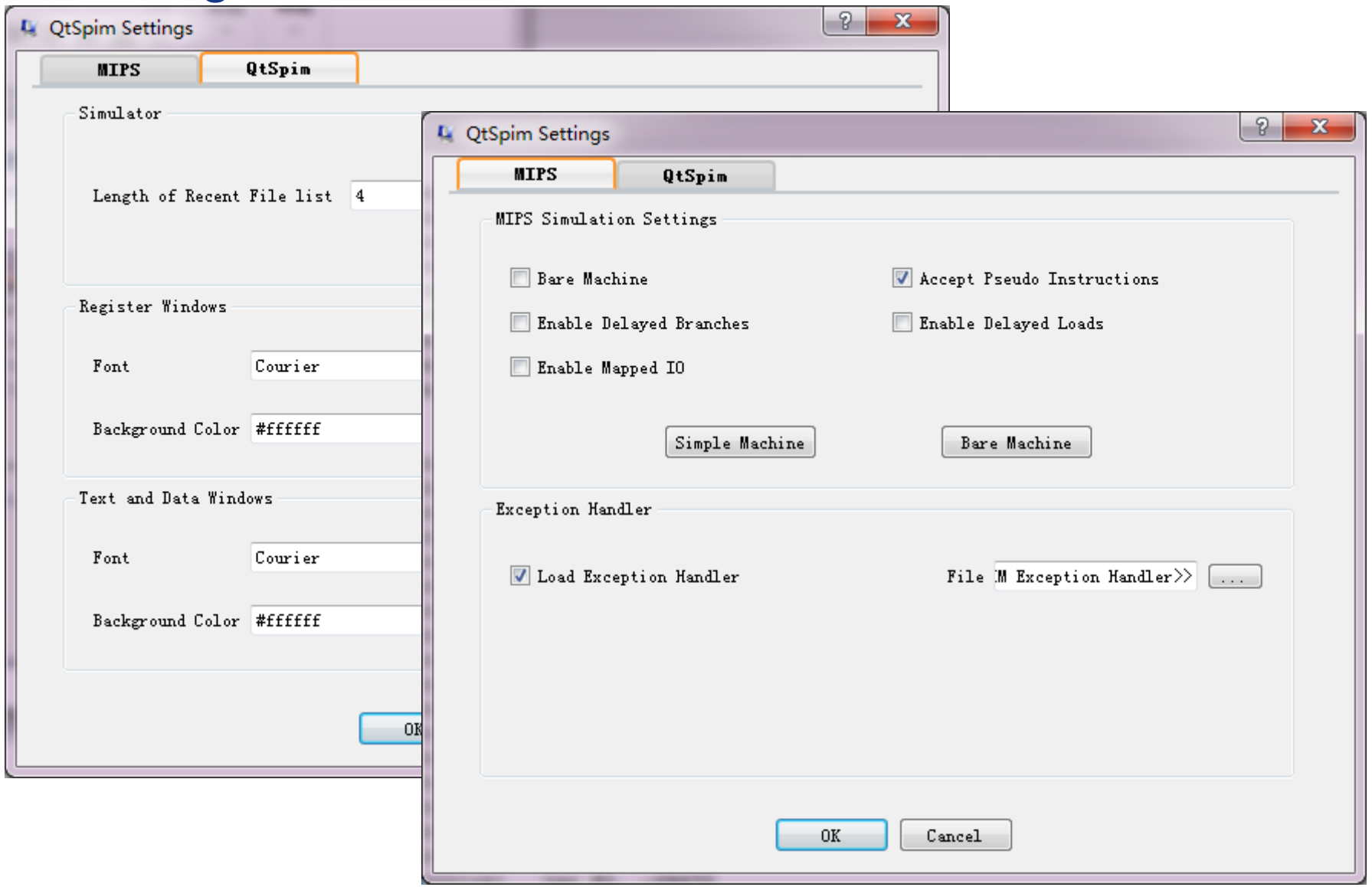
- Installation
  - Double click the installation file

