

---

# Computer Architecture

## SPIM Simulator

Made by Park, Byung-choon

---

# LIST

---

- ❑ **SPIM Definition**
- ❑ **Multiple versions**
- ❑ **Memory Layout**
- ❑ **PCSPIM**
- ❑ **Example**
- ❑ **Homework**

# SPIM

---

## ❑ Definition

- SPIM is a **software simulator that runs programs written for MIPS R2000/R3000 processors**
- SPIM can read and immediately execute assembly language files or MIPS executable files
- SPIM is a self-contained system
  - Debugger
  - A few operating system-like services

# SPIM of Multiple versions

---

## ❑ SPIM

- Command-line-driven program
- Requires only an alphanumeric terminal to display

## ❑ XSPIM

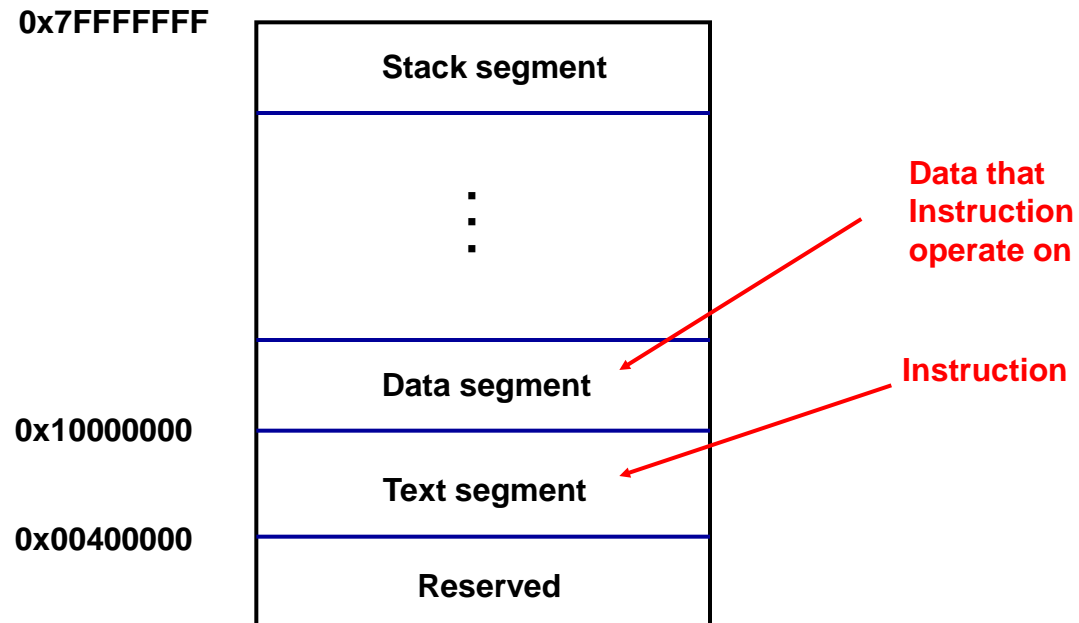
- X-windows environment
- Much easier program to learn

## ❑ PCSPIM

- Windows version of SPIM

# Memory Layout

---



# PCSPIM

---

- ❑ Download and Install
- ❑ Windows
  - Register display
  - Text segments
  - Data and stack segments
  - SPIM messages
- ❑ Function
  - Load
  - Go
  - Single step
  - Multiple steps
  - Breakpoint

