Spim Download

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

Latest version of Spim is QtSpim.

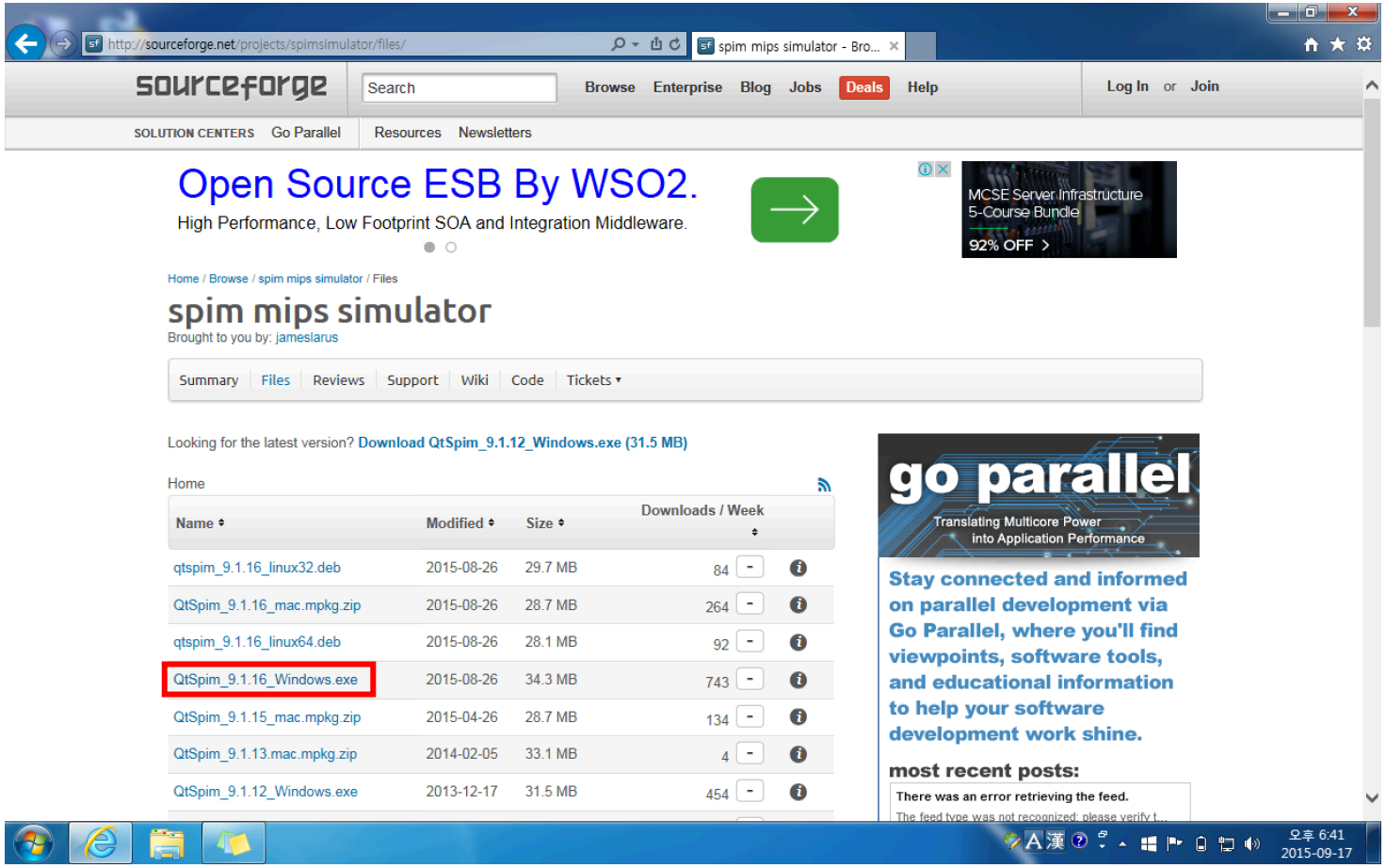
QtSpim is available on Microsoft Windows, Mac OS X and Linux environment.

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

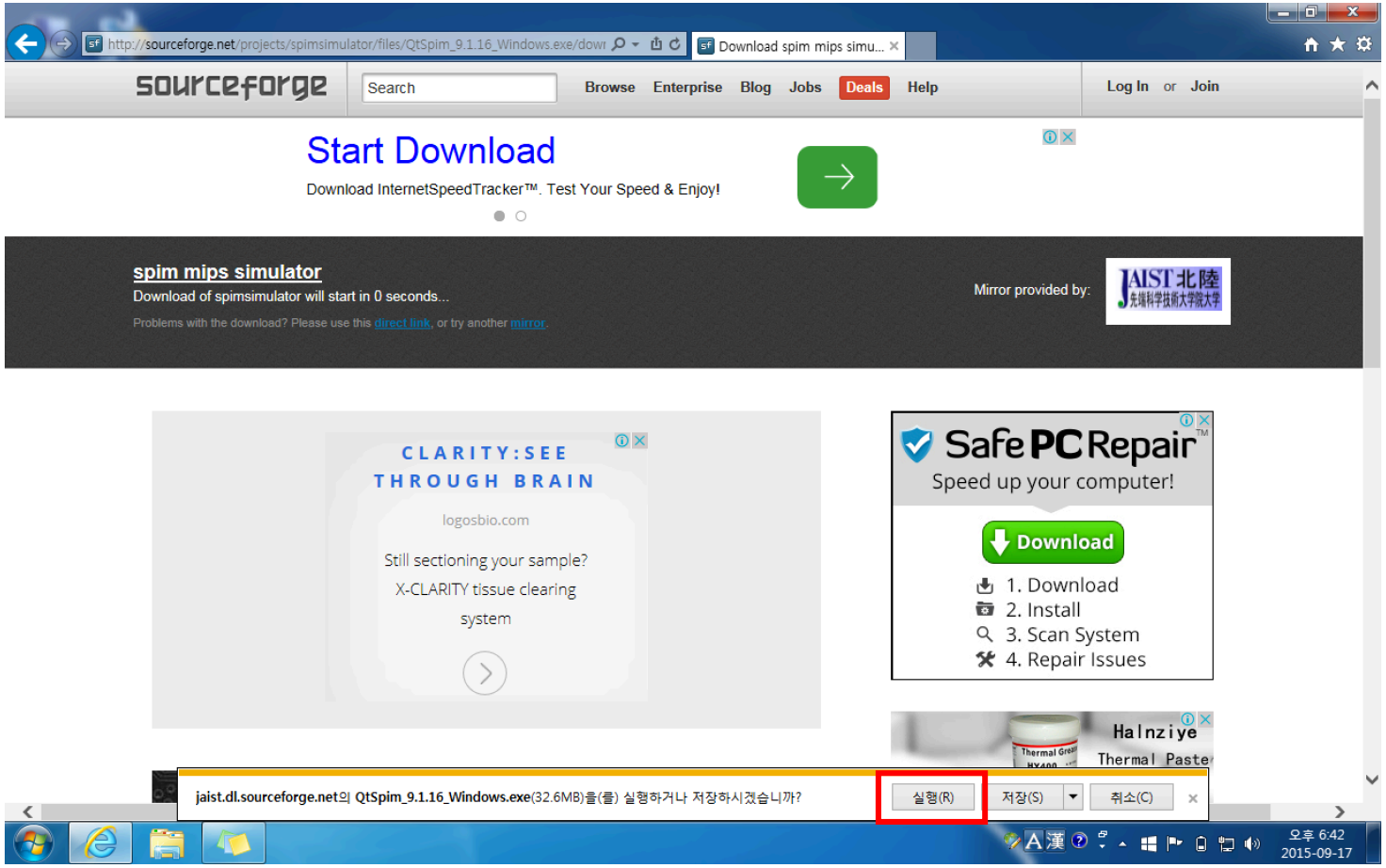
Go to url above and download appropriate file for your OS.

