# Spim Download

http://sourceforge.net/projects/spimsimulator/files/

Latest version of Spim is QtSpim.

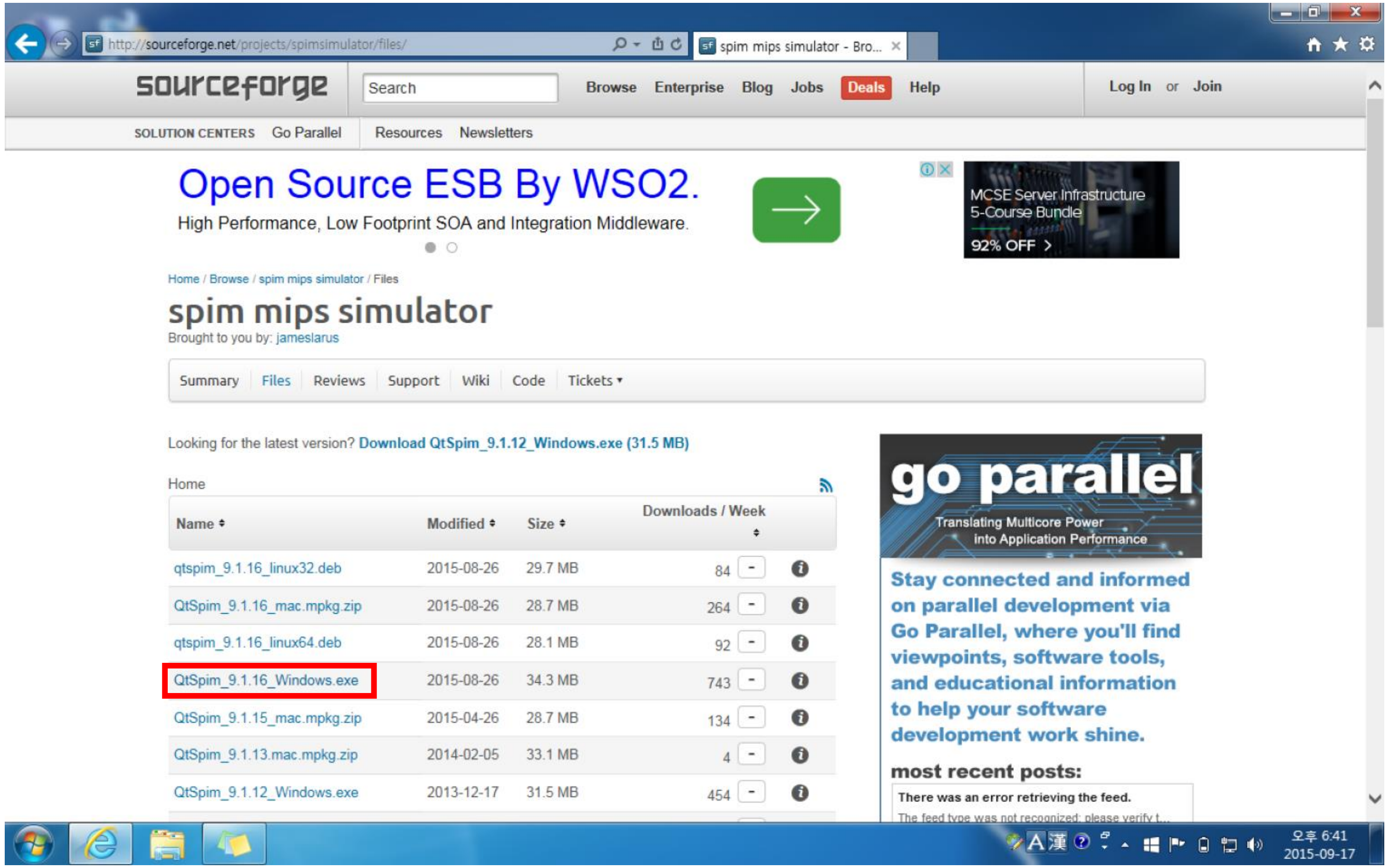
QtSpim is available on Microsoft Windows, Mac OS X and Linux environment.

Downloading previous version is possible, but using Qtspim is preferred.

Go to url above and download appropriate file for your OS.

- Linux – qtspim_9.1.16_linux32.deb (32-bit), qtspim_9.1.16_linux64.deb (64-bit)
- Mac OS X – QtSpim_9.1.16_mac.mpkg.zip
- Windows – QtSpim_9.1.16_Windows.exe

# Download and Install



http://sourceforge.net/projects/spimsimulator/files/

Home / Browse / spim mips simulator / Files

## spim mips simulator

Brought to you by: jameslarus

Looking for the latest version? Download QtSpim_9.1.12_Windows.exe (31.5 MB)

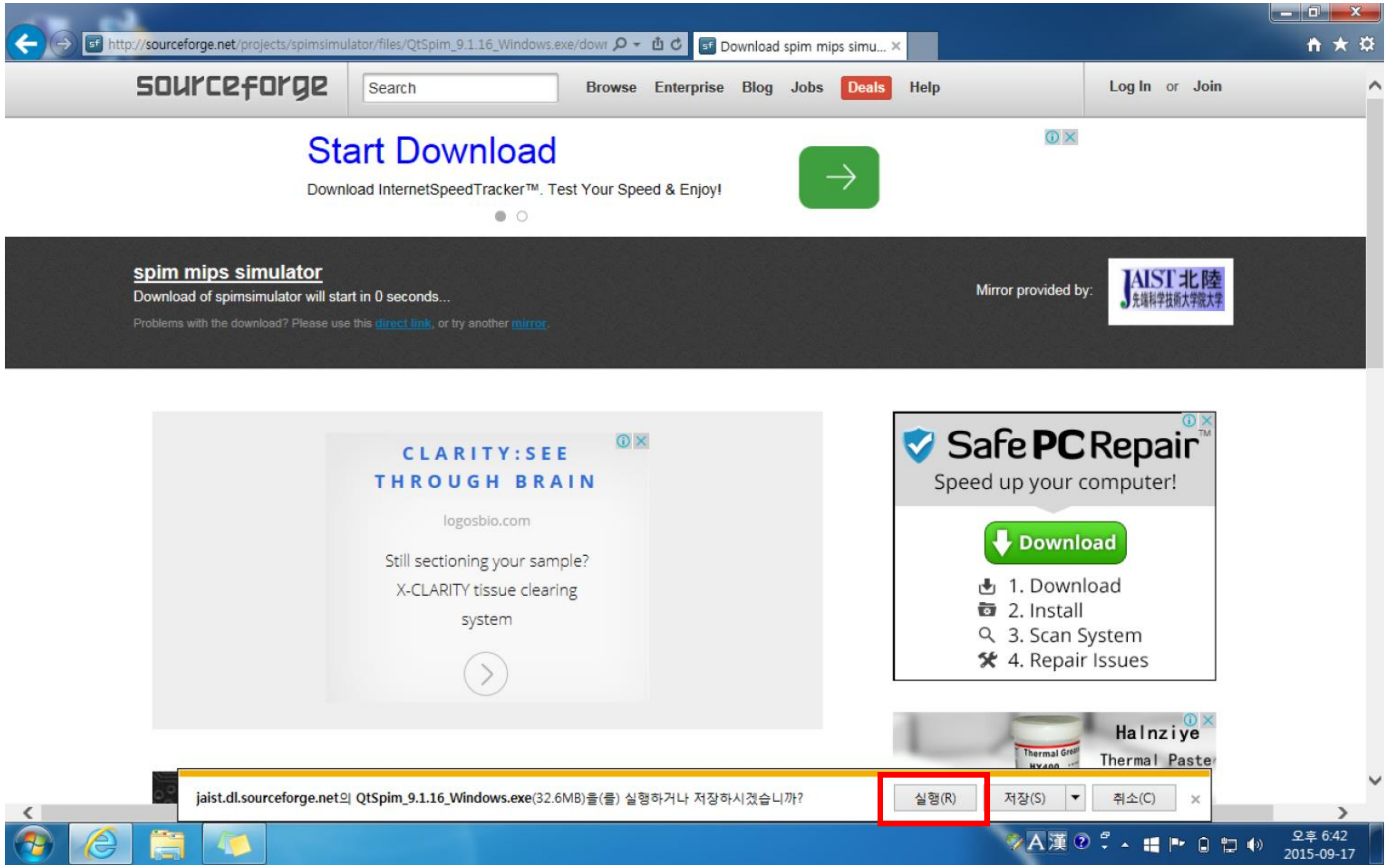
Home

| Name | Modified | Size | Downloads / Week |
|---|---|---|---|
| qtspim_9.1.16_linux32.deb | 2015-08-26 | 29.7 MB | 84 |
| QtSpim_9.1.16_mac.mpkg.zip | 2015-08-26 | 28.7 MB | 264 |
| qtspim_9.1.16_linux64.deb | 2015-08-26 | 28.1 MB | 92 |
| QtSpim_9.1.16_Windows.exe | 2015-08-26 | 34.3 MB | 743 |
| QtSpim_9.1.15_mac.mpkg.zip | 2015-04-26 | 28.7 MB | 134 |
| QtSpim_9.1.13.mac.mpkg.zip | 2014-02-05 | 33.1 MB | 4 |
| QtSpim_9.1.12_Windows.exe | 2013-12-17 | 31.5 MB | 454 |

most recent posts:

There was an error retrieving the feed.
The feed type was not recognized; please verify t...

# Download and Install



http://sourceforge.net/projects/spimsimulator/files/QtSpim_9.1.16_Windows.exe/dowr

spim mips simulator

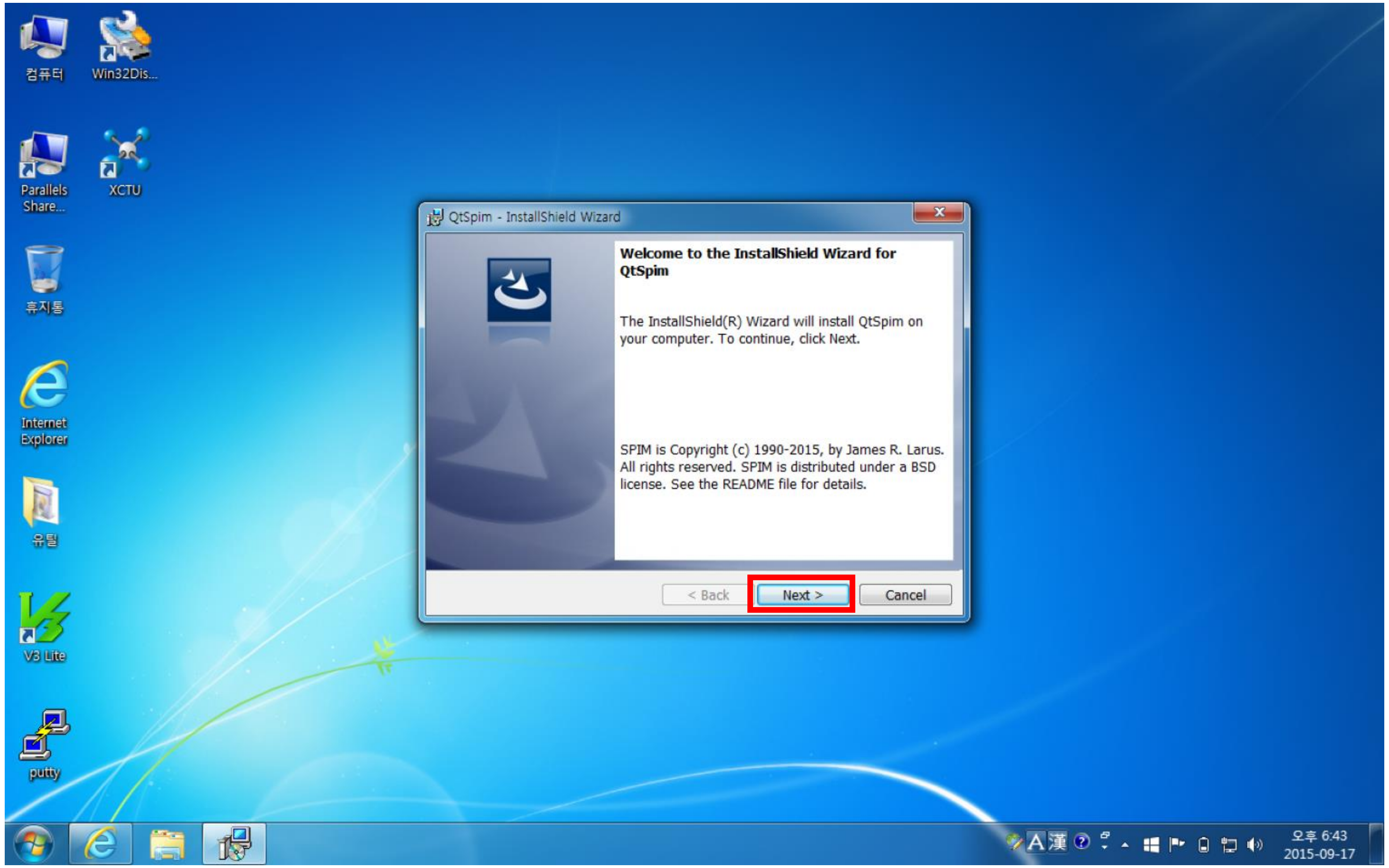
Download of spimsimulator will start in 0 seconds...

Problems with the download? Please use this direct link, or try another mirror.

Mirror provided by: JAIST 北陸先端科学技術大学院大学

jaist.dl.sourceforge.net의 QtSpim_9.1.16_Windows.exe(32.6MB)을(를) 실행하거나 저장하시겠습니까?