# PCSPIM Download

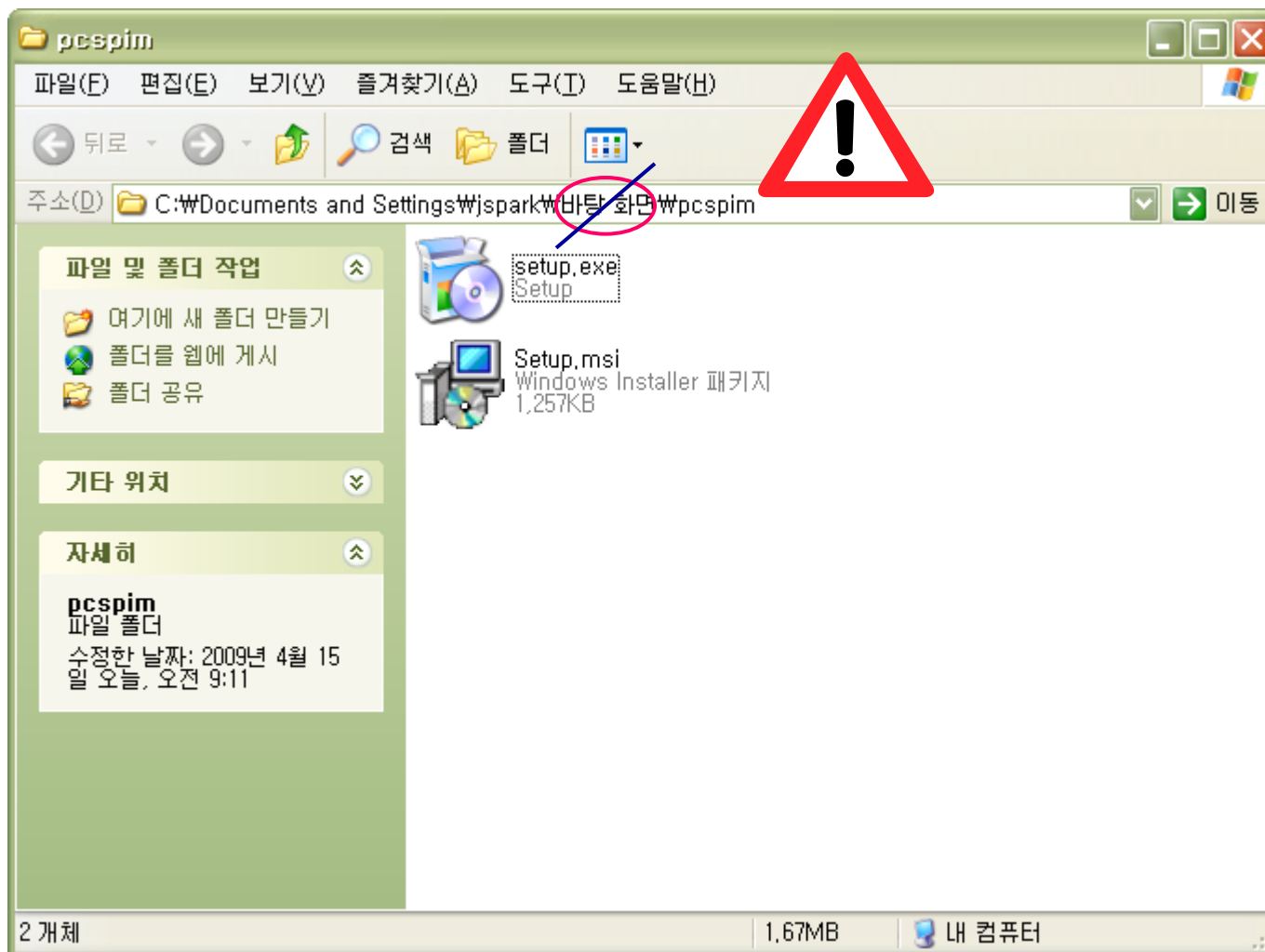
## ❑ Downloading PCSPIM

- <http://www.cs.wisc.edu/~larus/spim.html>

### Downloading SPIM

Platform	Program	Form	File
Unix or Linux system Mac OS X	<i>spim</i> <i>xspim</i>	Source code	<a href="http://www.cs.wisc.edu/~larus/SPIM/spim.tar.Z">http://www.cs.wisc.edu/~larus/SPIM/spim.tar.Z</a> or <a href="http://www.cs.wisc.edu/~larus/SPIM/spim.tar.gz">http://www.cs.wisc.edu/~larus/SPIM/spim.tar.gz</a>
Linux	<i>spim</i> <i>xspim</i>	Binary RPM for Fedora	<a href="http://www.cs.wisc.edu/cbi/downloads/">http://www.cs.wisc.edu/cbi/downloads/</a>