- 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

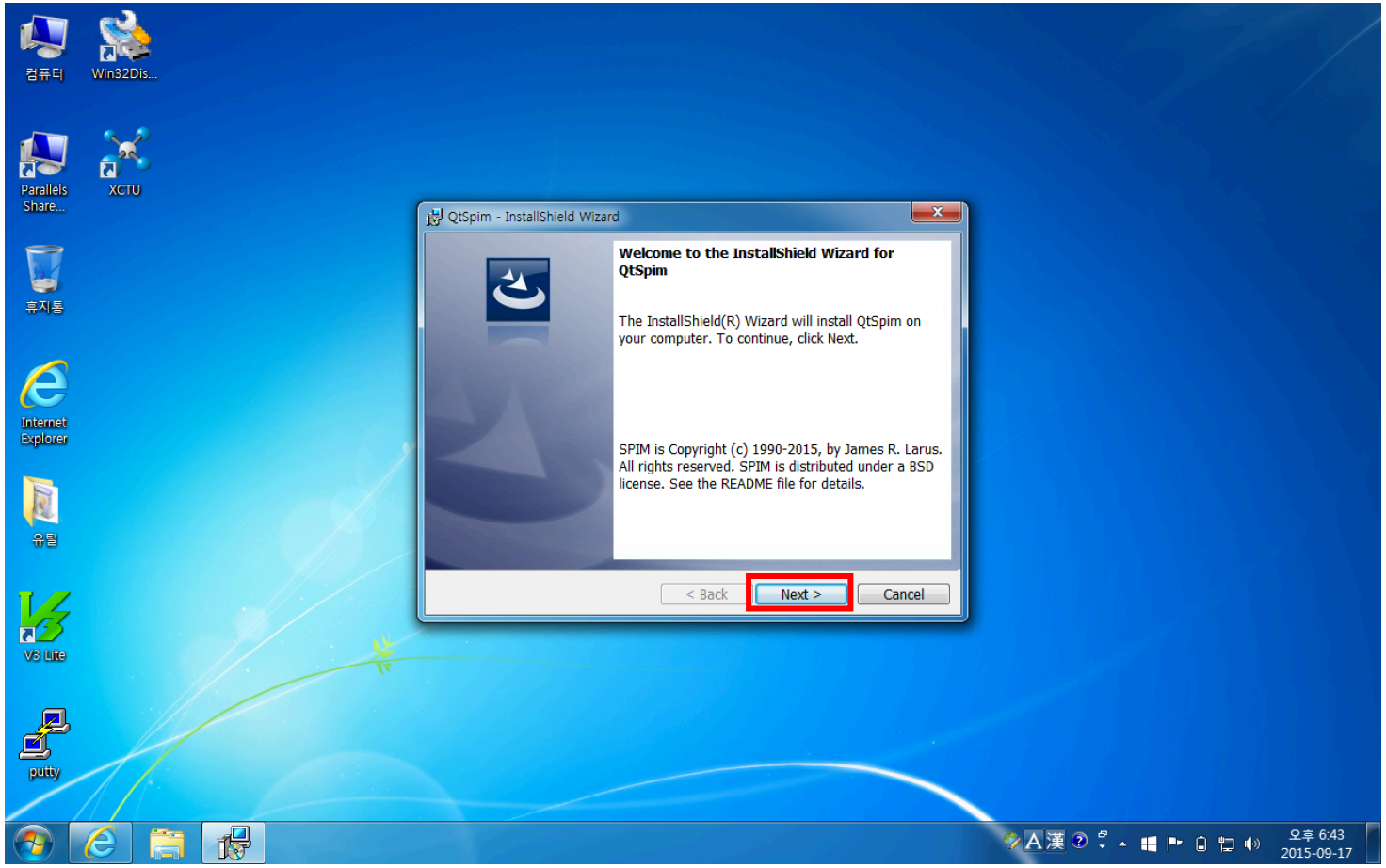
Download and Install



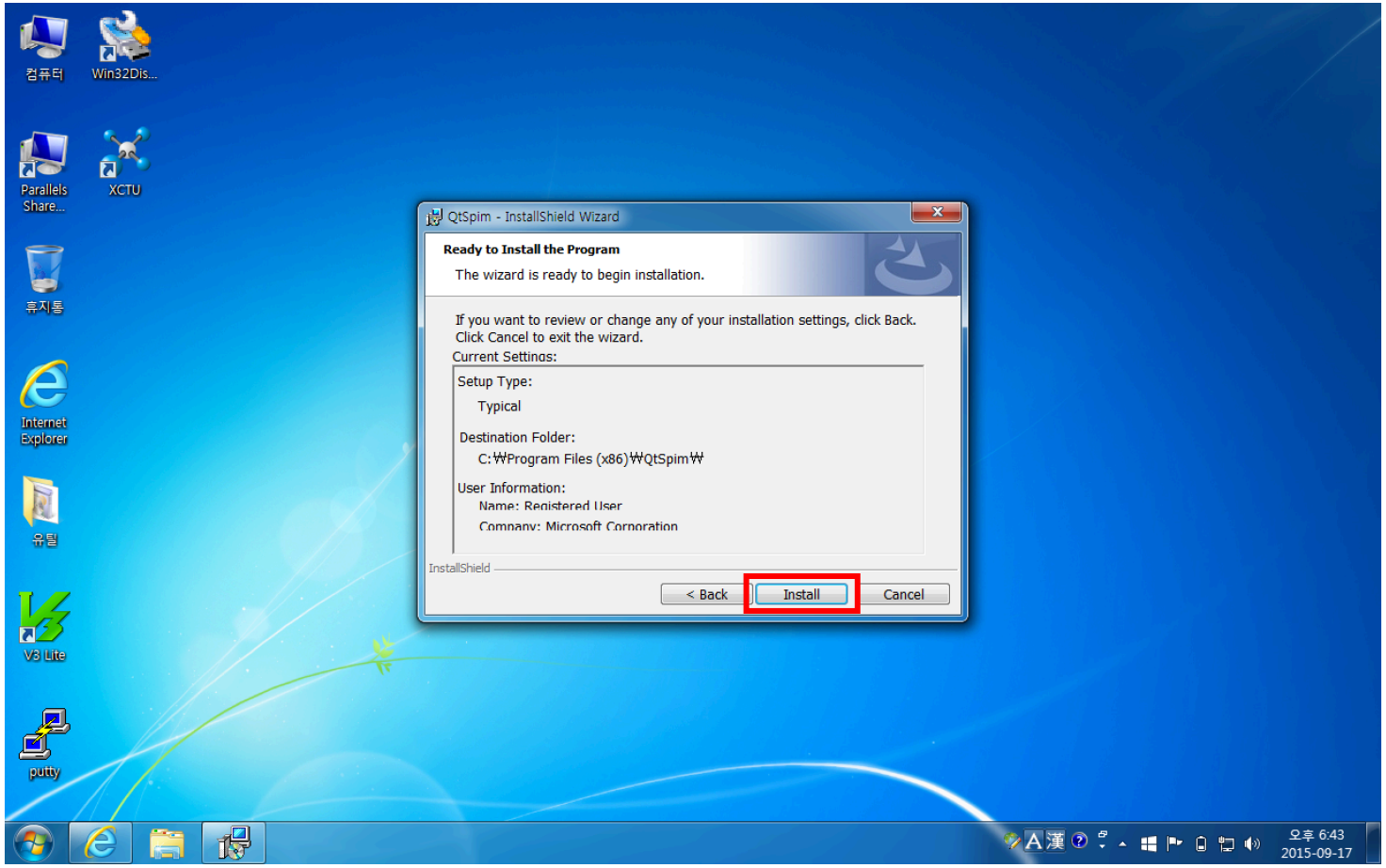
Download and Install



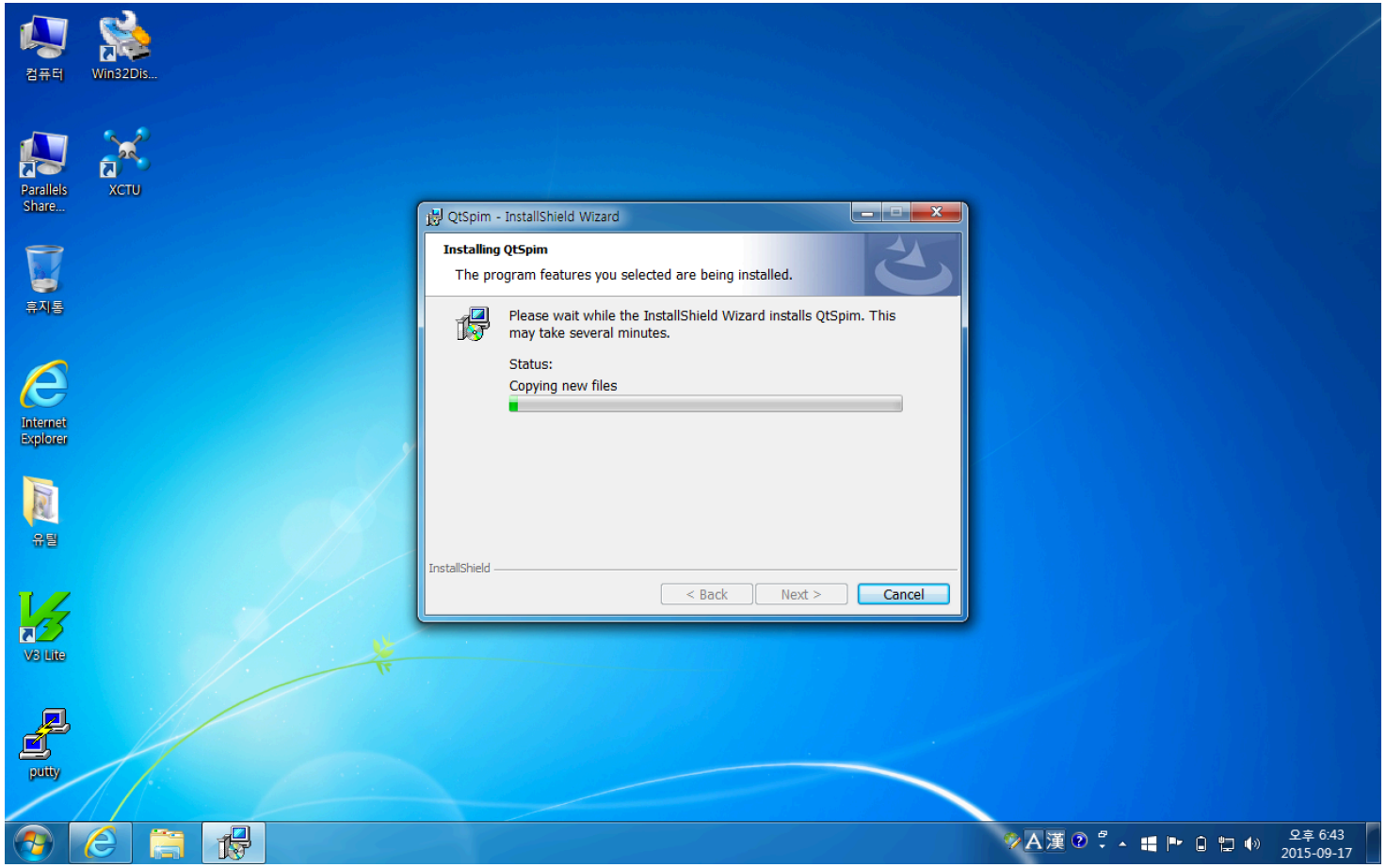
Download and Install



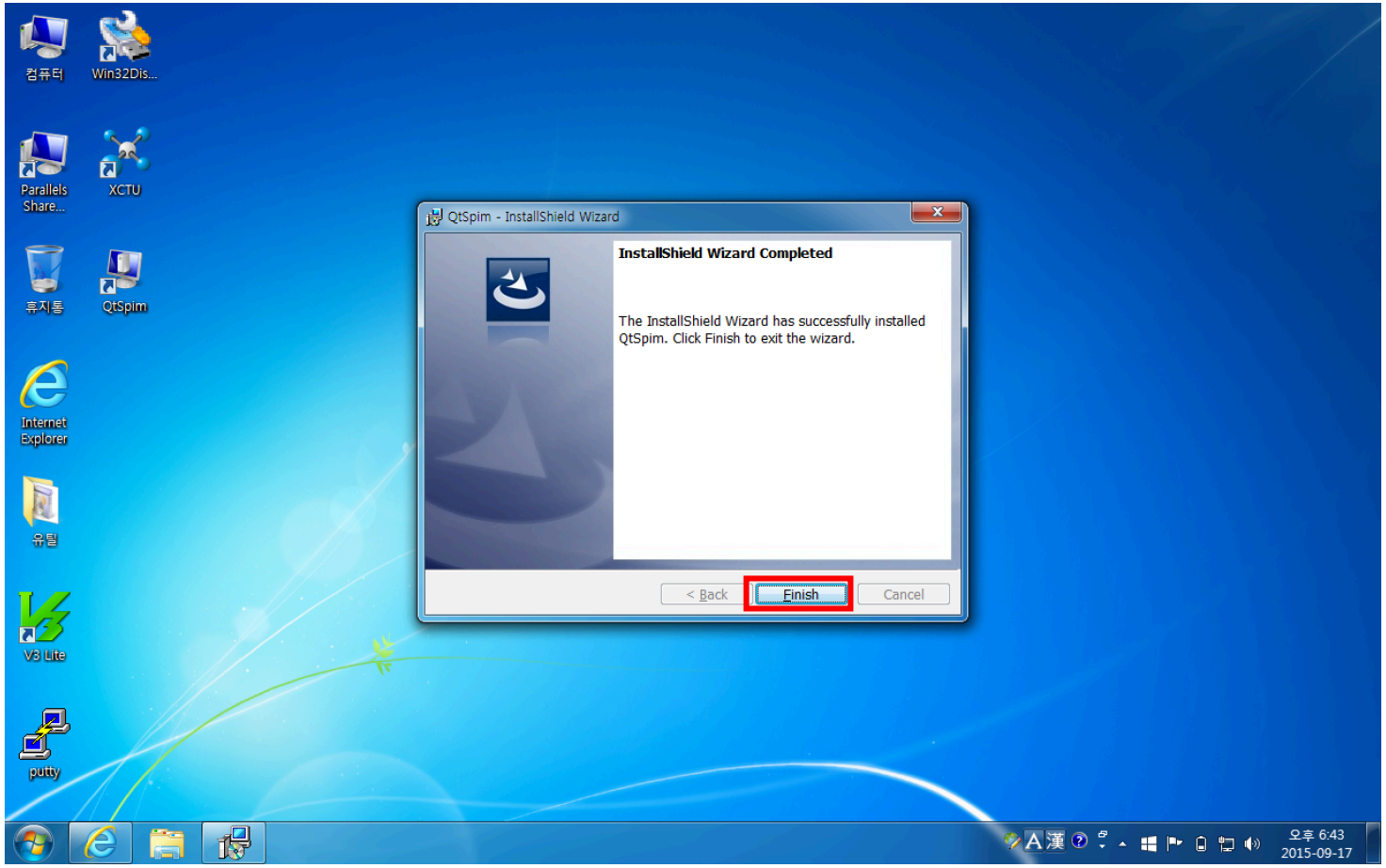
Download and Install



Download and Install