# Download and Install



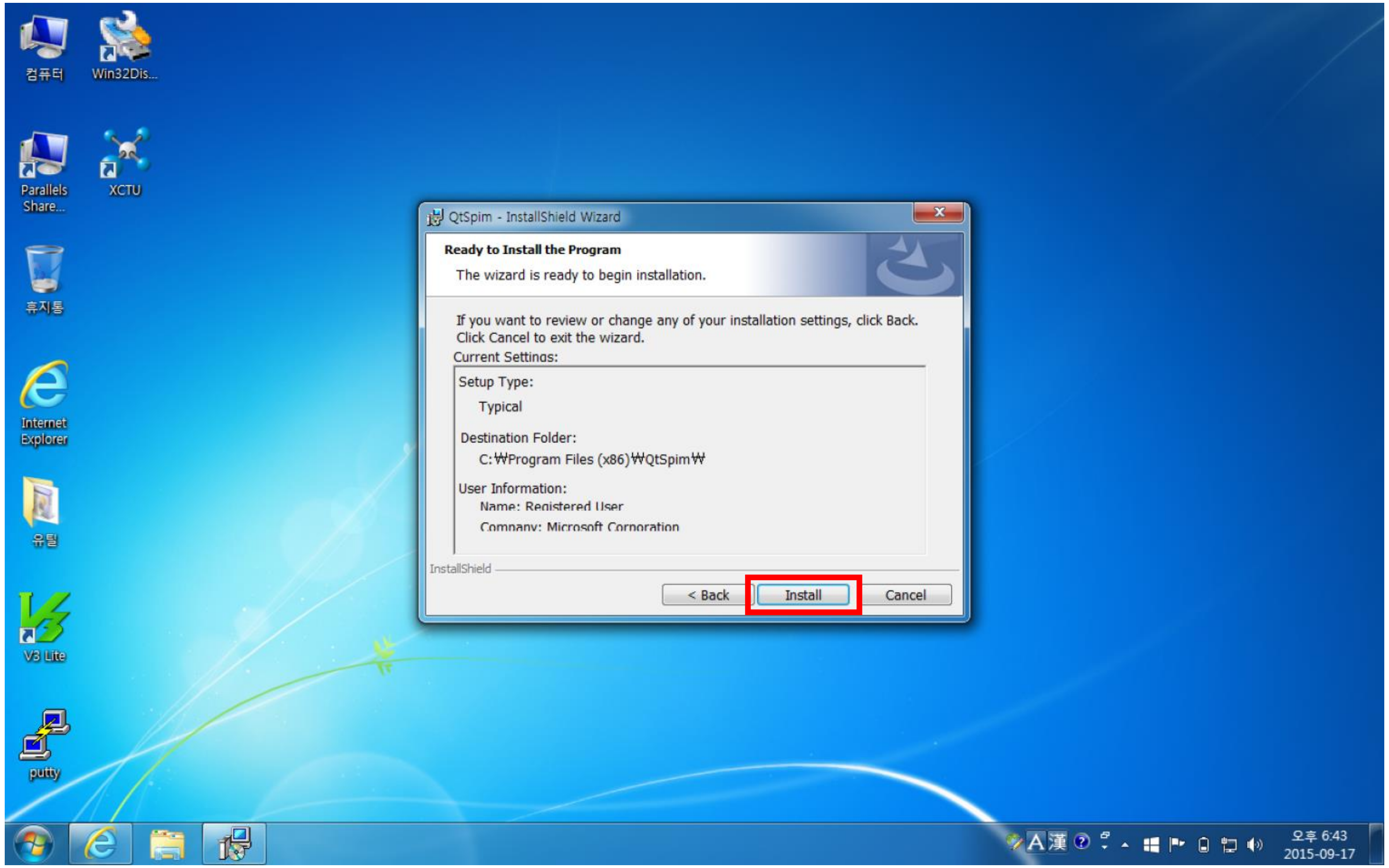
QtSpim - InstallShield Wizard

**Welcome to the InstallShield Wizard for QtSpim**

The InstallShield(R) Wizard will install QtSpim on your computer. To continue, click Next.

SPIM is Copyright (c) 1990-2015, by James R. Larus. All rights reserved. SPIM is distributed under a BSD license. See the README file for details.

# Download and Install



QtSpim - InstallShield Wizard

**Ready to Install the Program**

The wizard is ready to begin installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

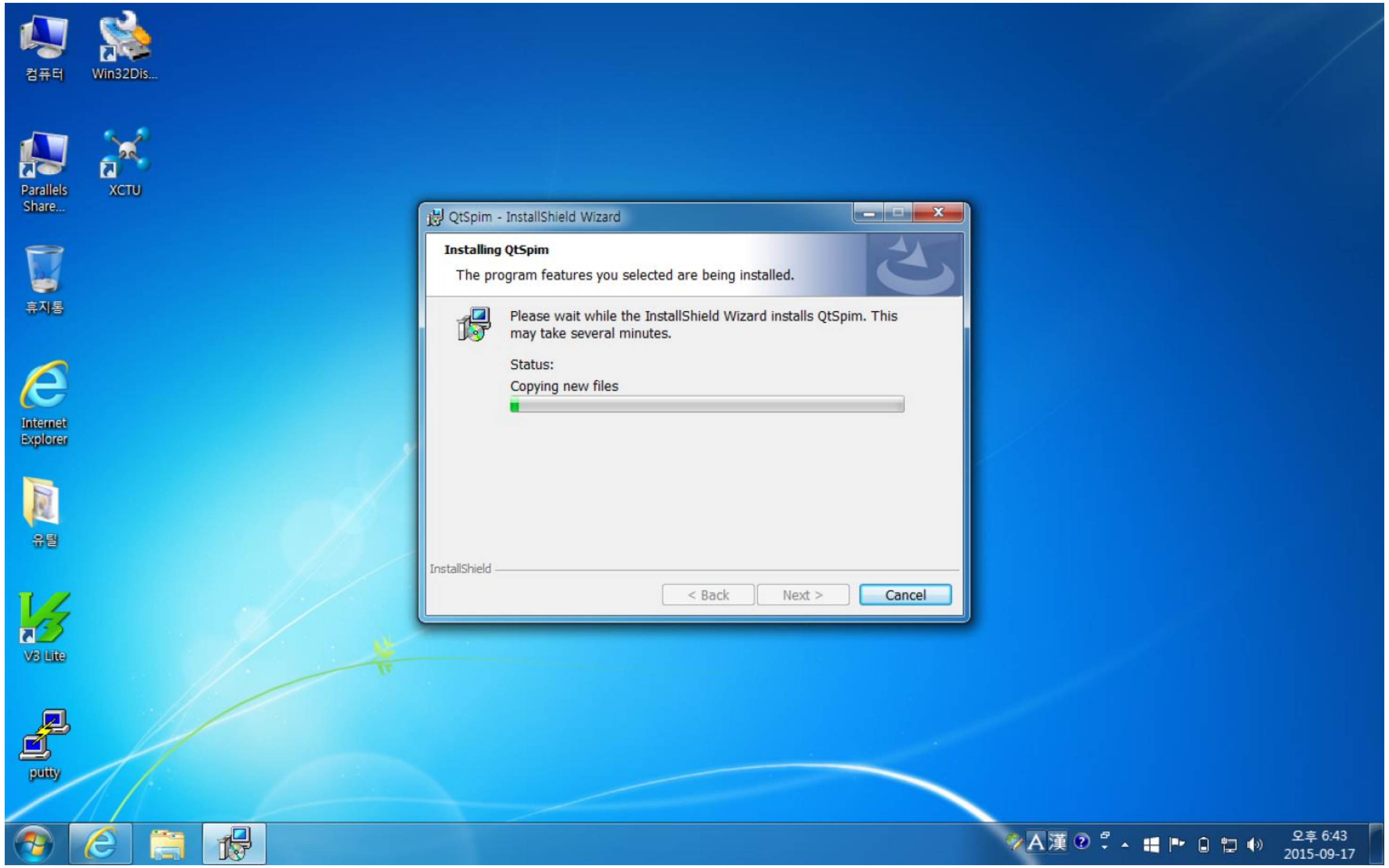
Current Settings:

Setup Type:
Typical

Destination Folder:
C:₩Program Files (x86)₩QtSpim₩

User Information:
Name: Registered User
Company: Microsoft Corporation

# Download and Install



QtSpim - InstallShield Wizard

**Installing QtSpim**

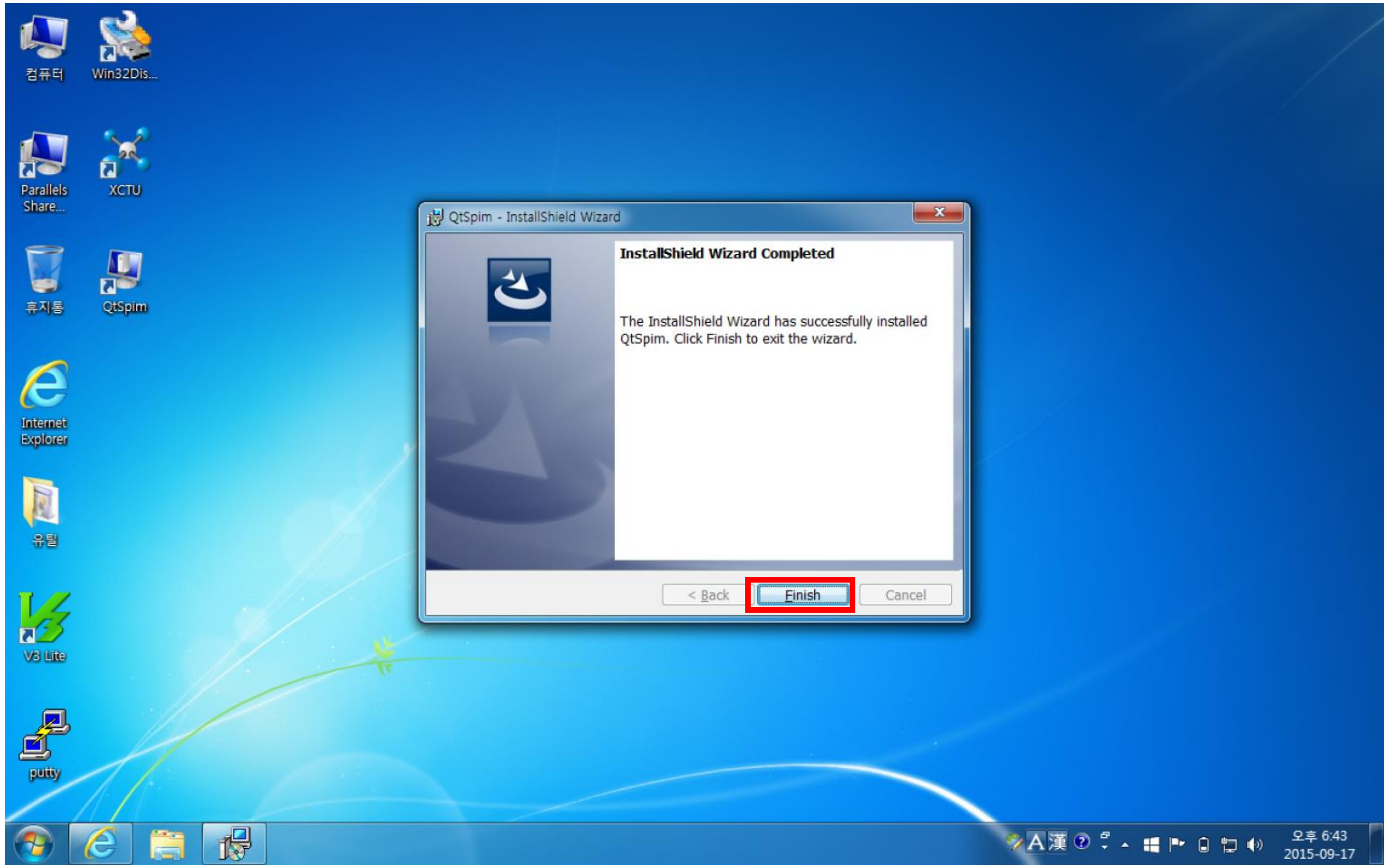
The program features you selected are being installed.

Please wait while the InstallShield Wizard installs QtSpim. This may take several minutes.