Microsoft Windows (Windows NT, 2000, XP)	<i>spim</i> <u><i>PCSpim</i></u>	Executable	<a href="http://www.cs.wisc.edu/~larus/SPIM/pcspim.zip">http://www.cs.wisc.edu/~larus/SPIM/pcspim.zip</a>
(spim 7.0 and later versions no longer run on Windows 95/98. Use version 6.5 or earlier.)		Source code	<a href="http://www.cs.wisc.edu/~larus/SPIM/pcspim_src.zip">http://www.cs.wisc.edu/~larus/SPIM/pcspim_src.zip</a>

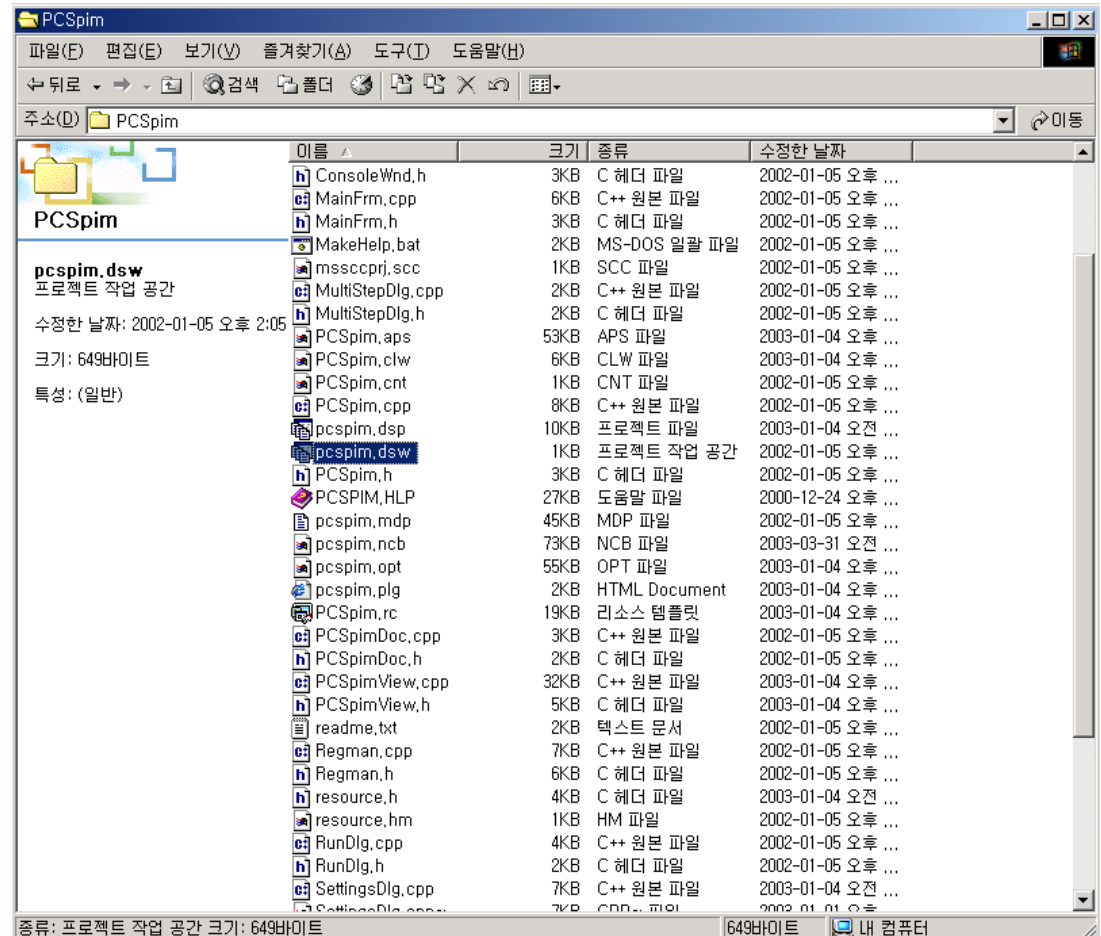
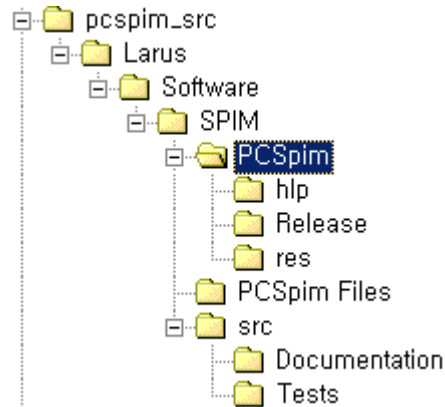
# Executable File Install