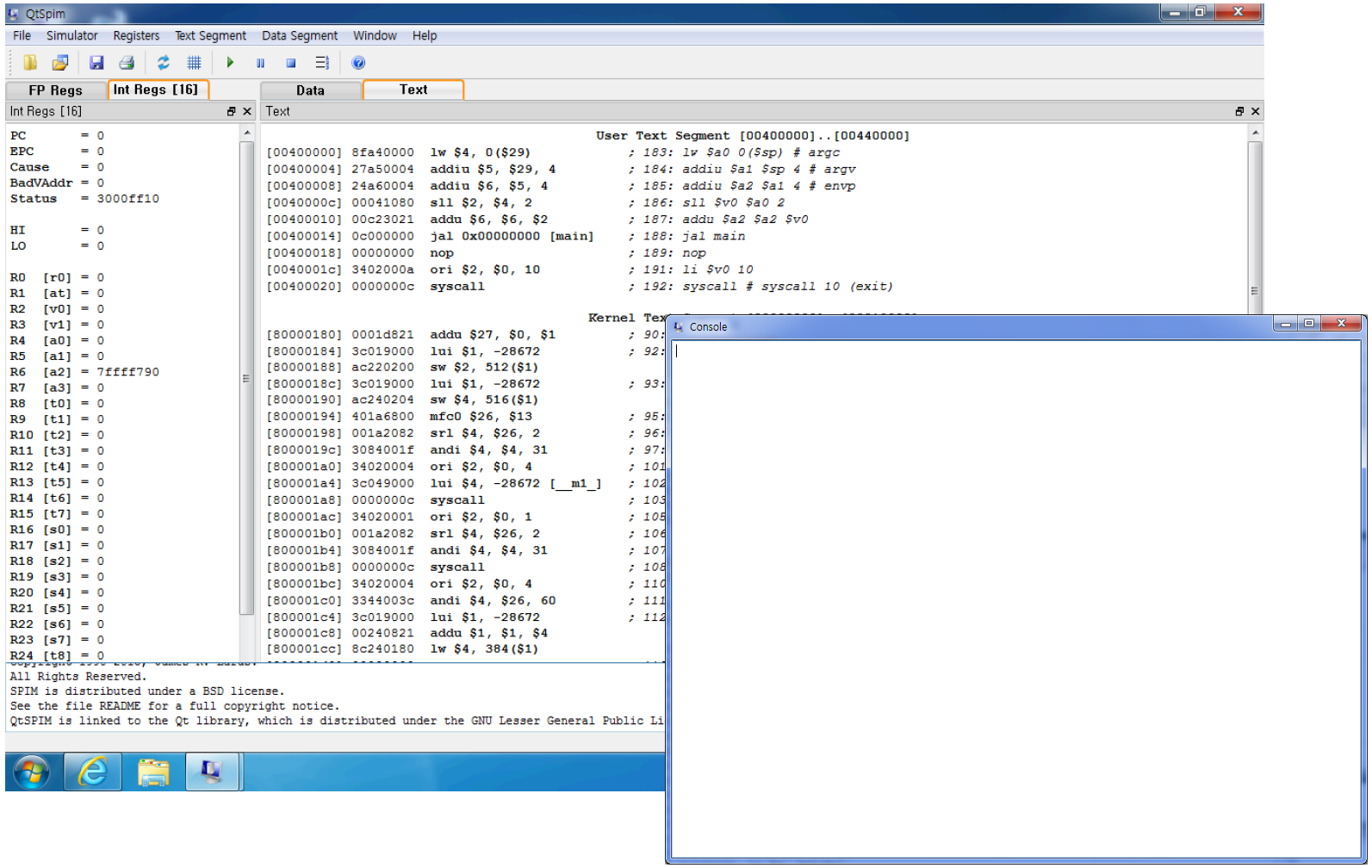
Download and Install



First screen of QtSpim

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

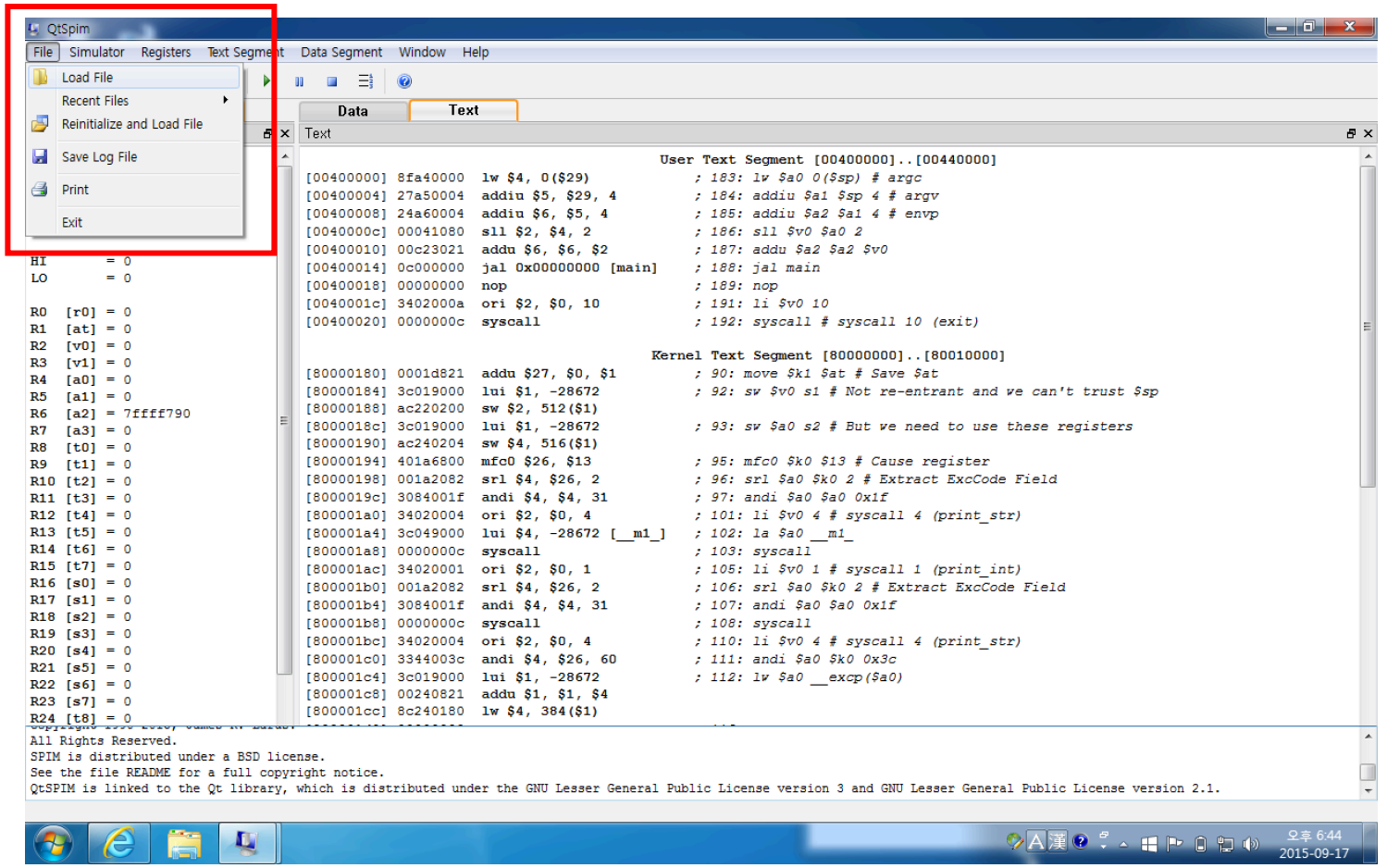
You can check the result with console window.



Loading assembly language file

Go to '**File**' ➔ '**Load File**', then load an assembly language file.

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