User Interface



User Interface

Settings

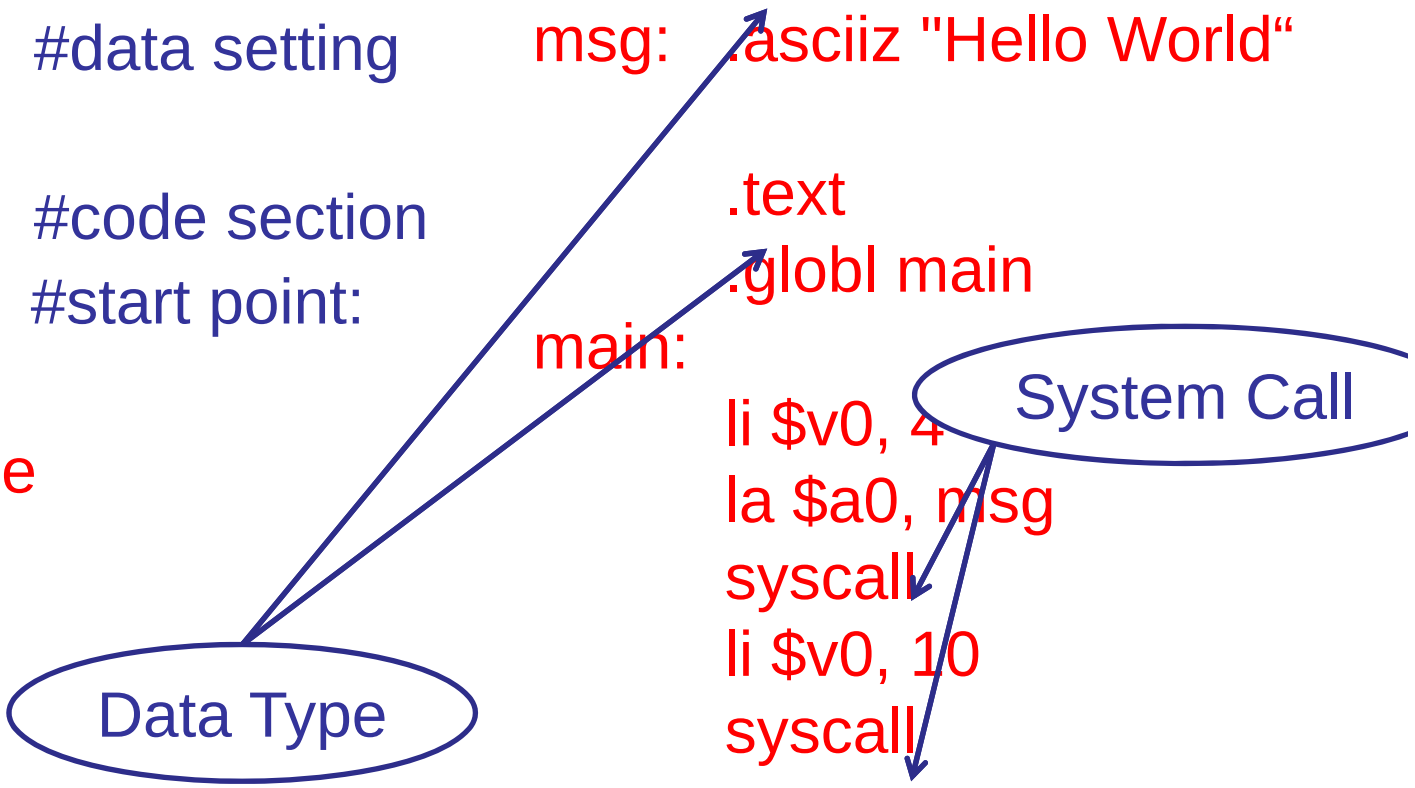


Program Layout

```
.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 Type