Status:

Copying new files

# Download and Install



QtSpim - InstallShield Wizard

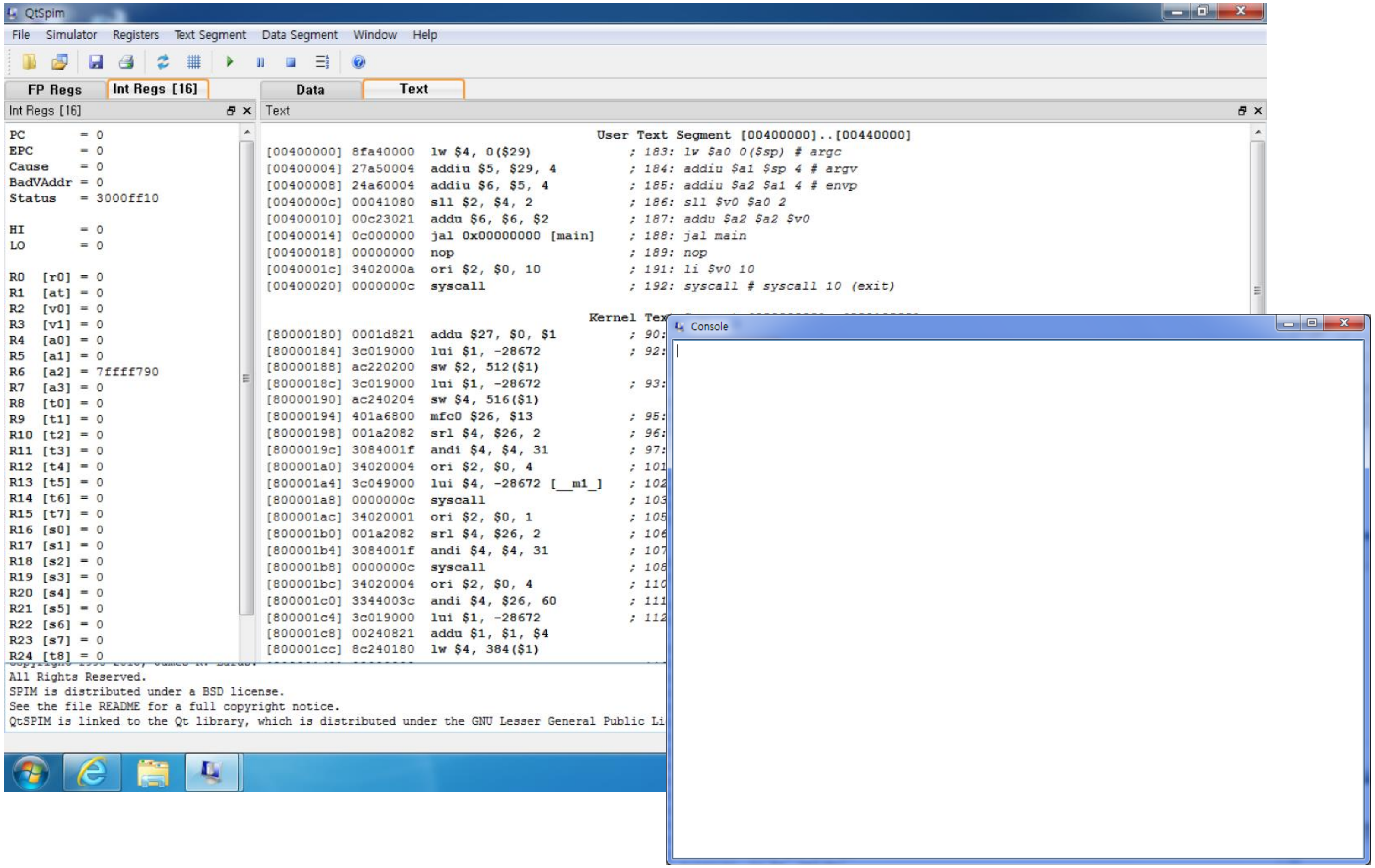
**InstallShield Wizard Completed**

The InstallShield Wizard has successfully installed QtSpim. Click Finish to exit the wizard.

# First screen of QtSpim

A window such as right picture and console window will be shown together.

You can check the result with console window.



```
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] = 7ffff790
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
```

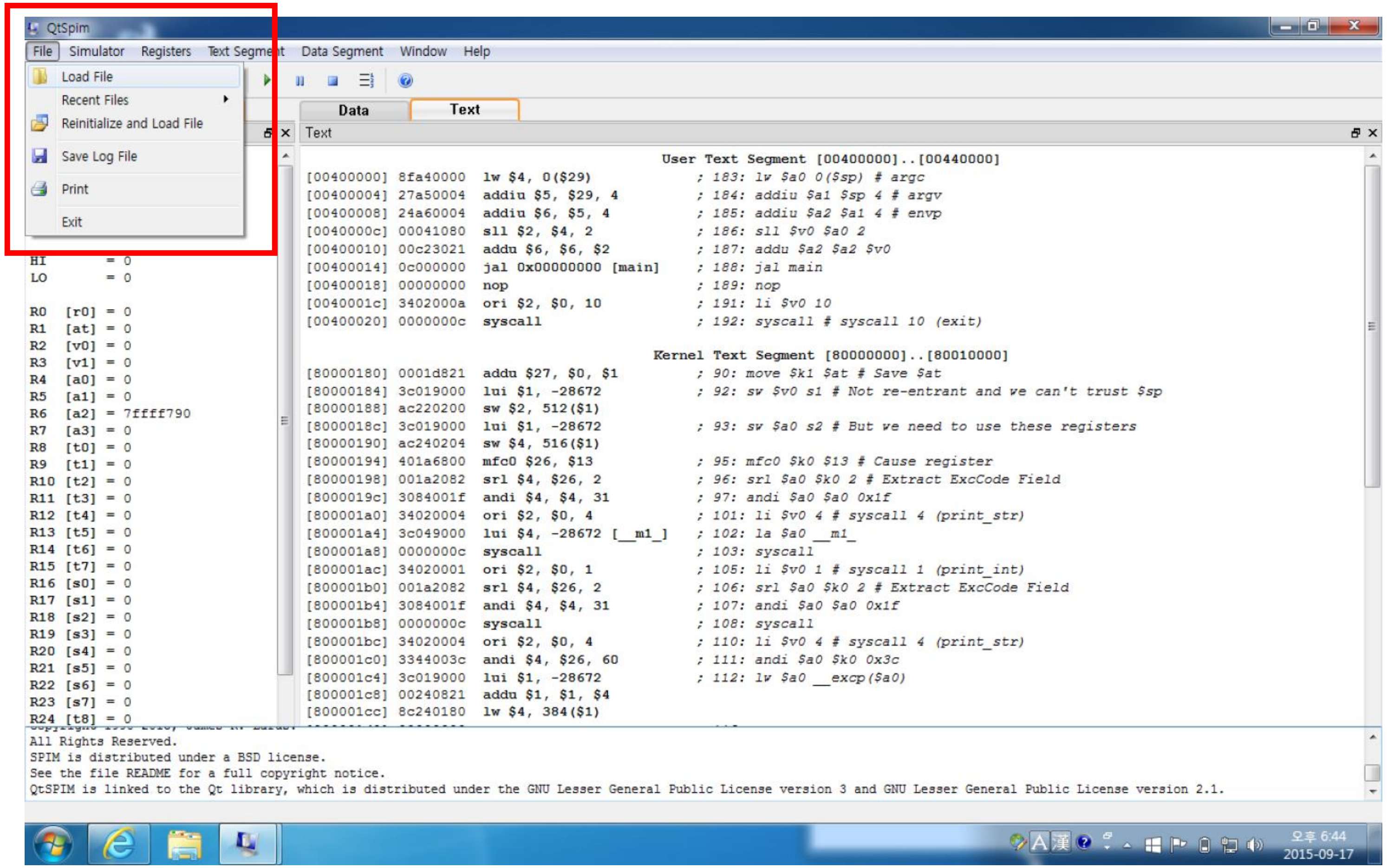
```
                                   User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)            ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4         ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4          ; 185: addiu $a2 $a1 4 # 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 Tex
[80000180] 0001d821  addu $27, $0, $1         ; 90:
[80000184] 3c019000  lui $1, -28672           ; 92:
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672           ; 93:
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13            ; 95:
[80000198] 001a2082  srl $4, $26, 2           ; 96:
[8000019c] 3084001f  andi $4, $4, 31          ; 97:
[800001a0] 34020004  ori $2, $0, 4            ; 101
[800001a4] 3c049000  lui $4, -28672 [__m1_]   ; 102
[800001a8] 0000000c  syscall                  ; 103
[800001ac] 34020001  ori $2, $0, 1            ; 105
[800001b0] 001a2082  srl $4, $26, 2           ; 106
[800001b4] 3084001f  andi $4, $4, 31          ; 107
[800001b8] 0000000c  syscall                  ; 108
[800001bc] 34020004  ori $2, $0, 4            ; 110
[800001c0] 3344003c  andi $4, $26, 60         ; 111
[800001c4] 3c019000  lui $1, -28672           ; 112
[800001c8] 00240821  addu $1, $1, $4
[800001cc] 8c240180  lw $4, 384($1)
```

```
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public Li
```

# Loading assembly language file

Go to ‘**File**’ ➔ ‘**Load File**’, then the load assembly language file.

(extension will be .a .s or .asm)



```
HI        = 0
LO        = 0

R0  [r0]  = 0
R1  [at]  = 0
R2  [v0]  = 0
R3  [v1]  = 0
R4  [a0]  = 0
R5  [a1]  = 0
R6  [a2]  = 7ffff790
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
```

```
                                   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)
```

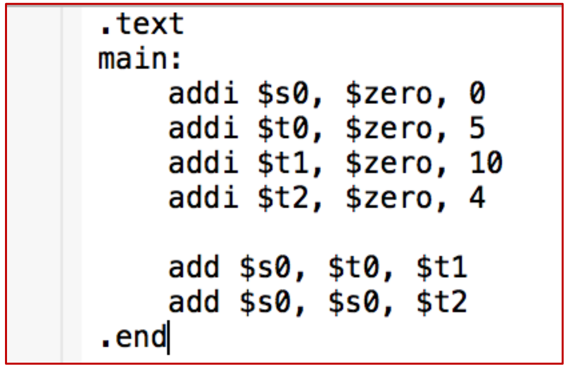
```
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and GNU Lesser General Public License version 2.1.
```

# User file displayed

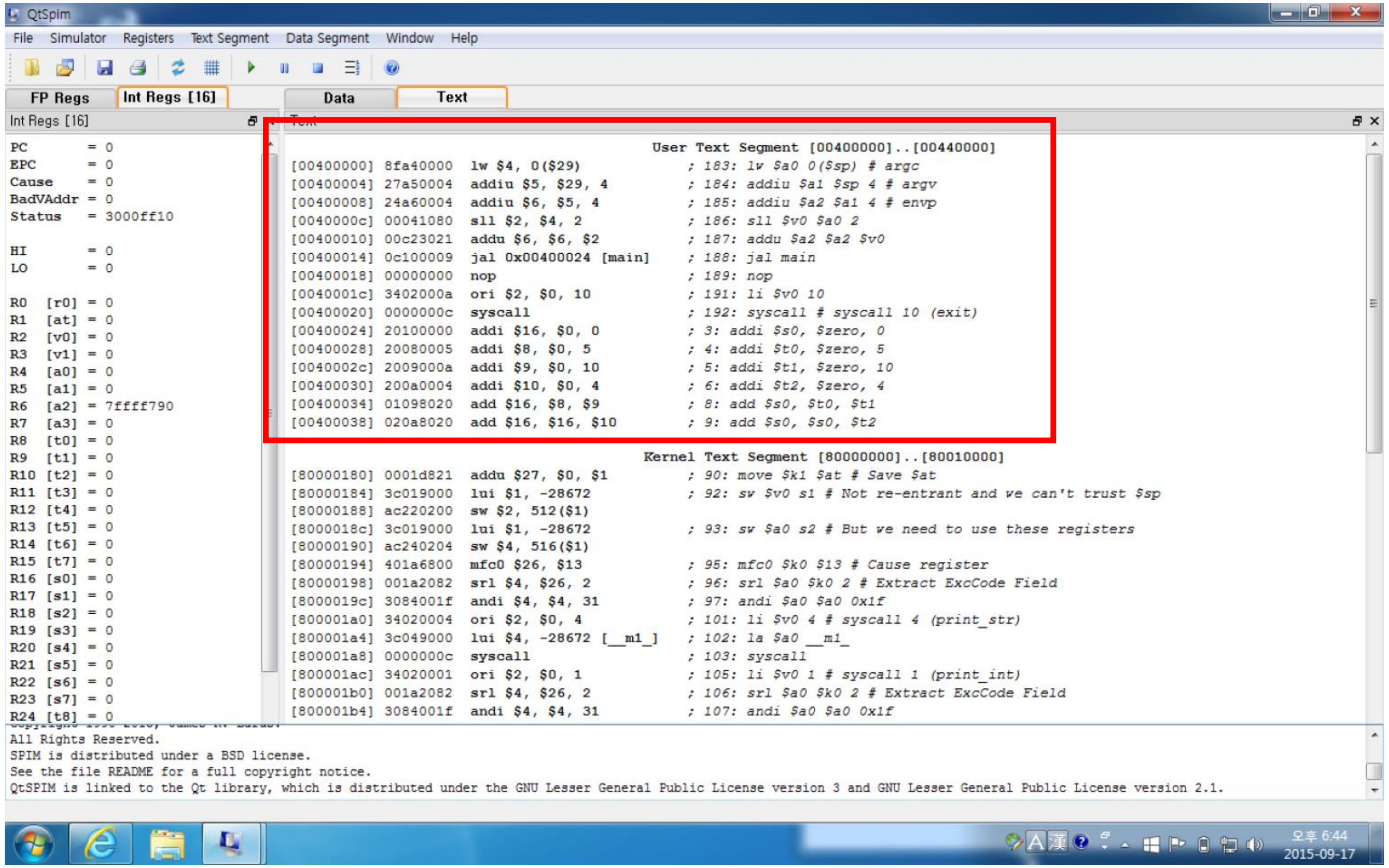
When file is open, code will appear on the red square.

Other part show detailed procedure of transformation of assembly code to machine language.