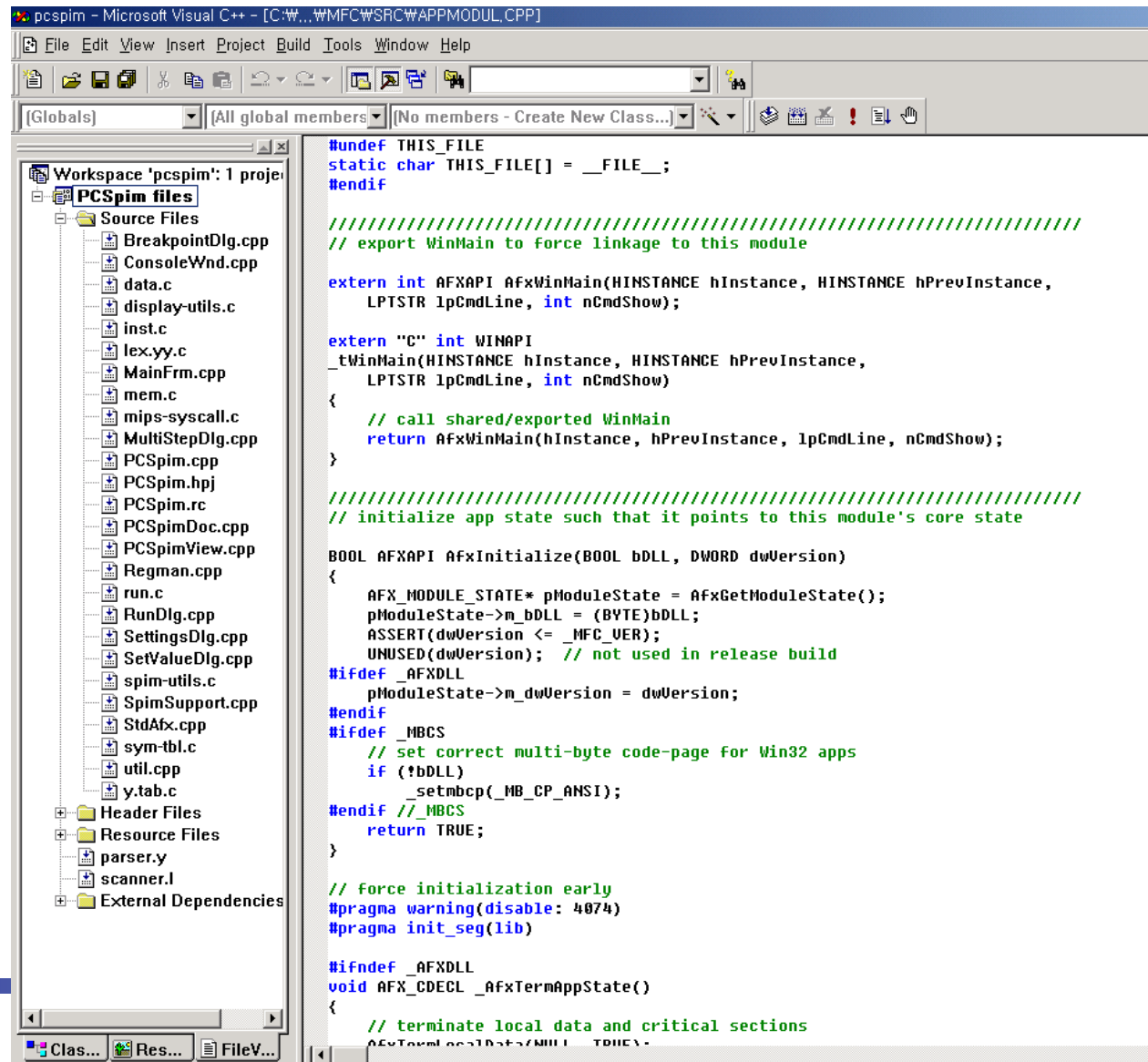
# Source Code Install

## Step 1



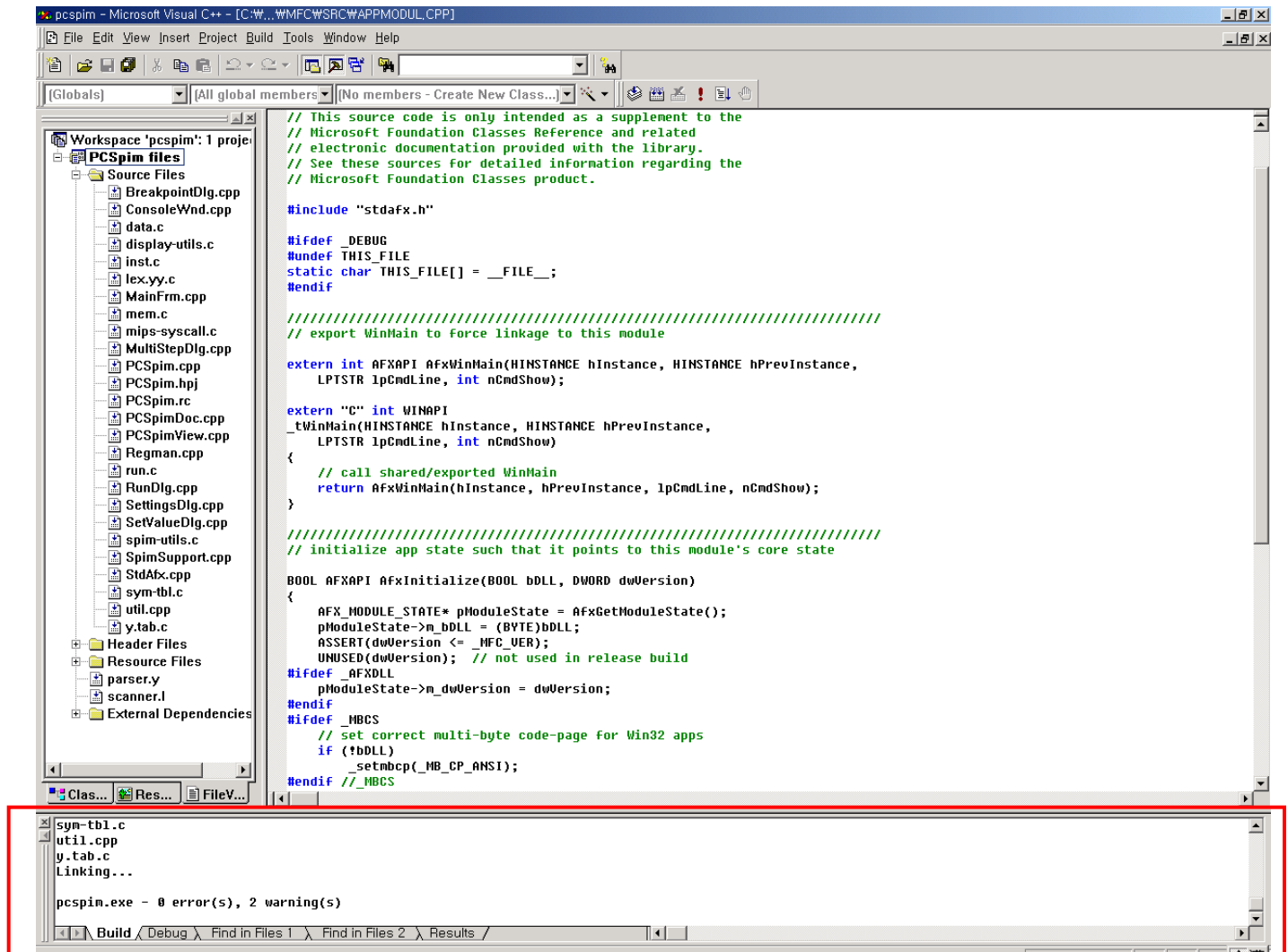
# Source Code Install

## Step 2



# Source Code Install

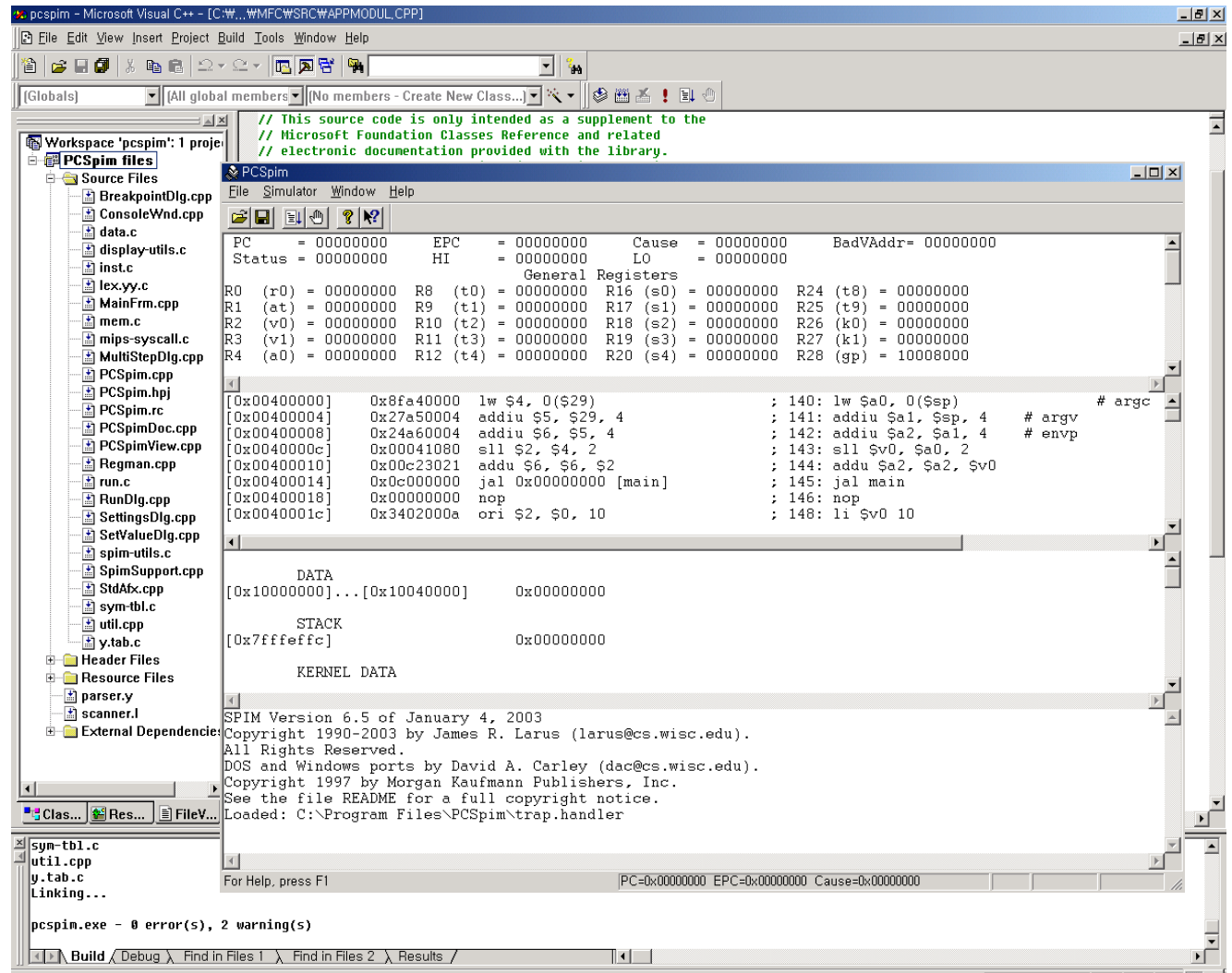
## Step 3



F7 - Compile

# Source Code Install

## Step 4



The screenshot shows the Microsoft Visual C++ IDE with the PCSpim project open. The left pane displays the project files, including source files, header files, resource files, and external dependencies. The right pane shows the assembly code for the main function, which is a MIPS assembly program. The code includes comments in Chinese and assembly instructions for loading arguments, setting up the stack, and calling the main function.

```
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.

PCSpim
File Simulator Window Help

PC = 00000000 EPC = 00000000 Cause = 00000000 BadVAddr= 00000000
Status = 00000000 HI = 00000000 LO = 00000000

General Registers
R0 (r0) = 00000000 R8 (t0) = 00000000 R16 (s0) = 00000000 R24 (t8) = 00000000
R1 (at) = 00000000 R9 (t1) = 00000000 R17 (s1) = 00000000 R25 (t9) = 00000000
R2 (v0) = 00000000 R10 (t2) = 00000000 R18 (s2) = 00000000 R26 (k0) = 00000000