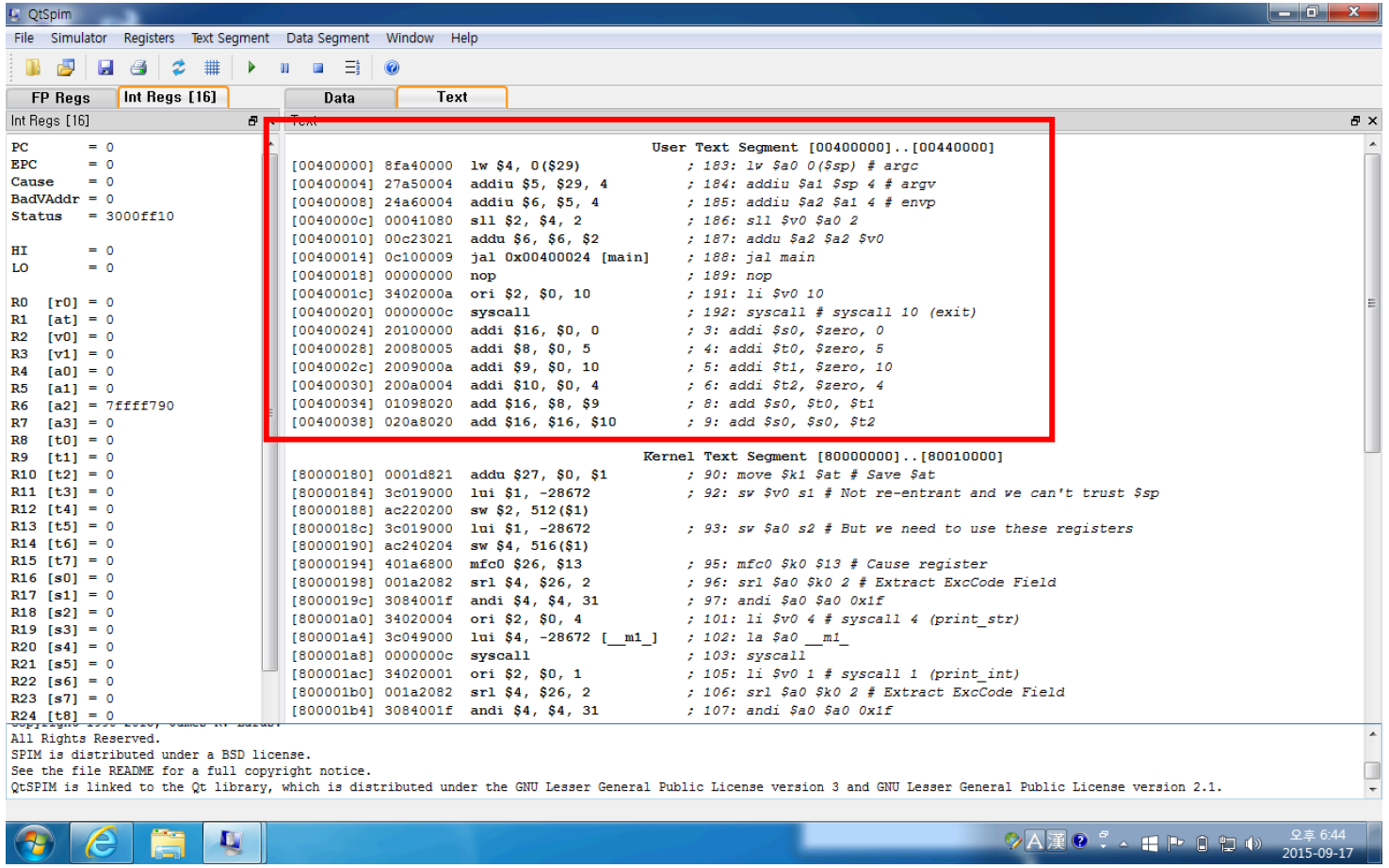
User file displayed

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

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

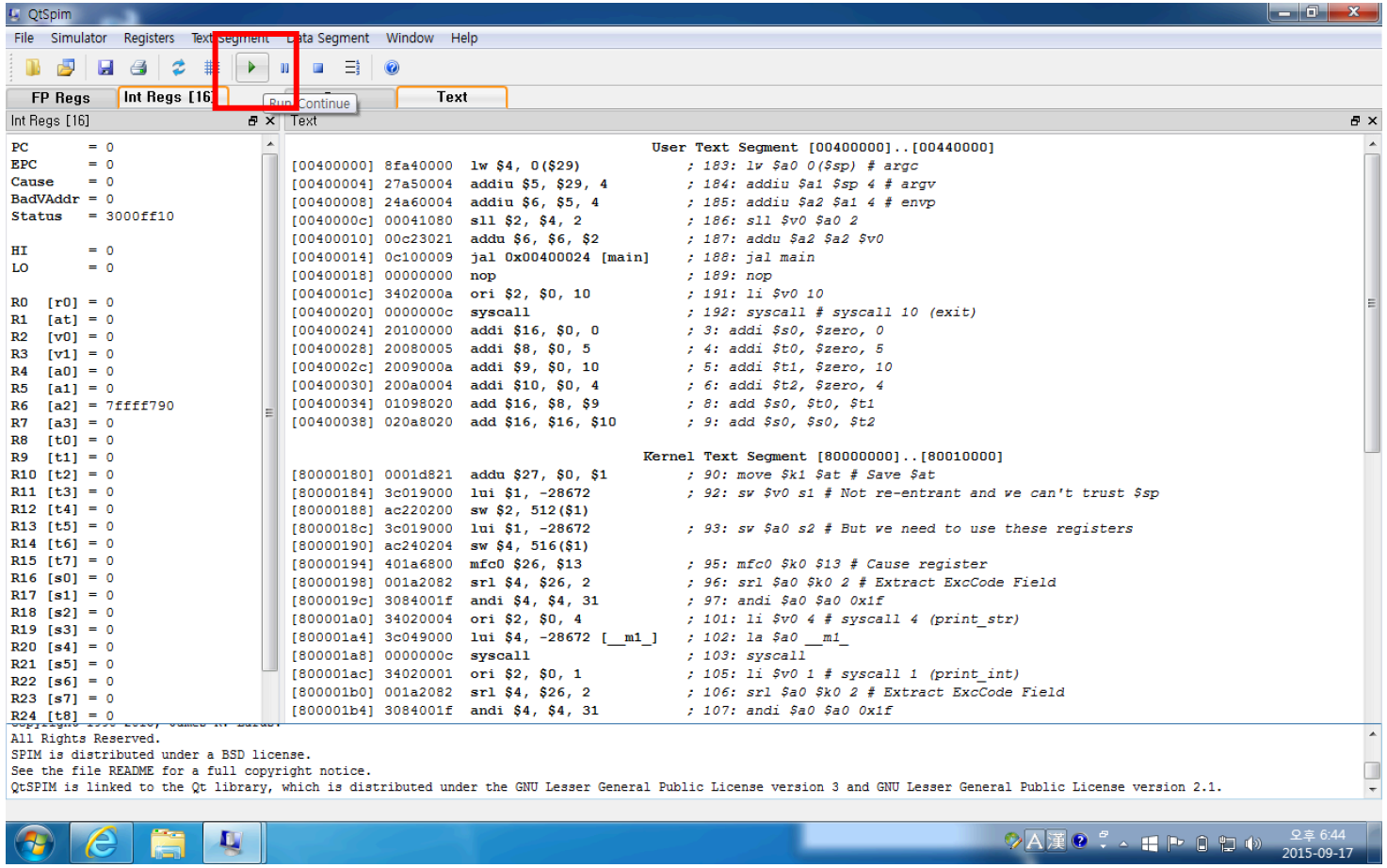
(The code below is the source used in the example.)

```
.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
```