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

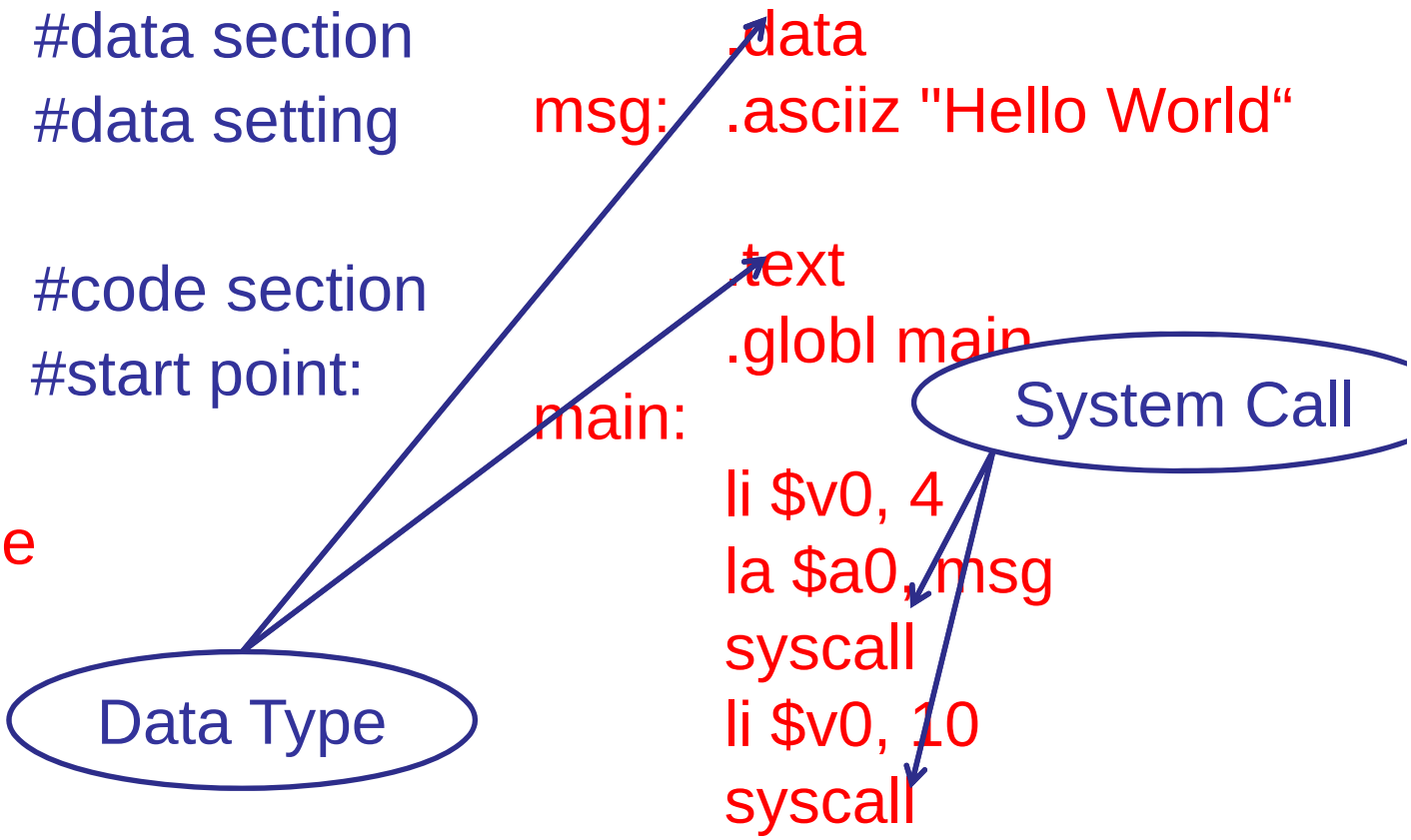
Registers

$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
```