R3 (v1) = 00000000 R11 (t3) = 00000000 R19 (s3) = 00000000 R27 (k1) = 00000000
R4 (a0) = 00000000 R12 (t4) = 00000000 R20 (s4) = 00000000 R28 (gp) = 10008000

[0x00400000] 0x8fa40000 lw $4, 0($29) ; 140: lw $a0, 0($sp) # argc
[0x00400004] 0x27a50004 addiu $5, $29, 4 ; 141: addiu $a1, $sp, 4 # argv
[0x00400008] 0x24a60004 addiu $6, $5, 4 ; 142: addiu $a2, $a1, 4 # envp
[0x0040000c] 0x00041080 sll $2, $4, 2 ; 143: sll $v0, $a0, 2
[0x00400010] 0x00c23021 addu $6, $6, $2 ; 144: addu $a2, $a2, $v0
[0x00400014] 0x0c000000 jal 0x00000000 [main] ; 145: jal main
[0x00400018] 0x00000000 nop ; 146: nop
[0x0040001c] 0x3402000a ori $2, $0, 10 ; 148: li $v0 10

DATA
[0x10000000]...[0x10040000] 0x00000000

STACK
[0x7fffffc] 0x00000000

KERNEL DATA

SPIM Version 6.5 of January 4, 2003
Copyright 1990-2003 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
DOS and Windows ports by David A. Carley (dac@cs.wisc.edu).
Copyright 1997 by Morgan Kaufmann Publishers, Inc.
See the file README for a full copyright notice.
Loaded: C:\Program Files\PCSpim\trap.handler

For Help, press F1 PC=0x00000000 EPC=0x00000000 Cause=0x00000000
```