(The code below is source used in example.)



```
.text
main:
    addi $s0, $zero, 0
    addi $t0, $zero, 5
    addi $t1, $zero, 10
    addi $t2, $zero, 4

    add $s0, $t0, $t1
    add $s0, $s0, $t2
.end
```



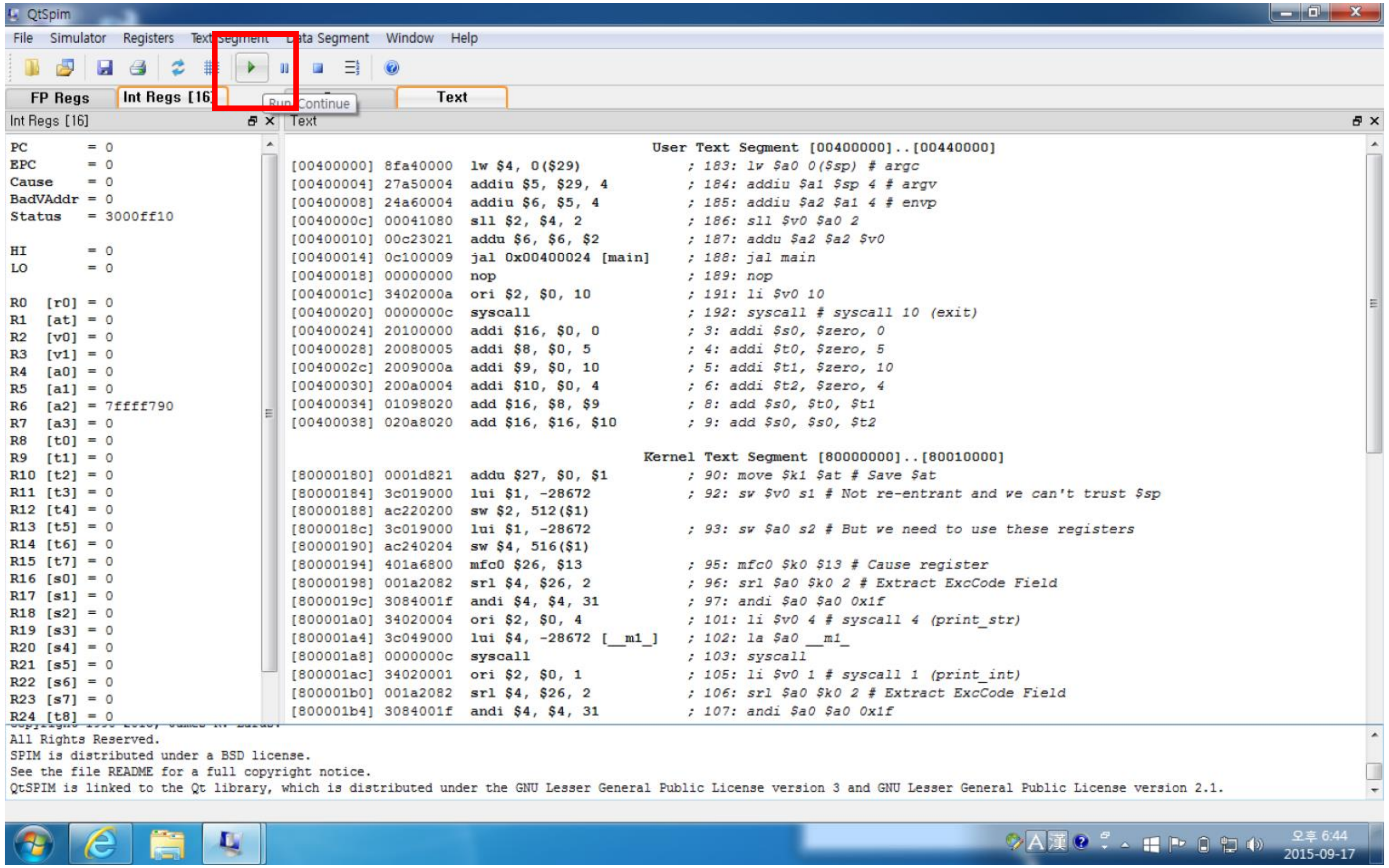
```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 0
R5  [a1] = 0
R6  [a2] = 7ffff790
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
```

```
                              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] 20100000  addi $16, $0, 0          ; 3: addi $s0, $zero, 0
[00400028] 20080005  addi $8, $0, 5           ; 4: addi $t0, $zero, 5
[0040002c] 2009000a  addi $9, $0, 10          ; 5: addi $t1, $zero, 10
[00400030] 200a0004  addi $10, $0, 4          ; 6: addi $t2, $zero, 4
[00400034] 01098020  add $16, $8, $9          ; 8: add $s0, $t0, $t1
[00400038] 020a8020  add $16, $16, $10        ; 9: add $s0, $s0, $t2

                              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
```

```
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and GNU Lesser General Public License version 2.1.
```

# Execution

When F5 or button in red box is pressed, assembly code will be executed.



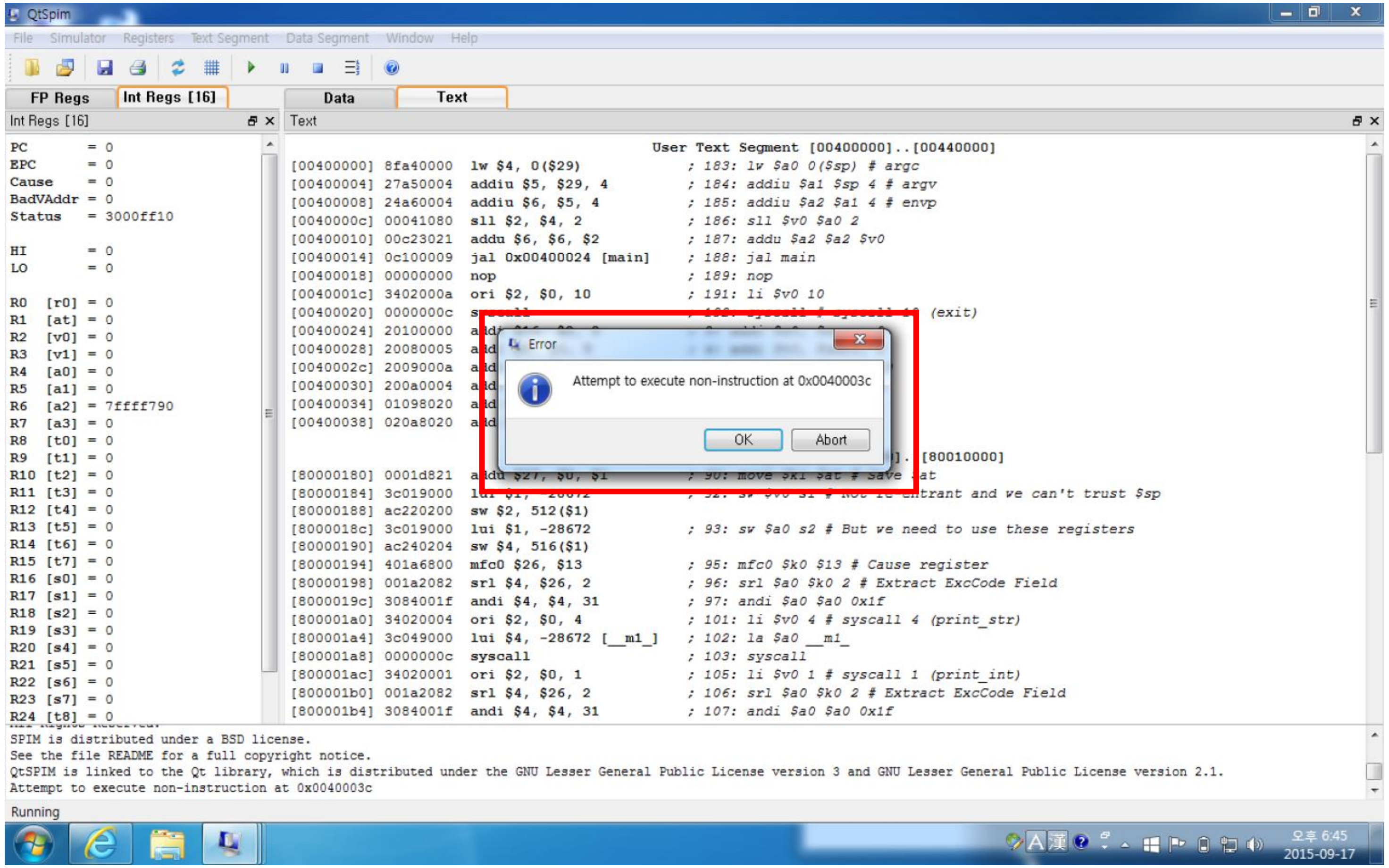
```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 0
R5  [a1] = 0
R6  [a2] = 7ffff790
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
```

```
                         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] 20100000  addi $16, $0, 0          ; 3: addi $s0, $zero, 0
[00400028] 20080005  addi $8, $0, 5           ; 4: addi $t0, $zero, 5
[0040002c] 2009000a  addi $9, $0, 10          ; 5: addi $t1, $zero, 10
[00400030] 200a0004  addi $10, $0, 4          ; 6: addi $t2, $zero, 4
[00400034] 01098020  add $16, $8, $9          ; 8: add $s0, $t0, $t1
[00400038] 020a8020  add $16, $16, $10        ; 9: add $s0, $s0, $t2

                        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
```

```
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and GNU Lesser General Public License version 2.1.
```

# Message

When assembly code is executed, message on right side will be popped up. Then just press 'OK' button.



```
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] = 7ffff790
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
```

```
                         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  s                                         (exit)
[00400024] 20100000  a
[00400028] 20080005  a
[0040002c] 2009000a  a
[00400030] 200a0004  a
[00400034] 01098020  a
[00400038] 020a8020  a

                                                  ]. [80010000]
[80000180] 0001d821  a                                  at
[80000184] 3c019000                                    ntrant 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
```

Attempt to execute non-instruction at 0x0040003c

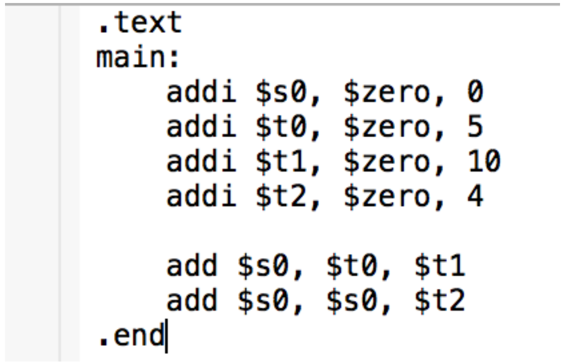
```
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and GNU Lesser General Public License version 2.1.
Attempt to execute non-instruction at 0x0040003c
```

# Checking register value

After execution of code, you can check each register value on left side of window.

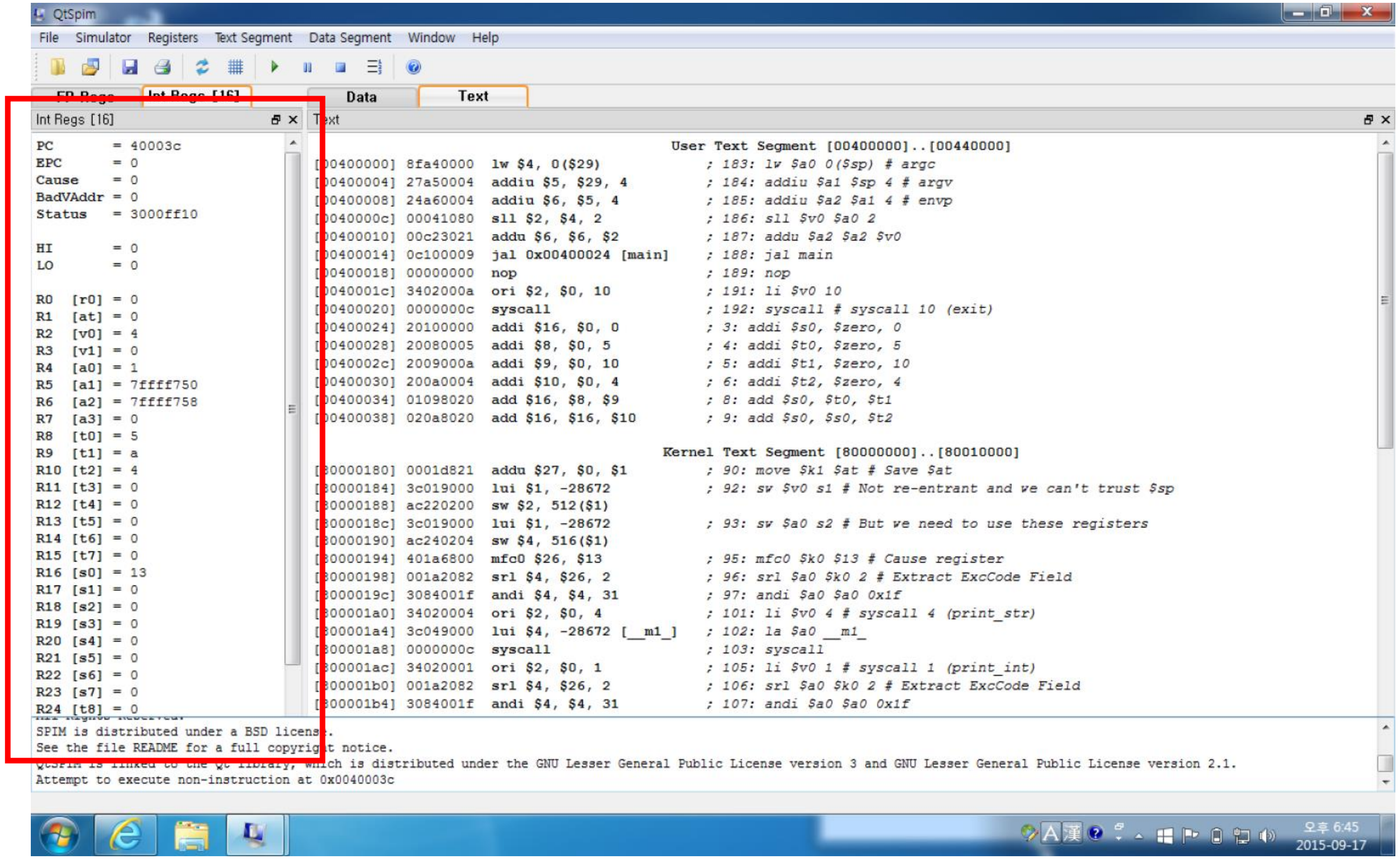
You can check the result of below source code.