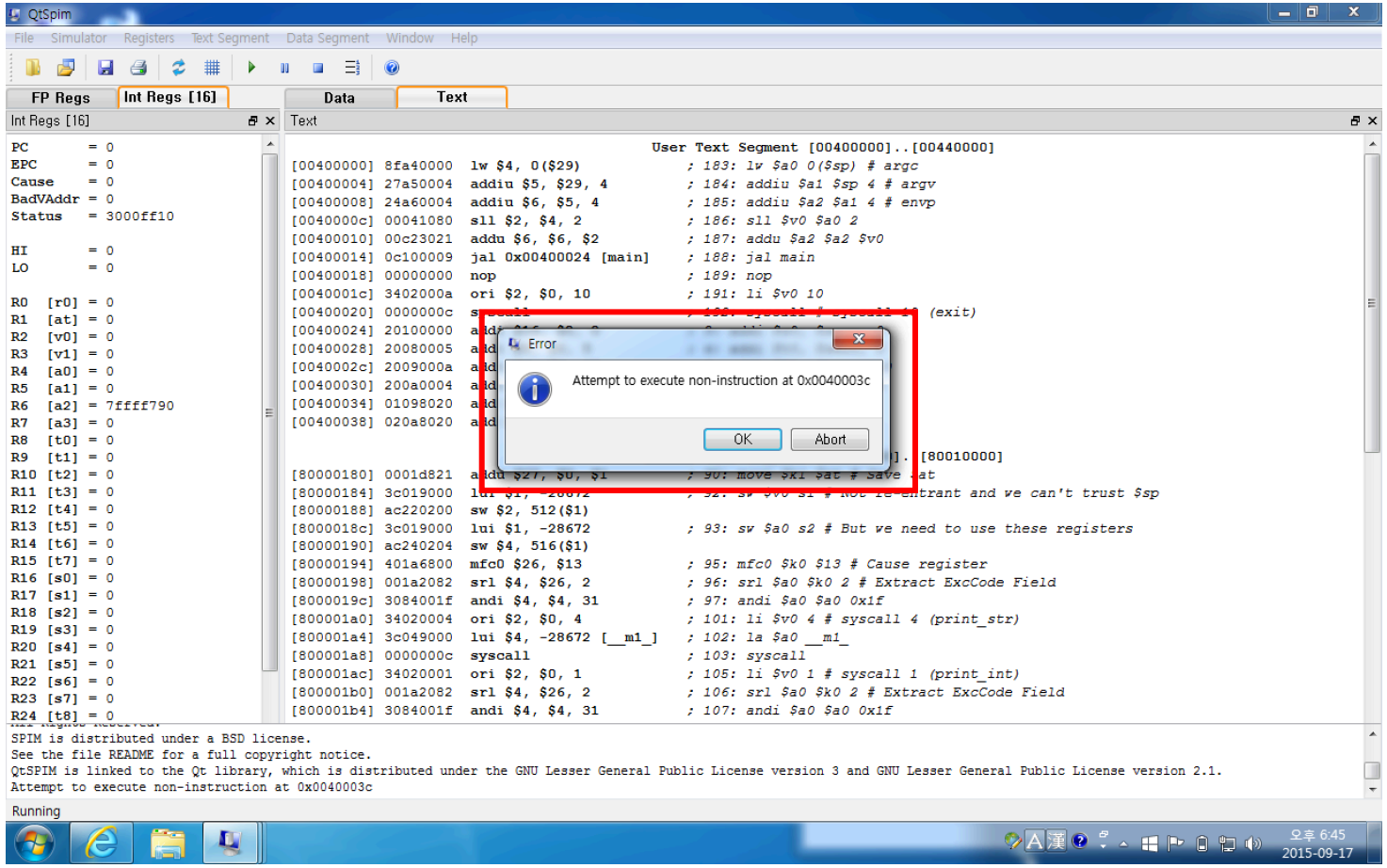
Execution

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



Message

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



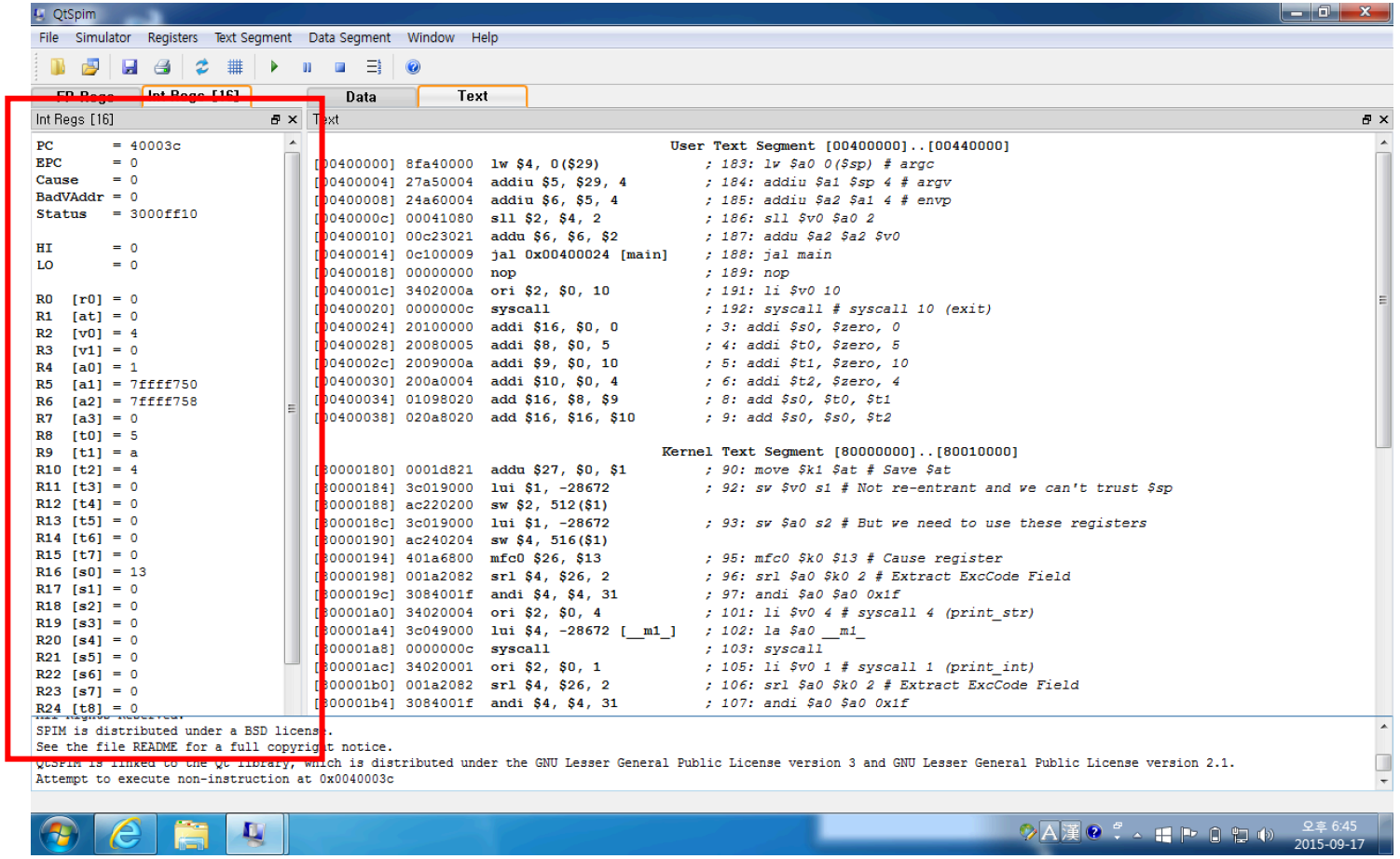
Checking register value

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

You can check the result of below source code.