**Ctrl + F5**  
**Program Run**

# Windows

The screenshot displays the PCSpim MIPS simulator interface. The window title is "PCSpim". The menu bar includes "File", "Simulator", "Window", and "Help". The toolbar contains icons for file operations and simulation control. The main display area is divided into several sections:

- Register display:** Shows the status of the processor (PC, Status, EPC, HI, Cause, L0, BadVAddr) and a table of 32 general registers (R0-R31) with their current values.
- Text segments:** Displays assembly code for the user program and the kernel. The user program code includes instructions like `lw $4, 0($29)`, `addiu $5, $29, 4`, `addiu $6, $5, 4`, `sll $2, $4, 2`, `addu $6, $6, $2`, `jal 0x00000000 [main]`, `nop`, `ori $2, $0, 10`, and `syscall`. The kernel code includes `lw $a0 0($ssp)`, `addiu $a1 $sp 4`, `addiu $a2 $a1 4`, `sll $v0 $a0 2`, `addu $a2 $a2 $v0`, `jal main`, `nop`, `li $v0 10`, and `syscall`.
- Data segments:** Shows the layout of data segments, including the `DATA` segment (address range `0x10000000` to `0x10040000`), the `STACK` segment (address `0x7ffffaa8`), and the `KERNEL DATA` segment (address range `0x90000000` to `0x90000030`).
- SPIM messages:** Displays version information (SPIM Version 8.0 of January 8, 2010), copyright notices (Copyright 1990-2010, James R. Larus; Copyright 1997, Morgan Kaufmann Publishers, Inc.), and the loaded program path (Loaded: C:\Program Files\PCSpim\exceptions.s).

The status bar at the bottom shows the current PC value (`PC=0x00000000`), EPC value (`EPC=0x00000000`), and Cause value (`Cause=0x00000000`).

Register display

Text segments

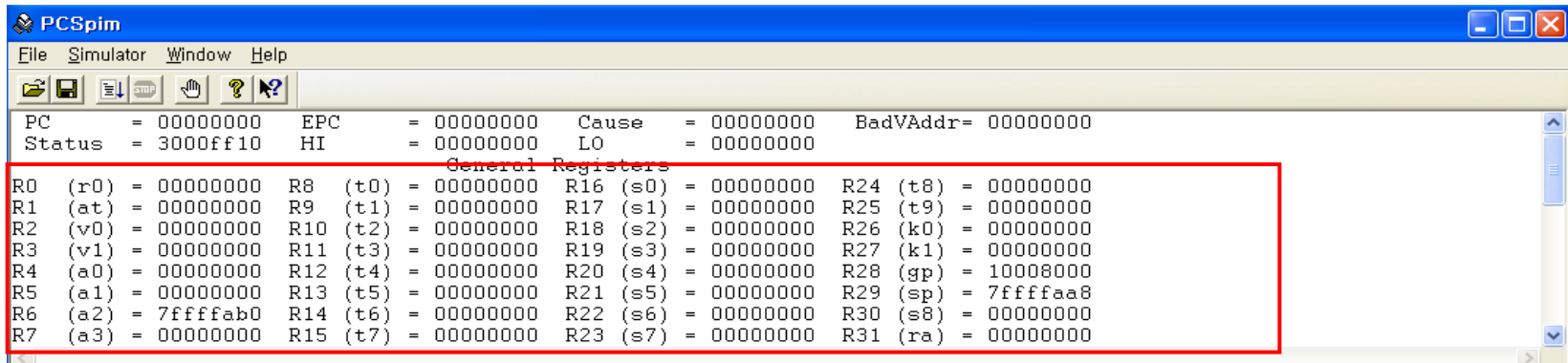
Data segments

SPIM messages

# Windows

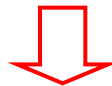
## ❑ Register display

It shows the values of all registers in the MIPS CPU and FPU.



The image shows a screenshot of the PCSpim MIPS simulator window. The title bar is blue with the text 'PCSpim'. Below the title bar is a menu bar with 'File', 'Simulator', 'Window', and 'Help'. Under the 'Simulator' menu, there are icons for running, stopping, and other simulation controls. The main display area shows the state of the MIPS CPU and FPU. At the top, it displays 'PC = 00000000', 'Status = 3000ff10', 'EPC = 00000000', 'HI = 00000000', 'Cause = 00000000', 'LO = 00000000', and 'BadVAddr = 00000000'. Below this, a section titled 'General Registers' is highlighted with a red border. This section contains a table of 32 registers, each with its name, alias, and value. The registers are arranged in four columns: R0-R7, R8-R15, R16-R23, and R24-R31. The values for R0-R7 are all 00000000. The values for R8-R15 are also 00000000. The values for R16-R23 are 00000000. The values for R24-R31 are 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 10008000, and 7ffffaa8 respectively.