Example

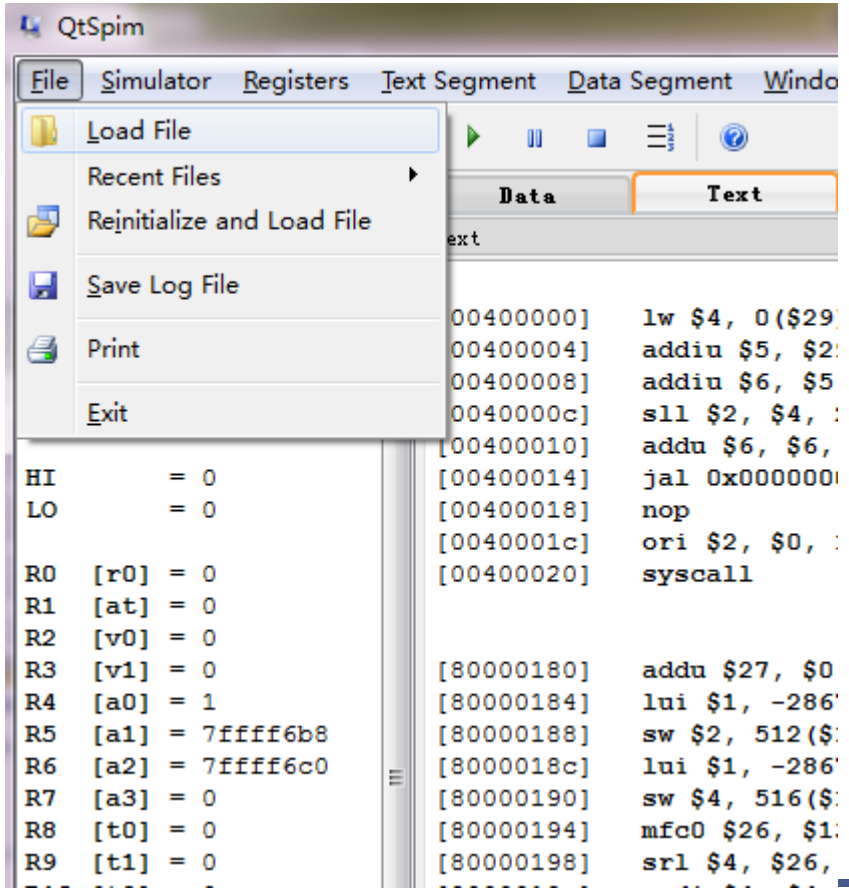
Addition:

```
        .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

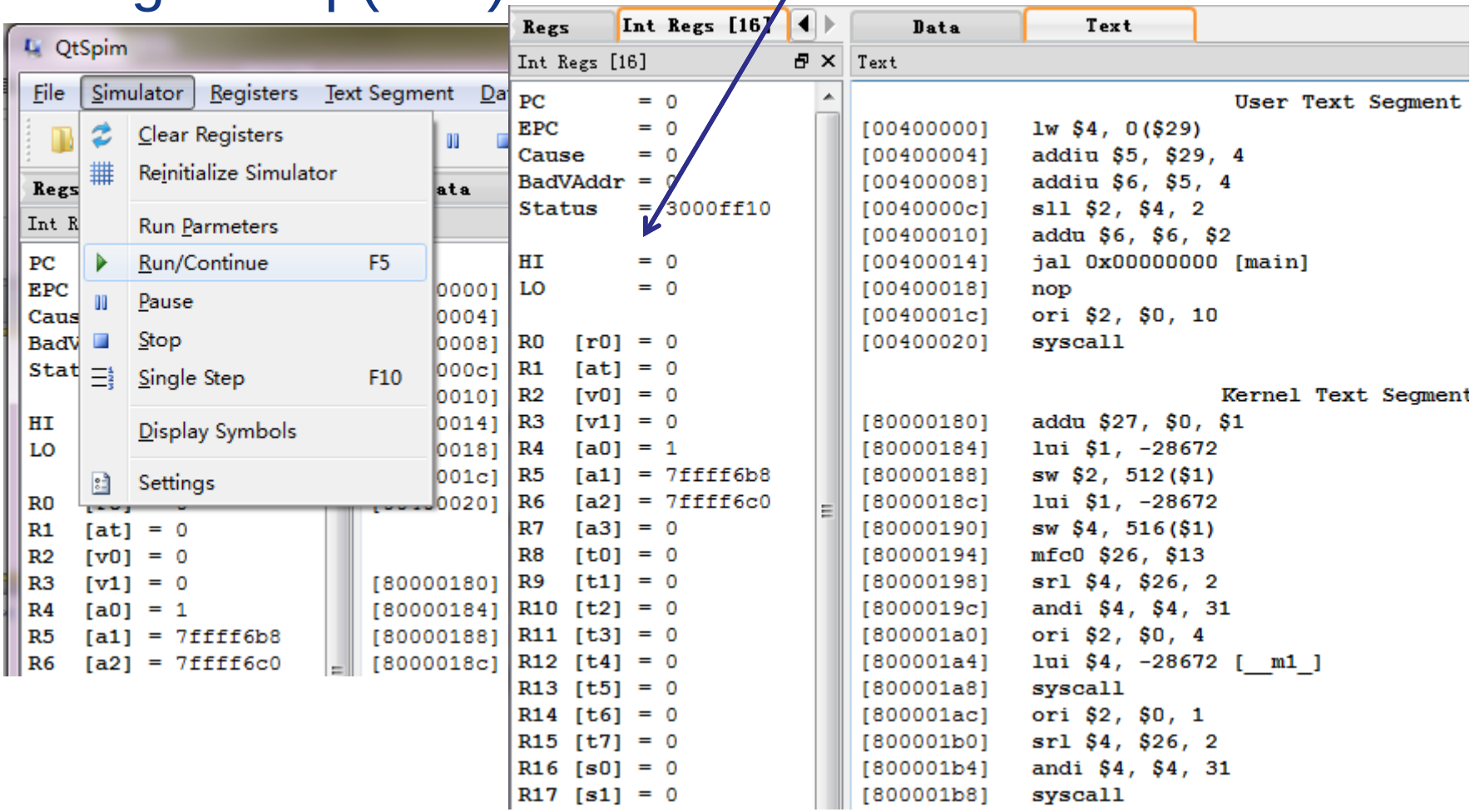
- Write your code in text editor, and save as “*.s” file
- Load File from QTSpim
  - File -> Load File
- Run Your Program
  - Simulator -> Run/Continue or F5
  - ‘F10’ for single step
- Observe the Changes of Registers

- Load File or
- Reinitialize and Load File



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

Observe the Registers



Exercise

- Add from 1 to 20

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

- Name: Zhang Siming
- Email: jasmine.qw825@gmail.com
- Tel: 138-1892-5464

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Thank You!