$t0 = 5, $t1 = a, $t2 = 4, $s0 = 13



```
.text
main:
    addi $s0, $zero, 0
    addi $t0, $zero, 5
    addi $t1, $zero, 10
    addi $t2, $zero, 4

    add $s0, $t0, $t1
    add $s0, $s0, $t2
.end
```

Regiter value is hexadecimal number



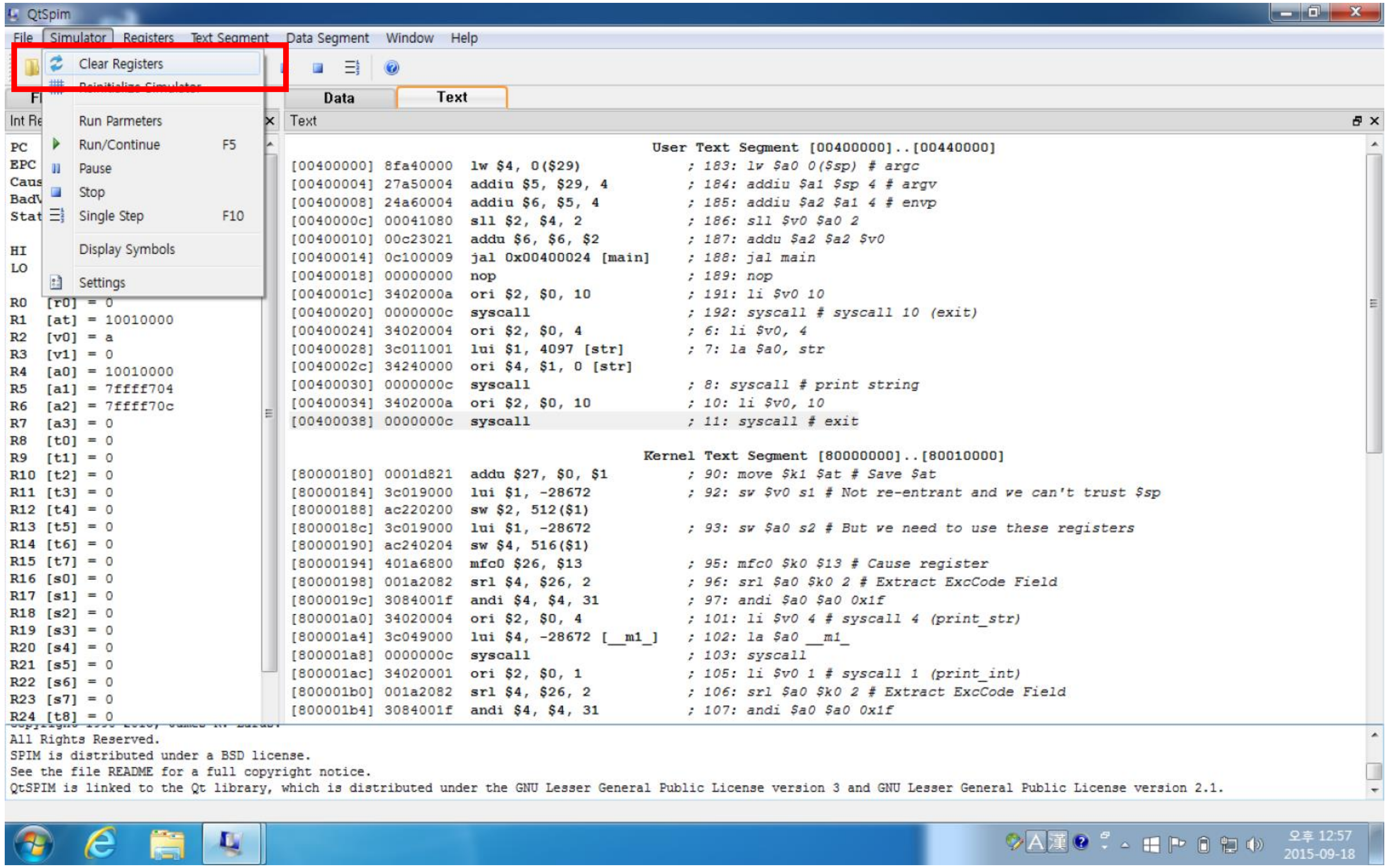
```
PC        = 40003c
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 3000ff10

HI        = 0
LO        = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 7ffff750
R6  [a2] = 7ffff758
R7  [a3] = 0
R8  [t0] = 5
R9  [t1] = a
R10 [t2] = 4
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 13
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
```

```
                         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] 20100000  addi $16, $0, 0          ; 3: addi $s0, $zero, 0
[00400028] 20080005  addi $8, $0, 5           ; 4: addi $t0, $zero, 5
[0040002c] 2009000a  addi $9, $0, 10          ; 5: addi $t1, $zero, 10
[00400030] 200a0004  addi $10, $0, 4          ; 6: addi $t2, $zero, 4
[00400034] 01098020  add $16, $8, $9          ; 8: add $s0, $t0, $t1
[00400038] 020a8020  add $16, $16, $10        ; 9: add $s0, $s0, $t2

                        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
```

```
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and GNU Lesser General Public License version 2.1.
Attempt to execute non-instruction at 0x0040003c
```

# Re-Execution

If you want to re-execute source code, go to ‘**Simulator**’ ➔ ‘**Clear Registers**’ and execute source file.



```
PC
EPC
Caus
BadV
Stat

HI
LO

R0   [r0] = 0
R1   [at] = 10010000
R2   [v0] = a
R3   [v1] = 0
R4   [a0] = 10010000
R5   [a1] = 7ffff704
R6   [a2] = 7ffff70c
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
```

```
                              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            ; 6: li $v0, 4
[00400028] 3c011001  lui $1, 4097 [str]       ; 7: la $a0, str
[0040002c] 34240000  ori $4, $1, 0 [str]
[00400030] 0000000c  syscall                  ; 8: syscall # print string
[00400034] 3402000a  ori $2, $0, 10           ; 10: li $v0, 10
[00400038] 0000000c  syscall                  ; 11: syscall # 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
```

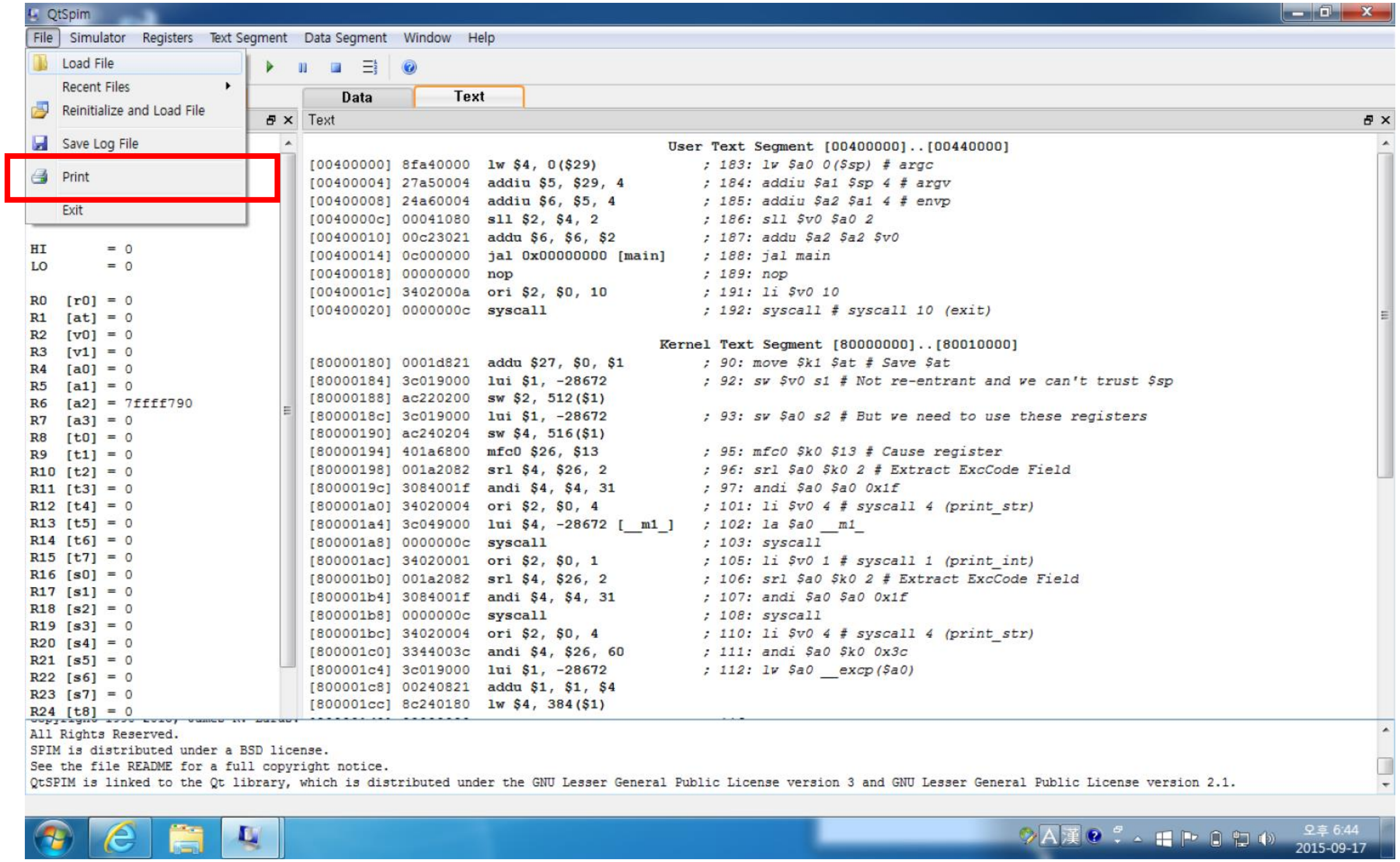
```
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and GNU Lesser General Public License version 2.1.
```

# Loading another file

If you want to load another assembly file you can do it with

‘File’ ➔’Reinitialize and Load File’

Even when message below shows up, press ‘File’ ➔’Reinitialize and Load File’



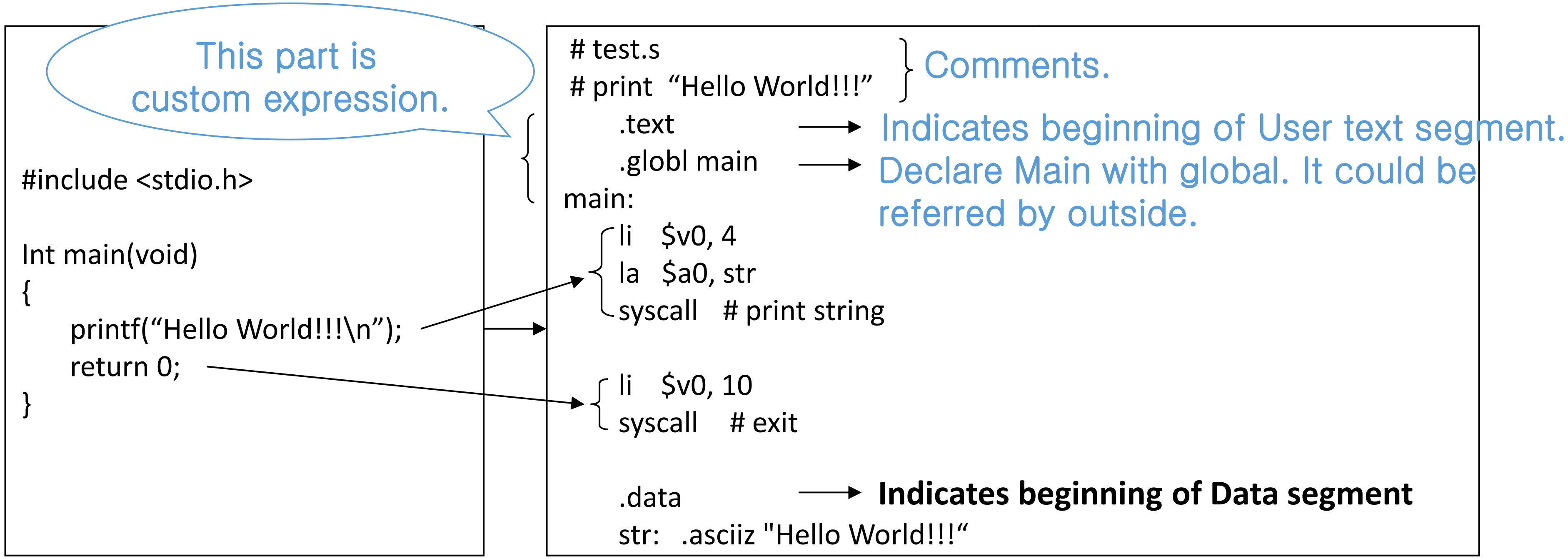
```
HI      = 0
LO      = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 0
R5  [a1] = 0
R6  [a2] = 7ffff790
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
```

```
                        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)
```

```
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and GNU Lesser General Public License version 2.1.
```

```
spim: (parser) Label is defined for the second time on line 9 of file
        main:
          ^
```

# Example



This part is custom expression.

```
#include <stdio.h>

Int main(void)
{
      printf(“Hello World!!!\n”);
      return 0;
}
```

```
# test.s
# print  “Hello World!!!”
      .text
      .globl main
main:
      li   $v0, 4
      la   $a0, str
      syscall   # print string

      li   $v0, 10
      syscall    # exit

      .data
      str:  .asciiz "Hello World!!!“
```