Register	Value
R0 (r0)	00000000
R1 (at)	00000000
R2 (v0)	00000000
R3 (v1)	00000000
R4 (a0)	00000000
R5 (a1)	00000000
R6 (a2)	7ffffab0
R7 (a3)	00000000
R8 (t0)	00000000
R9 (t1)	00000000
R10 (t2)	00000000
R11 (t3)	00000000
R12 (t4)	00000000
R13 (t5)	00000000
R14 (t6)	00000000
R15 (t7)	00000000
R16 (s0)	00000000
R17 (s1)	00000000
R18 (s2)	00000000
R19 (s3)	00000000
R20 (s4)	00000000
R21 (s5)	00000000
R22 (s6)	00000000
R23 (s7)	00000000
R24 (t8)	00000000
R25 (t9)	00000000
R26 (k0)	00000000
R27 (k1)	00000000
R28 (gp)	10008000
R29 (sp)	7ffffaa8
R30 (s8)	00000000
R31 (ra)	00000000



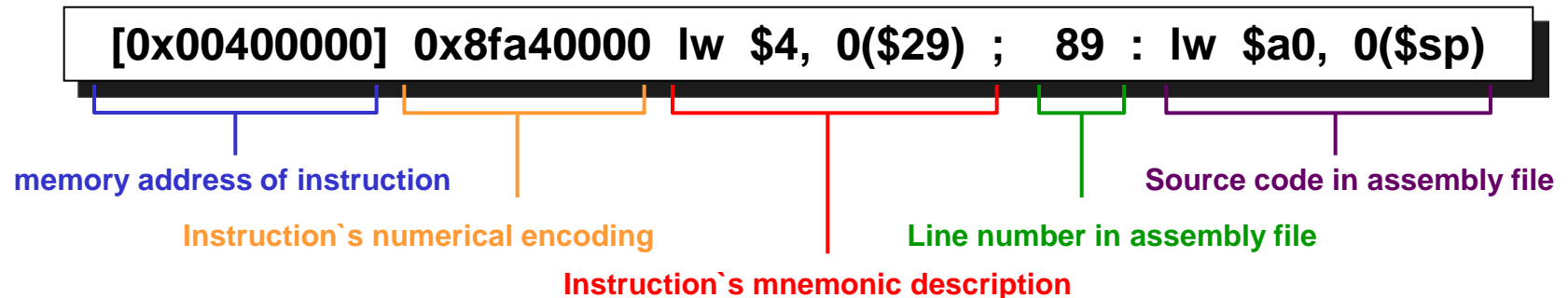
**General-purpose registers**

# Windows

## ❑ Text segments

Displays instructions both from your program and the system code that is loaded automatically when PCSPIM starts running

```
[0x00400000] 0x8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
[0x00400004] 0x27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
[0x00400008] 0x24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
[0x0040000c] 0x00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[0x00400010] 0x00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
[0x00400014] 0x0c000000 jal 0x00000000 [main] ; 188: jal main
[0x00400018] 0x00000000 nop ; 189: nop
[0x0040001c] 0x3402000a ori $2, $0, 10 ; 191: li $v0 10
[0x00400020] 0x0000000c syscall ; 192: syscall # syscall 10 (exit)
```



# Windows

---

- ❑ **Data segments (Data and stack segments)**

Displays the data loaded into your program`s memory and the data on the program`s stack