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

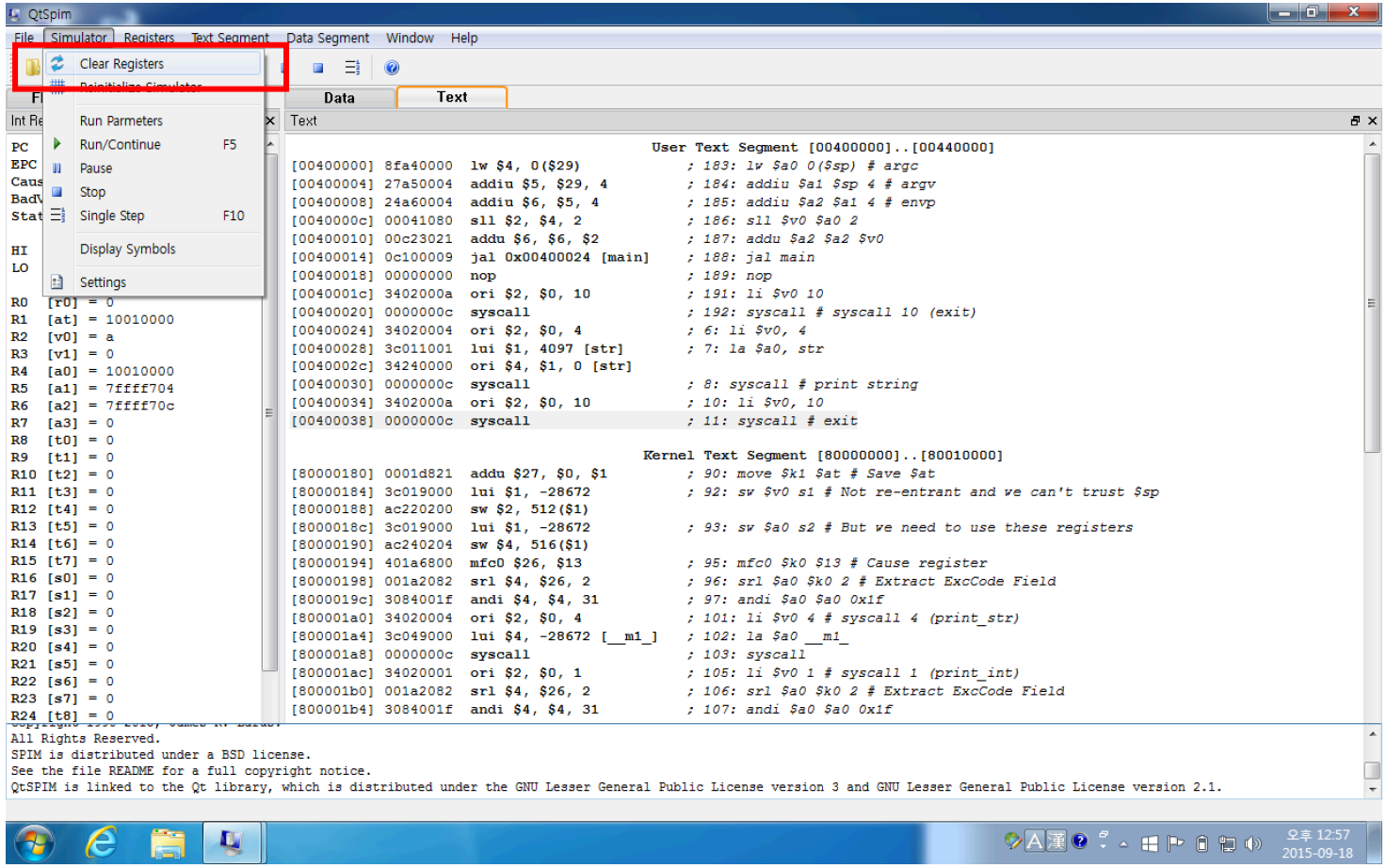
```
.text
main:
    addi $s0, $zero, 0
    addi $t0, $zero, 5
    addi $t1, $zero, 10
    addi $t2, $zero, 4

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



Register values are hexadecimal numbers

Re-Execution

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

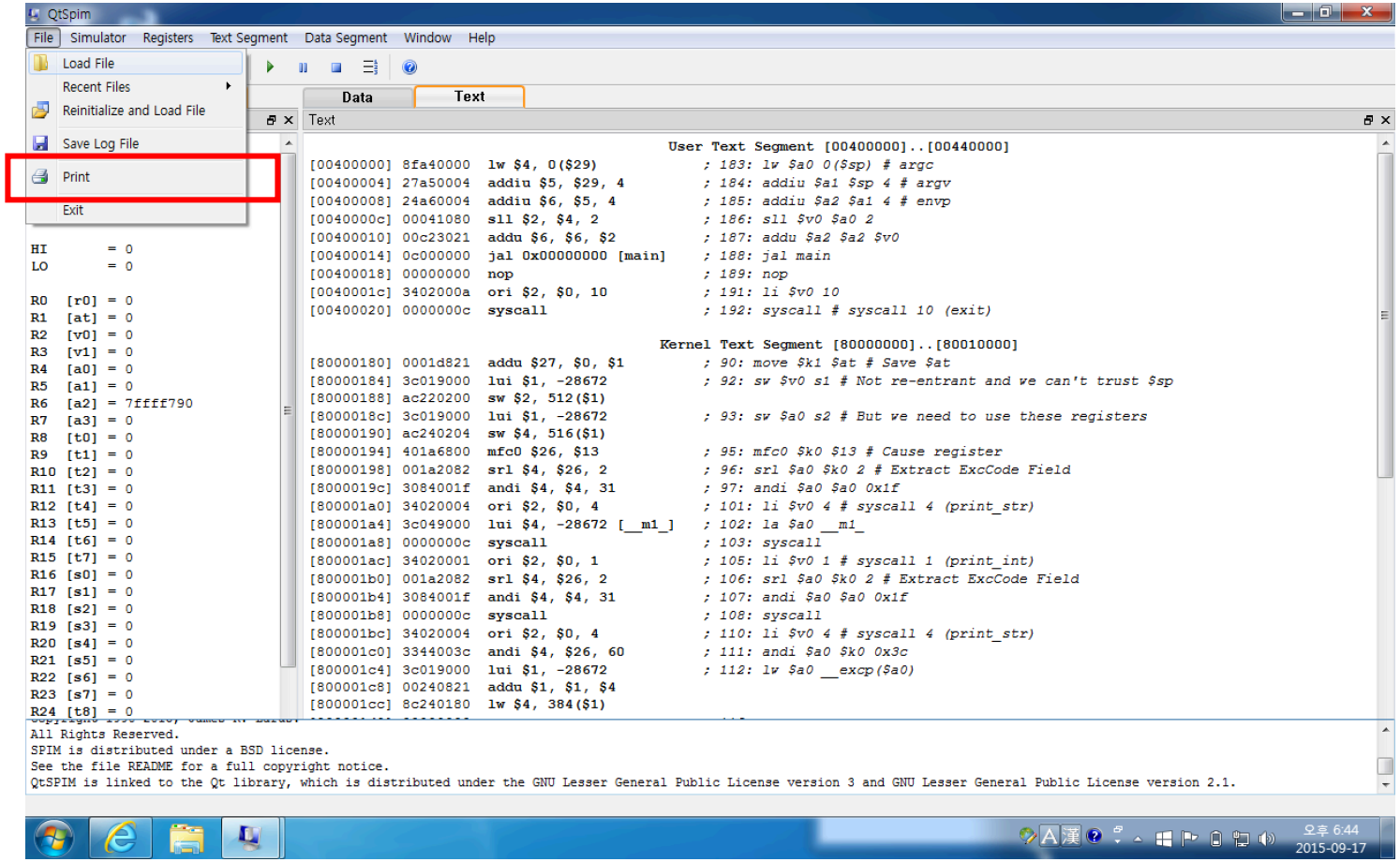


Loading another file

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

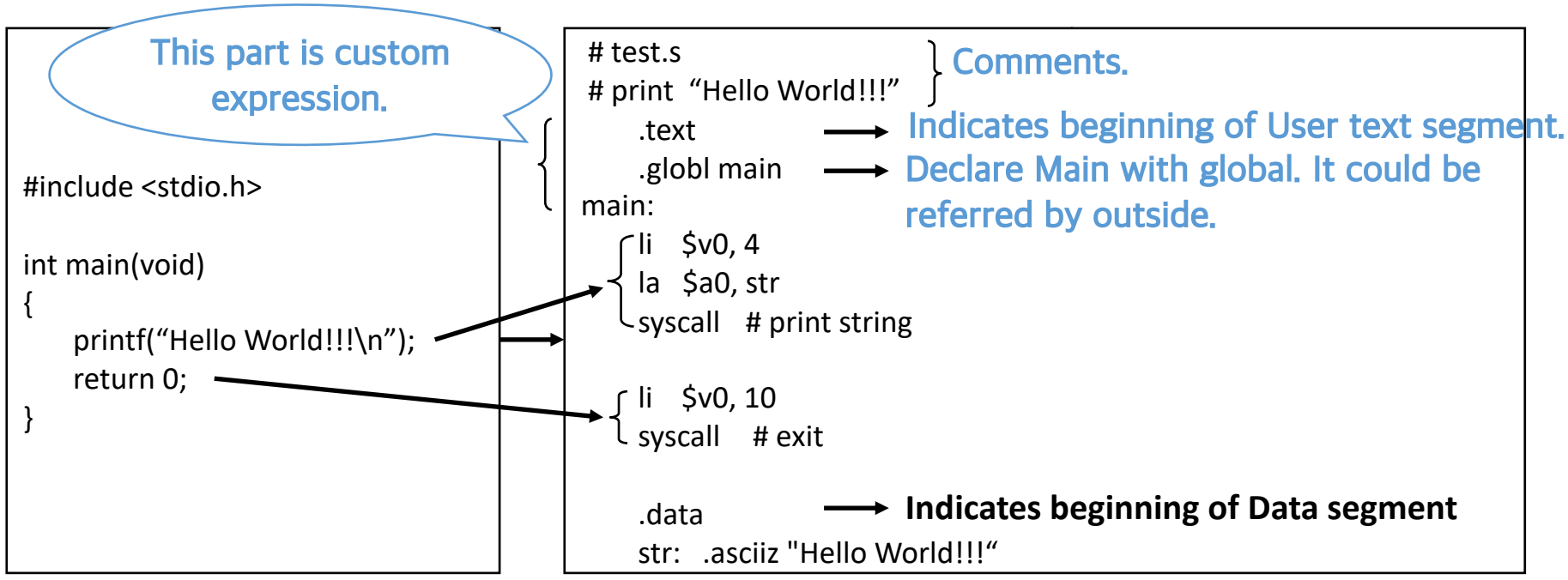
‘File’ ➔’Reinitialize and Load File’

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

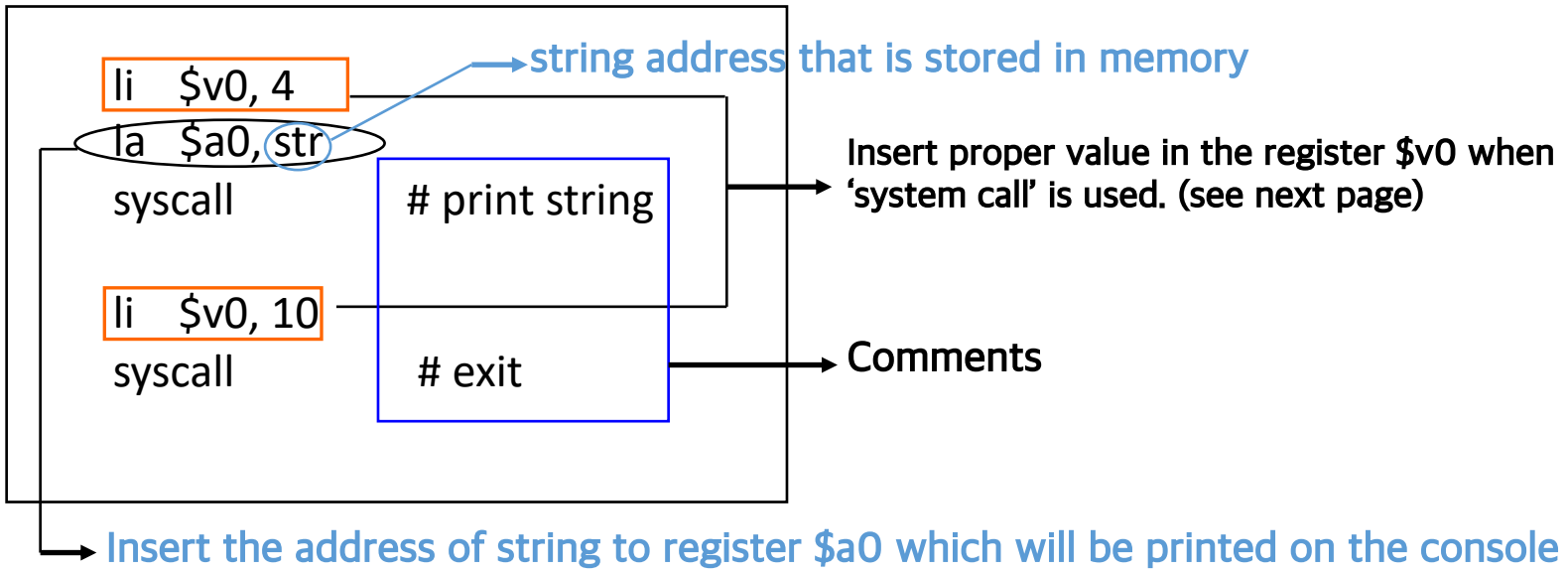