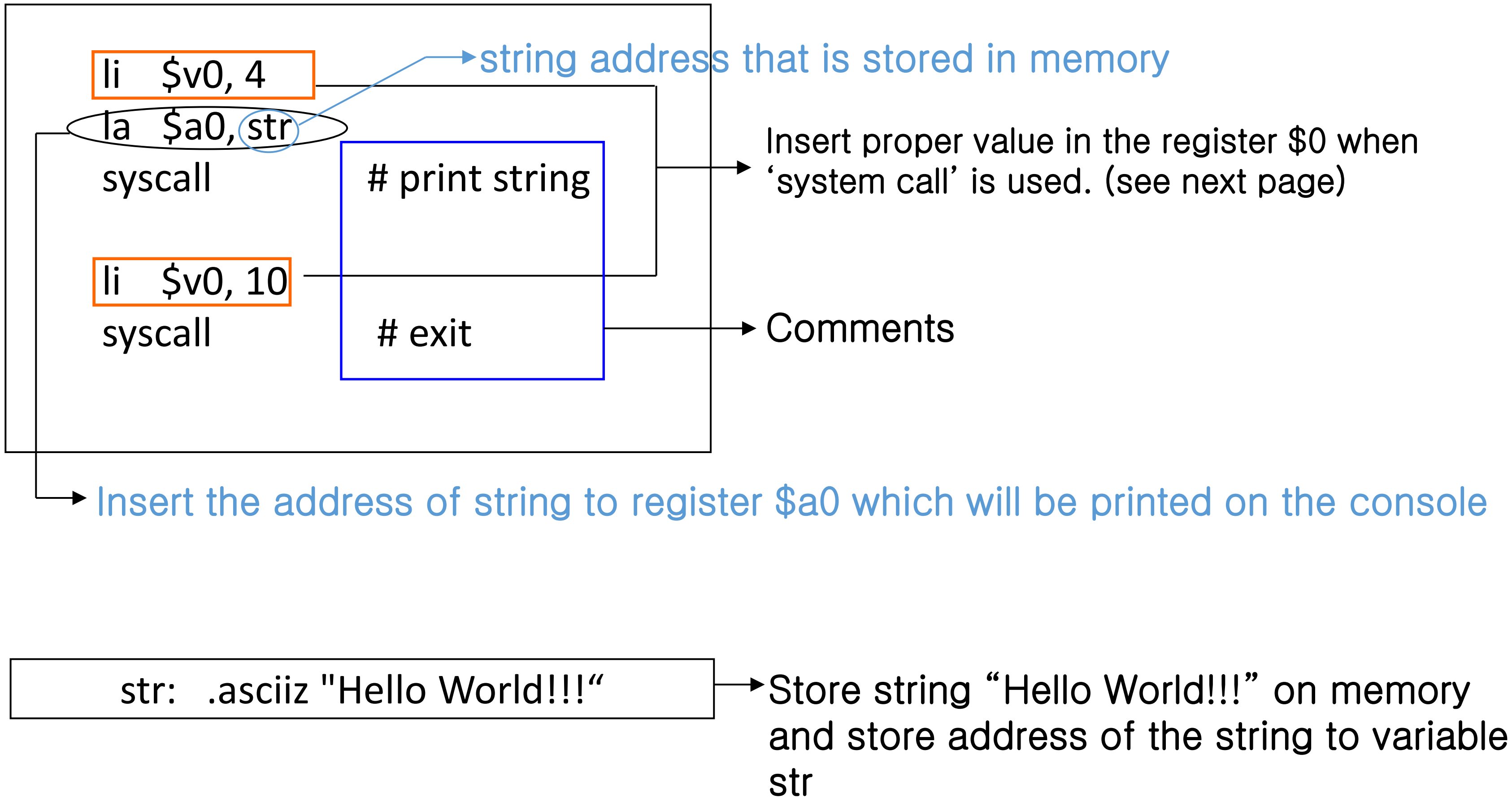
Comments.

Indicates beginning of User text segment.

Declare Main with global. It could be referred by outside.

**Indicates beginning of Data segment**

# System call



```
li   $v0, 4
la   $a0, str
syscall          # print string

li   $v0, 10
syscall          # exit
```

string address that is stored in memory

Insert proper value in the register $0 when
‘system call’ is used. (see next page)

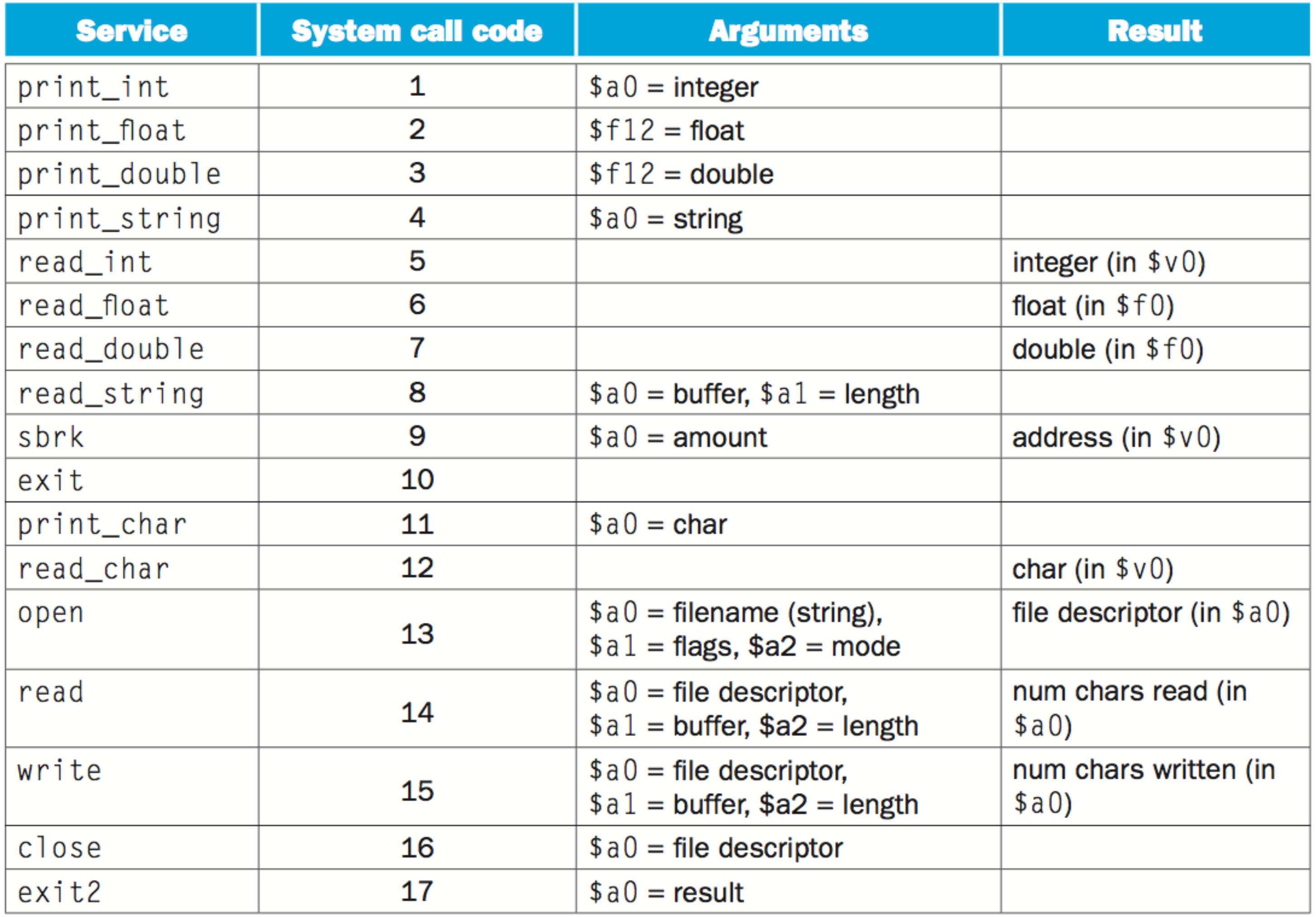
Comments

Insert the address of string to register $a0 which will be printed on the console

```
str:   .asciiz "Hello World!!!“
```

Store string “Hello World!!!” on memory and store address of the string to variable str

# System Call (System services)

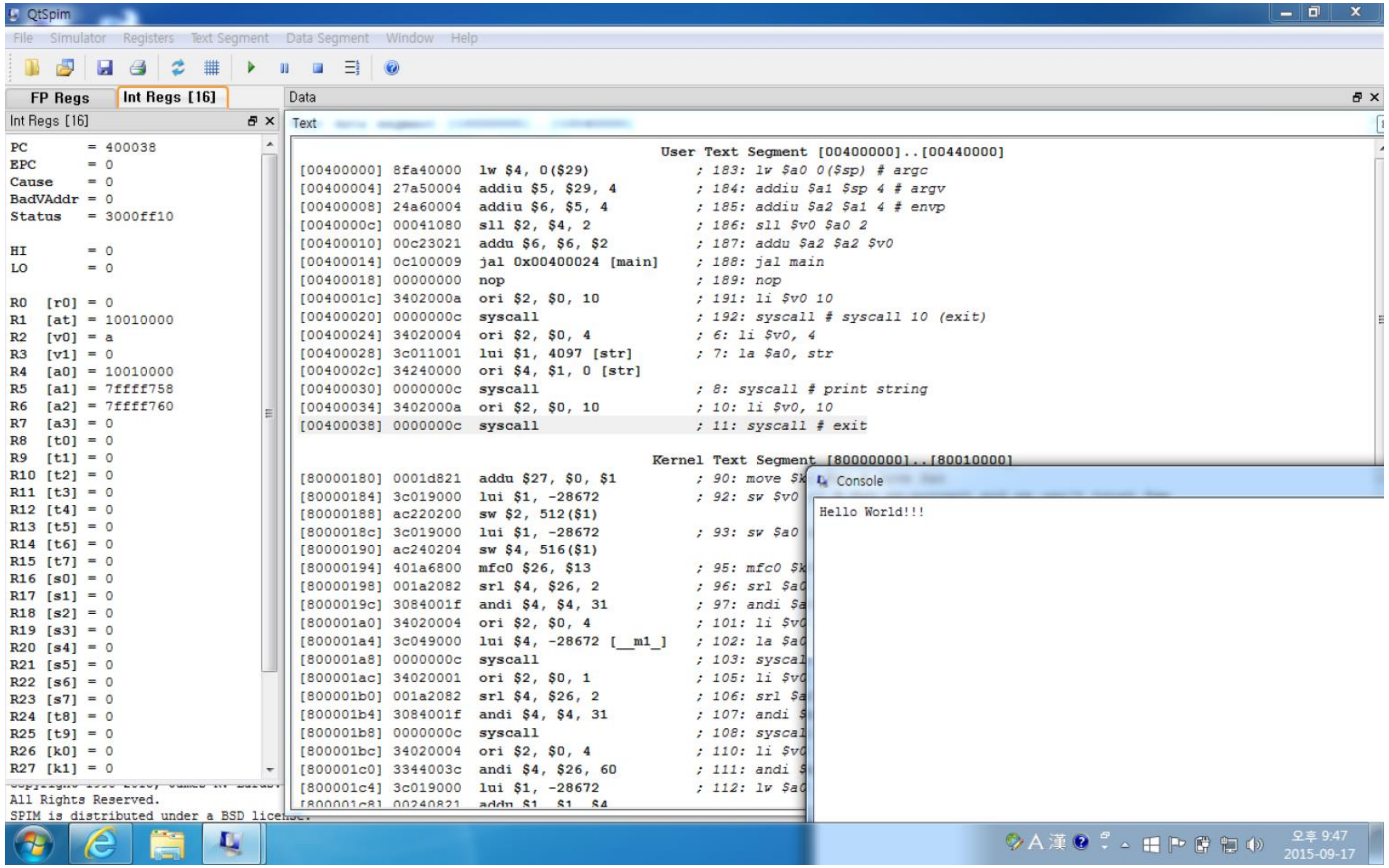


| Service | System call code | Arguments | Result |
|---|---|---|---|
| print_int | 1 | $a0 = integer | |
| print_float | 2 | $f12 = float | |
| print_double | 3 | $f12 = double | |
| print_string | 4 | $a0 = string | |
| read_int | 5 | | integer (in $v0) |
| read_float | 6 | | float (in $f0) |
| read_double | 7 | | double (in $f0) |
| read_string | 8 | $a0 = buffer, $a1 = length | |
| sbrk | 9 | $a0 = amount | address (in $v0) |
| exit | 10 | | |
| print_char | 11 | $a0 = char | |
| read_char | 12 | | char (in $v0) |
| open | 13 | $a0 = filename (string), $a1 = flags, $a2 = mode | file descriptor (in $a0) |
| read | 14 | $a0 = file descriptor, $a1 = buffer, $a2 = length | num chars read (in $a0) |
| write | 15 | $a0 = file descriptor, $a1 = buffer, $a2 = length | num chars written (in $a0) |
| close | 16 | $a0 = file descriptor | |
| exit2 | 17 | $a0 = result | |

**FIGURE B.9.1** **System services.**

# Previous Example

```
# test.s
# print "Hello World!!!"
    .text
    .globl main
main:
    li $v0, 4
    la $a0, str
    syscall      # print string

    li $v0, 10
    syscall      # exit

    .data
    str: .asciiz "Hello World!!!"
```


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
Console
오후 9:47
2015-09-17

Int Regs [16]

```
PC       = 400038
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] = 10010000
R5  [a1] = 7ffff758
R6  [a2] = 7ffff760
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
```

```
                                        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             ; 6: li $v0, 4
[00400028] 3c011001  lui $1, 4097 [str]        ; 7: la $a0, str
[0040002c] 34240000  ori $4, $1, 0 [str]
[00400030] 0000000c  syscall                   ; 8: syscall # print string
[00400034] 3402000a  ori $2, $0, 10            ; 10: li $v0, 10
[00400038] 0000000c  syscall                   ; 11: syscall # exit

                                        Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1          ; 90: move $k
[80000184] 3c019000  lui $1, -28672            ; 92: sw $v0
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672            ; 93: sw $a0
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13             ; 95: mfc0 $k
[80000198] 001a2082  srl $4, $26, 2            ; 96: srl $a0
[8000019c] 3084001f  andi $4, $4, 31           ; 97: andi $a
[800001a0] 34020004  ori $2, $0, 4             ; 101: li $v0
[800001a4] 3c049000  lui $4, -28672 [__m1_]    ; 102: la $a0
[800001a8] 0000000c  syscall                   ; 103: syscal
[800001ac] 34020001  ori $2, $0, 1             ; 105: li $v0
[800001b0] 001a2082  srl $4, $26, 2            ; 106: srl $a
[800001b4] 3084001f  andi $4, $4, 31           ; 107: andi $
[800001b8] 0000000c  syscall                   ; 108: syscal
[800001bc] 34020004  ori $2, $0, 4             ; 110: li $v0
[800001c0] 3344003c  andi $4, $26, 60          ; 111: andi $
[800001c4] 3c019000  lui $1, -28672            ; 112: lw $a0
[800001c8] 00240821  addu $1, $1, $4
```

```
All Rights Reserved.
SPIM is distributed under a BSD license.
```