```
spim: (parser) Label is defined for the second time on line 9 of file
        main:
             ^
```

Example


This part is custom expression.
#include <stdio.h>
int main(void)
{
printf("Hello World!!!\n");
return 0;
}
# test.s
# print "Hello World!!!"
Comments.
.text
Indicates beginning of User text segment.
.globl main
Declare Main with global. It could be referred by outside.
main:
li $v0, 4
la $a0, str
syscall # print string
li $v0, 10
syscall # exit
.data
Indicates beginning of Data segment
str: .asciiz "Hello World!!!"

System call




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

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

System Call (System services)

$v0

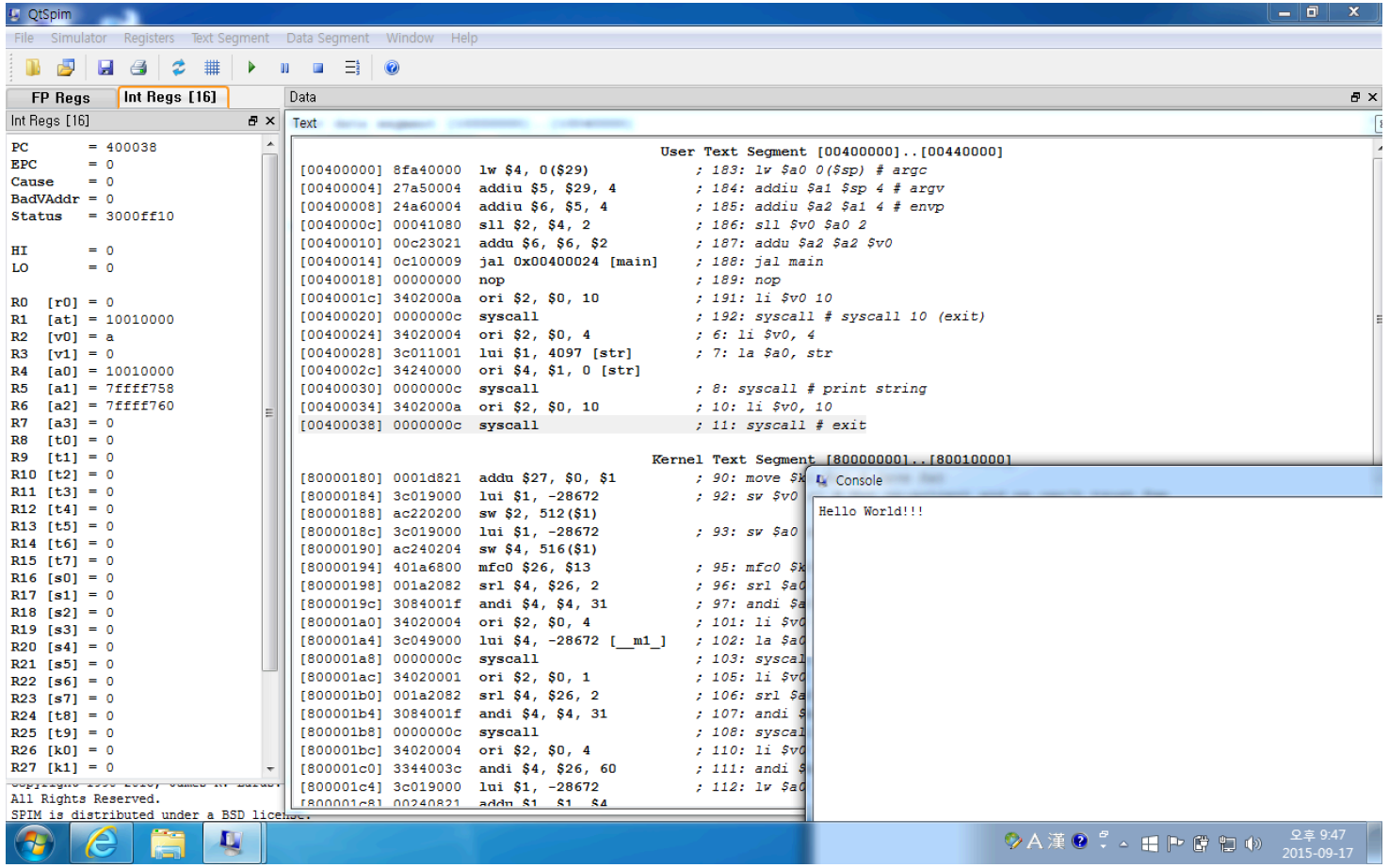
Service	System call code	Arguments	Result
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read_float	6		float (in $f0)
read_double	7		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	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	

malloc

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!!!"
```

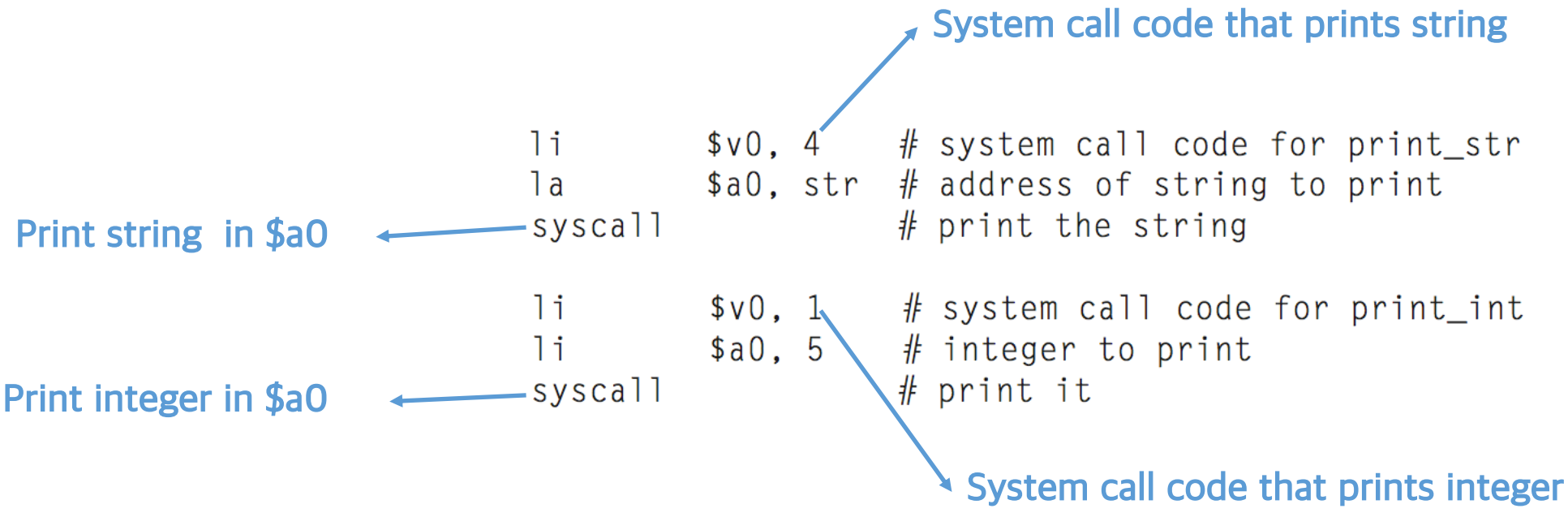


Pseudo Assembly Instruction

- **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

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 (syscall)

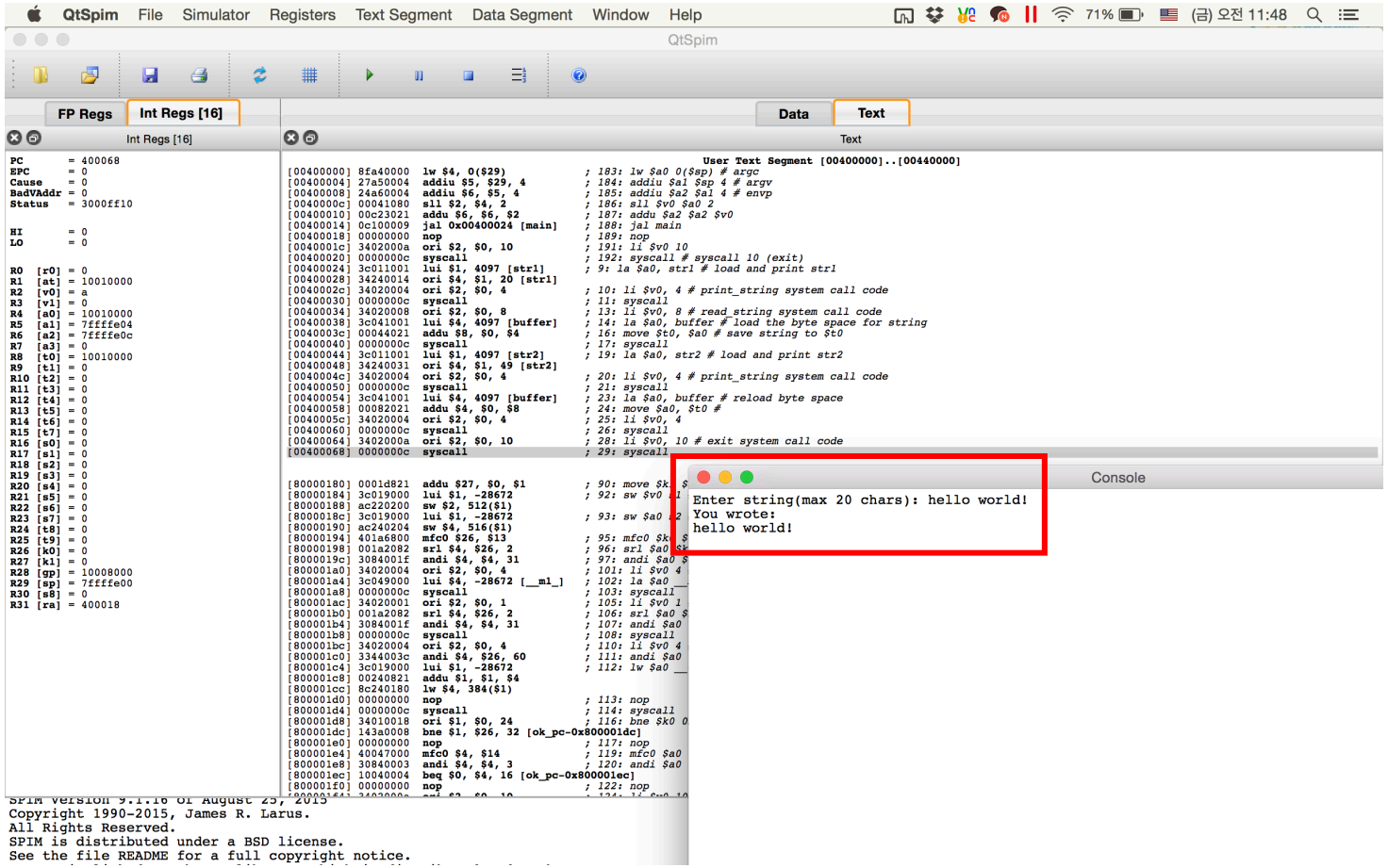
This program receives string from user and prints this string.

```
.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.

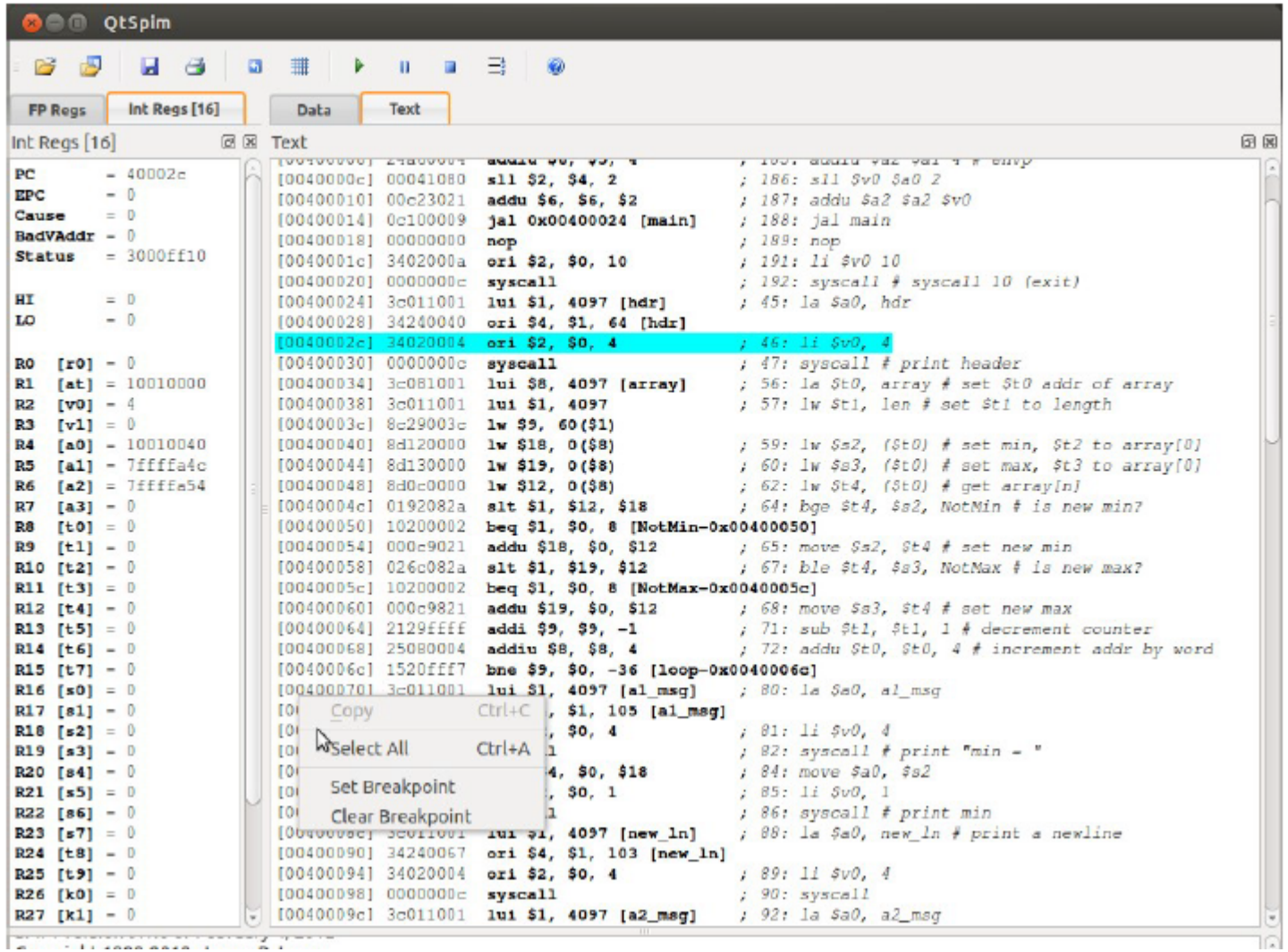
System Call (syscall)



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


QtSpim
FP Regs
Int Regs [16]
Data
Text
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] = 7fffffa4c
R6 [a2] = 7fffffa54
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
Copy Ctrl+C
Select All Ctrl+A
Set Breakpoint
Clear Breakpoint
[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