Console

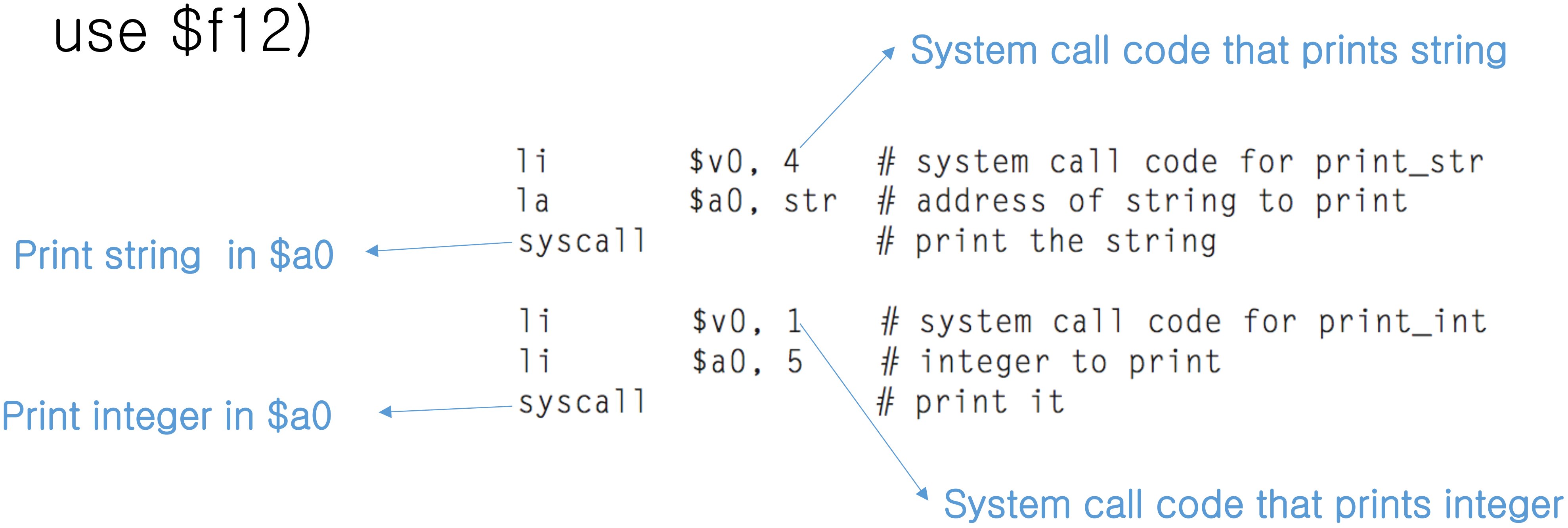
```
Hello World!!!
```

# Pseudo Assembly Instruction

- **move $rt, $rs**: copy contents of $rs to $rt
- **li $rs, immed**: Load immediate value(immed) to $rs
- **la $rs, addr**: Load address(addr) to $rs
- **lw $rt, big($rs)**: Load the value of $rs which the offset is added and store this value to $rt

# System Call (syscall)

- Load 'system call code' that you want to use on the register $v0
- Use $a0 ~ $a3 to load argument (If value is floating-point, use $f12)



System call code that prints string

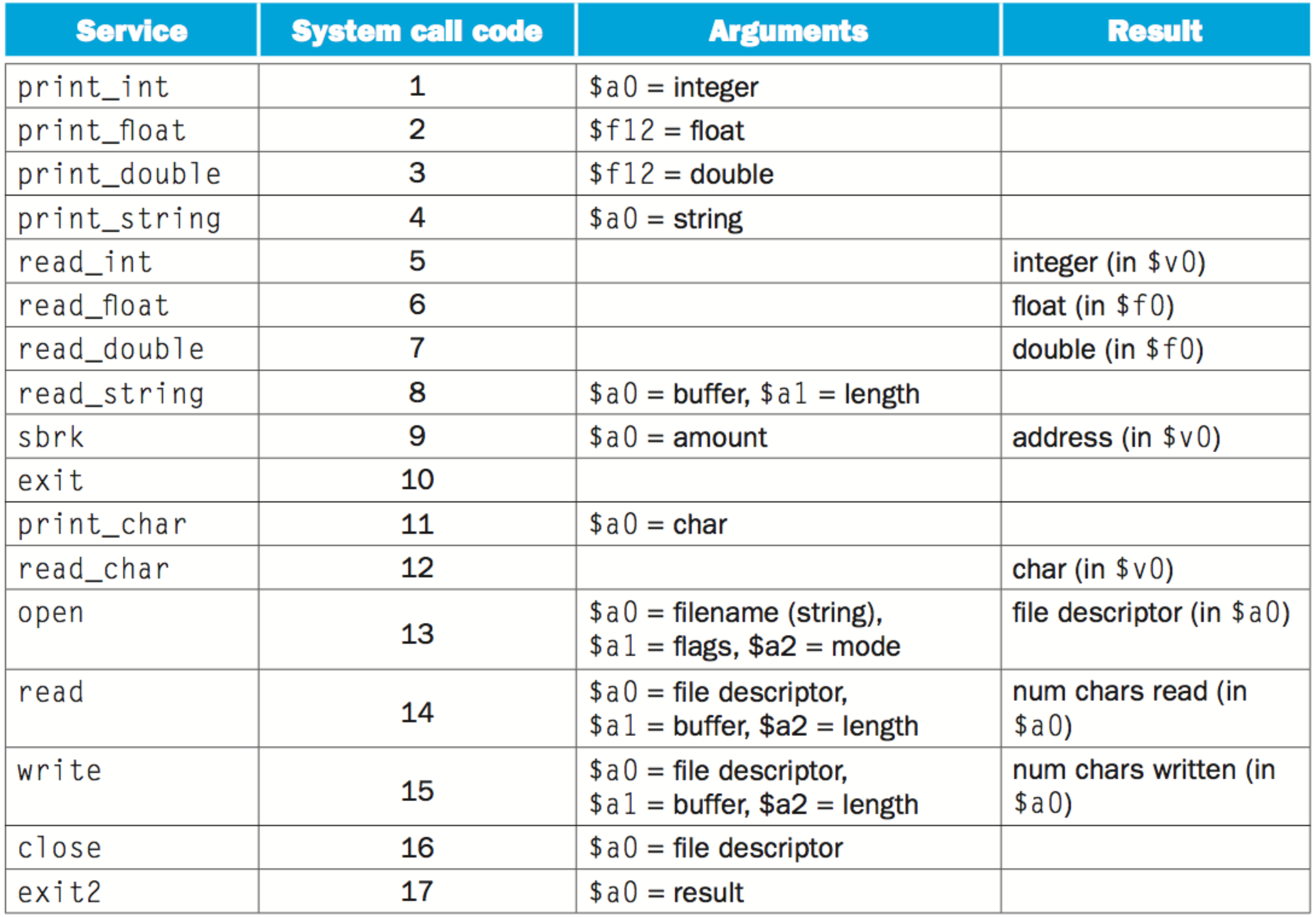
```
li       $v0, 4    # system call code for print_str
la       $a0, str  # address of string to print
syscall            # print the string

li       $v0, 1    # system call code for print_int
li       $a0, 5    # integer to print
syscall            # print it
```

Print string  in $a0

Print integer in $a0

System call code that prints integer

# System Call (syscall)

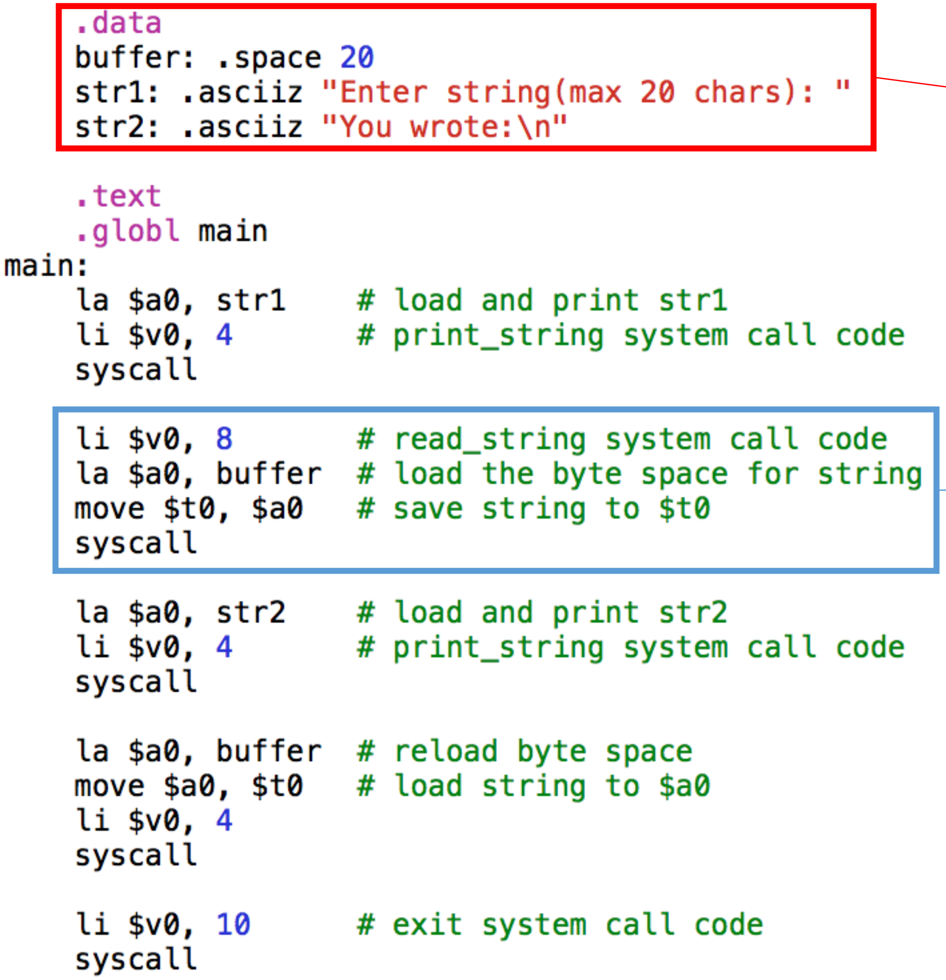


| Service | System call code | Arguments | Result |
|---|---|---|---|
| print_int | 1 | $a0 = integer | |
| print_float | 2 | $f12 = float | |
| print_double | 3 | $f12 = double | |
| print_string | 4 | $a0 = string | |
| read_int | 5 | | integer (in $v0) |
| read_float | 6 | | float (in $f0) |
| read_double | 7 | | double (in $f0) |
| read_string | 8 | $a0 = buffer, $a1 = length | |
| sbrk | 9 | $a0 = amount | address (in $v0) |
| exit | 10 | | |
| print_char | 11 | $a0 = char | |
| read_char | 12 | | char (in $v0) |
| open | 13 | $a0 = filename (string), $a1 = flags, $a2 = mode | file descriptor (in $a0) |
| read | 14 | $a0 = file descriptor, $a1 = buffer, $a2 = length | num chars read (in $a0) |
| write | 15 | $a0 = file descriptor, $a1 = buffer, $a2 = length | num chars written (in $a0) |
| close | 16 | $a0 = file descriptor | |
| exit2 | 17 | $a0 = result | |

**FIGURE B.9.1 System services.**

# System Call (syscall)

This program receives string from user and prints this string.

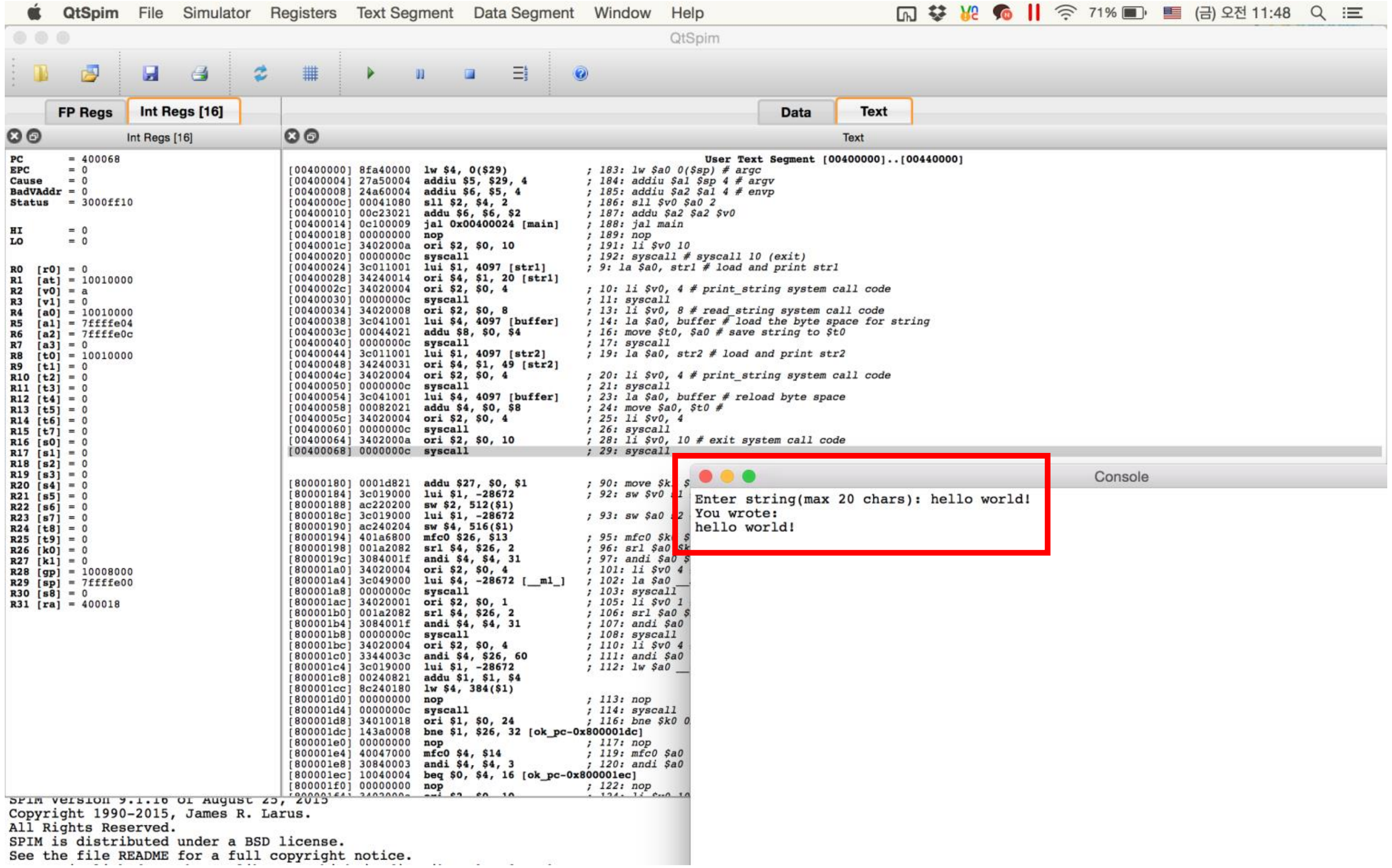


```
  .data
  buffer: .space 20
  str1: .asciiz "Enter string(max 20 chars): "
  str2: .asciiz "You wrote:\n"

  .text
  .globl main
main:
  la $a0, str1    # load and print str1
  li $v0, 4       # print_string system call code
  syscall

  li $v0, 8       # read_string system call code
  la $a0, buffer  # load the byte space for string
  move $t0, $a0   # save string to $t0
  syscall

  la $a0, str2    # load and print str2
  li $v0, 4       # print_string system call code
  syscall

  la $a0, buffer  # reload byte space
  move $a0, $t0   # load string to $a0
  li $v0, 4
  syscall

  li $v0, 10      # exit system call code
  syscall
```

Declare the variables which will be used.

Code that receive string.

# System Call (syscall)



```
PC       = 400068
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] = 10010000
R5  [a1] = 7ffffe04
R6  [a2] = 7ffffe0c
R7  [a3] = 0
R8  [t0] = 10010000
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
R28 [gp] = 10008000
R29 [sp] = 7ffffe00
R30 [s8] = 0
R31 [ra] = 400018
```

```
                                                 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 [str1]        ; 9: la $a0, str1 # load and print str1
[00400028] 34240014  ori $4, $1, 20 [str1]
[0040002c] 34020004  ori $2, $0, 4              ; 10: li $v0, 4 # print_string system call code
[00400030] 0000000c  syscall                    ; 11: syscall
[00400034] 34020008  ori $2, $0, 8              ; 13: li $v0, 8 # read_string system call code
[00400038] 3c041001  lui $4, 4097 [buffer]      ; 14: la $a0, buffer # load the byte space for string
[0040003c] 00044021  addu $8, $0, $4            ; 16: move $t0, $a0 # save string to $t0
[00400040] 0000000c  syscall                    ; 17: syscall
[00400044] 3c011001  lui $1, 4097 [str2]        ; 19: la $a0, str2 # load and print str2
[00400048] 34240031  ori $4, $1, 49 [str2]
[0040004c] 34020004  ori $2, $0, 4              ; 20: li $v0, 4 # print_string system call code
[00400050] 0000000c  syscall                    ; 21: syscall
[00400054] 3c041001  lui $4, 4097 [buffer]      ; 23: la $a0, buffer # reload byte space
[00400058] 00082021  addu $4, $0, $8            ; 24: move $a0, $t0 #
[0040005c] 34020004  ori $2, $0, 4              ; 25: li $v0, 4
[00400060] 0000000c  syscall                    ; 26: syscall
[00400064] 3402000a  ori $2, $0, 10             ; 28: li $v0, 10 # exit system call code
[00400068] 0000000c  syscall                    ; 29: syscall


[80000180] 0001d821  addu $27, $0, $1           ; 90: move $k
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672             ; 93: sw $a0
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13              ; 95: mfc0 $k
[80000198] 001a2082  srl $4, $26, 2             ; 96: srl $a0
[8000019c] 3084001f  andi $4, $4, 31            ; 97: andi $a0
[800001a0] 34020004  ori $2, $0, 4              ; 101: li $v0 4
[800001a4] 3c049000  lui $4, -28672 [__m1_]     ; 102: la $a0 __
[800001a8] 0000000c  syscall                    ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1              ; 105: li $v0 1
[800001b0] 001a2082  srl $4, $26, 2             ; 106: srl $a0 $
[800001b4] 3084001f  andi $4, $4, 31            ; 107: andi $a0
[800001b8] 0000000c  syscall                    ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4              ; 110: li $v0 4
[800001c0] 3344003c  andi $4, $26, 60           ; 111: andi $a0
[800001c4] 3c019000  lui $1, -28672             ; 112: lw $a0 __
[800001c8] 00240821  addu $1, $1, $4
[800001cc] 8c240180  lw $4, 384($1)
[800001d0] 00000000  nop                        ; 113: nop
[800001d4] 0000000c  syscall                    ; 114: syscall
[800001d8] 34010018  ori $1, $0, 24             ; 116: bne $k0 0
[800001dc] 143a0008  bne $1, $26, 32 [ok_pc-0x800001dc]
[800001e0] 00000000  nop                        ; 117: nop
[800001e4] 40047000  mfc0 $4, $14               ; 119: mfc0 $a0
[800001e8] 30840003  andi $4, $4, 3             ; 120: andi $a0
[800001ec] 10040004  beq $0, $4, 16 [ok_pc-0x800001ec]
[800001f0] 00000000  nop                        ; 122: nop
```

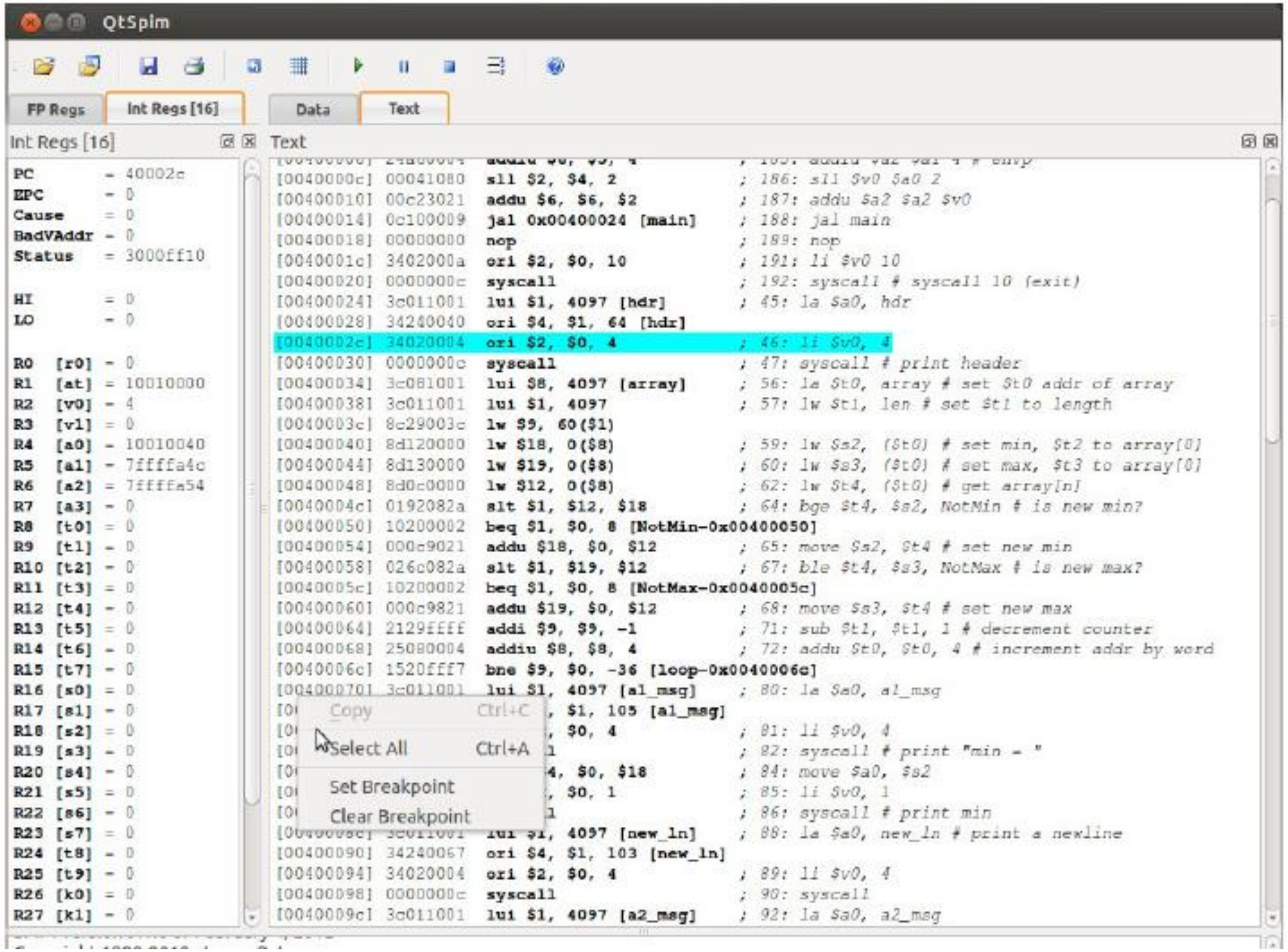
```
Enter string(max 20 chars): hello world!
You wrote:
hello world!
```

```
SPIM Version 9.1.16 of August 25, 2015
Copyright 1990-2015, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
```

# Breakpoint

- When an appropriate instruction is determined, move the cursor to the instruction address and right-click. The right-click will display the breakpoint menu as shown in the image below.

# Breakpoint



Int Regs [16]

```
PC        = 40002c
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 3000ff10

HI        = 0
LO        = 0

R0  [r0] = 0
R1  [at] = 10010000
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 10010040
R5  [a1] = 7ffffa4c
R6  [a2] = 7ffffa54
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

```
[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 [hdr]        ; 45: la $a0, hdr
[00400028] 34240040  ori $4, $1, 64 [hdr]
[0040002c] 34020004  ori $2, $0, 4             ; 46: li $v0, 4
[00400030] 0000000c  syscall                   ; 47: syscall # print header
[00400034] 3c081001  lui $8, 4097 [array]      ; 56: la $t0, array # set $t0 addr of array
[00400038] 3c011001  lui $1, 4097              ; 57: lw $t1, len # set $t1 to length
[0040003c] 8c29003c  lw $9, 60($1)
[00400040] 8d120000  lw $18, 0($8)             ; 59: lw $s2, ($t0) # set min, $t2 to array[0]
[00400044] 8d130000  lw $19, 0($8)             ; 60: lw $s3, ($t0) # set max, $t3 to array[0]
[00400048] 8d0c0000  lw $12, 0($8)             ; 62: lw $t4, ($t0) # get array[n]
[0040004c] 0192082a  slt $1, $12, $18          ; 64: bge $t4, $s2, NotMin # is new min?
[00400050] 10200002  beq $1, $0, 8 [NotMin-0x00400050]
[00400054] 000c9021  addu $18, $0, $12         ; 65: move $s2, $t4 # set new min
[00400058] 026c082a  slt $1, $19, $12          ; 67: ble $t4, $s3, NotMax # is new max?
[0040005c] 10200002  beq $1, $0, 8 [NotMax-0x0040005c]
[00400060] 000c9821  addu $19, $0, $12         ; 68: move $s3, $t4 # set new max
[00400064] 2129ffff  addi $9, $9, -1           ; 71: sub $t1, $t1, 1 # decrement counter
[00400068] 25080004  addiu $8, $8, 4           ; 72: addu $t0, $t0, 4 # increment addr by word
[0040006c] 1520fff7  bne $9, $0, -36 [loop-0x0040006c]
[00400070] 3c011001  lui $1, 4097 [a1_msg]     ; 80: la $a0, a1_msg
[0(                   , $1, 105 [a1_msg]
[0(                   , $0, 4                  ; 81: li $v0, 4
[0(                   1                        ; 82: syscall # print "min = "
[0(                   4, $0, $18               ; 84: move $a0, $s2
[0(                   , $0, 1                  ; 85: li $v0, 1
[0(                   1                        ; 86: syscall # print min
                     lui $1, 4097 [new_ln]     ; 88: la $a0, new_ln # print a newline
[00400090] 34240067  ori $4, $1, 103 [new_ln]
[00400094] 34020004  ori $2, $0, 4             ; 89: li $v0, 4
[00400098] 0000000c  syscall                   ; 90: syscall
[0040009c] 3c011001  lui $1, 4097 [a2_msg]     ; 92: la $a0, a2_msg
```