- ❑ **SPIM messages**

This is where error messages appear

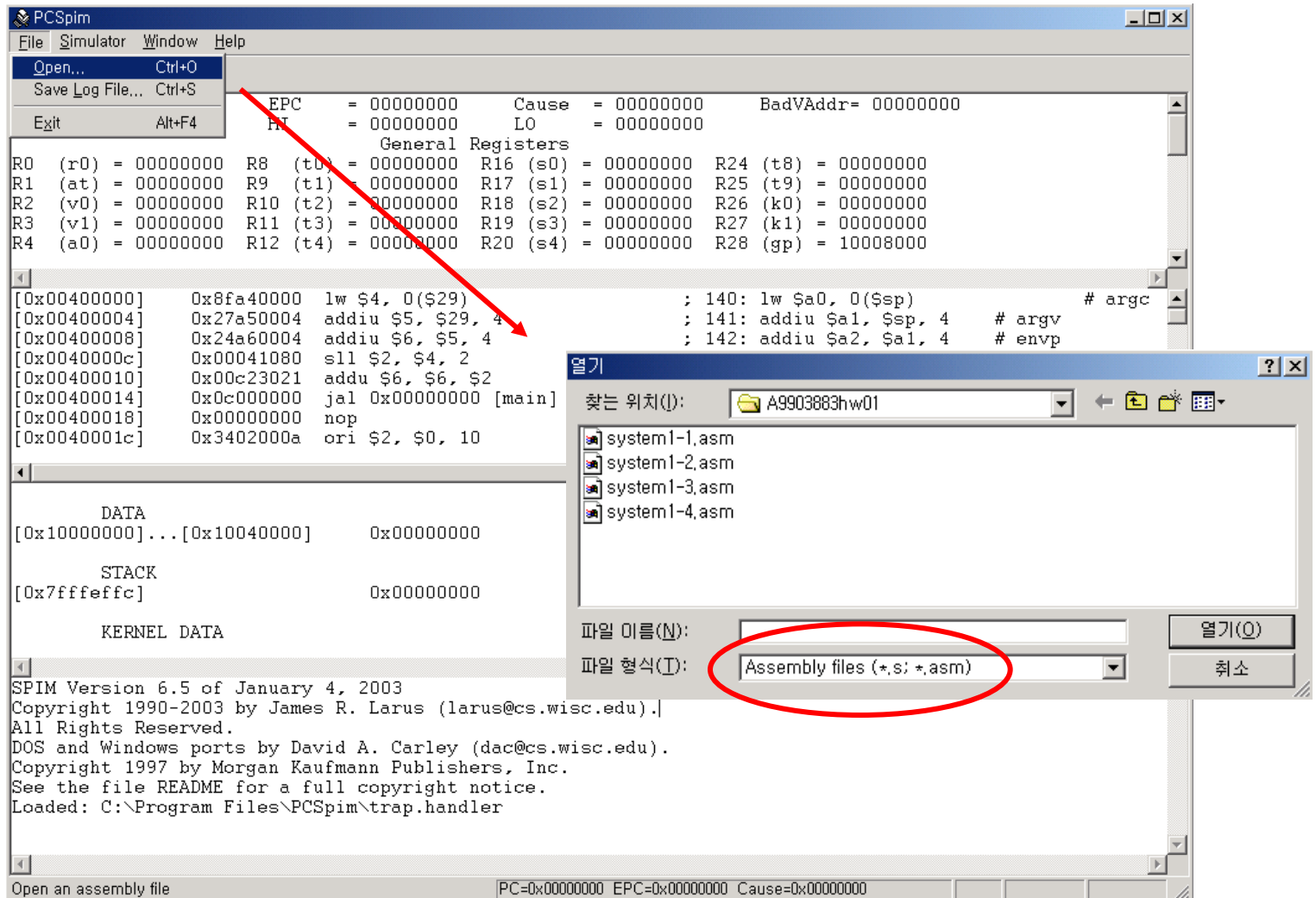


# Simulator Function

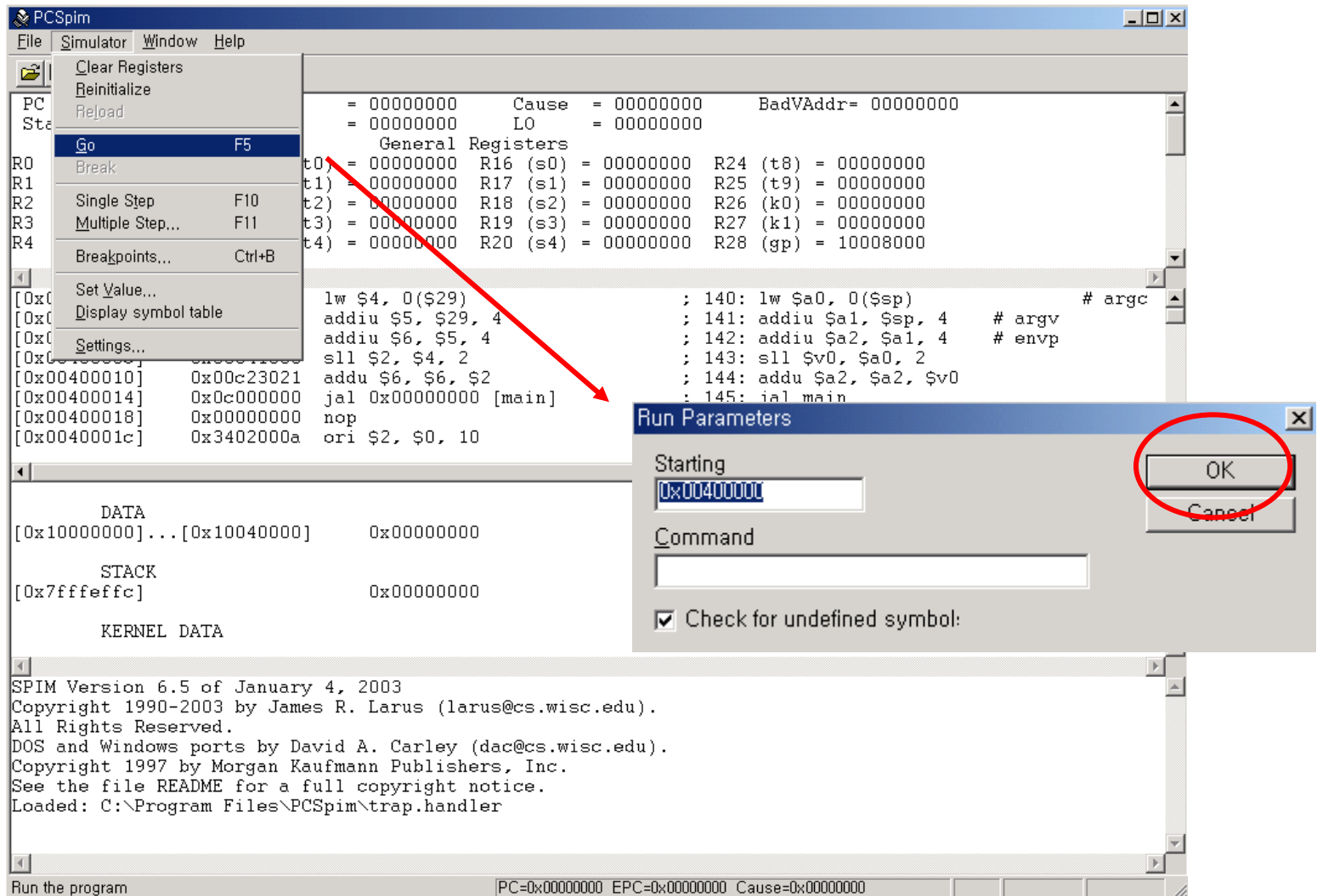
---

- ❑ **Load**
  - File → Open
  - Select assembly file
- ❑ **Reload**
  - File → Reload
  - Reload source file after change it with editor program
- ❑ **Go**
  - Simulator → Go
  - Results are displayed in console
- ❑ **Single step**
  - Simulator → Single Step
  - Run an instruction at a time
- ❑ **Multiple step**
  - Simulator → Multiple Step
  - Run given number of instruction at a time
- ❑ **Breakpoint**
  - Simulator → Breakpoint
  - Stop program immediately before it executes a particular instruction

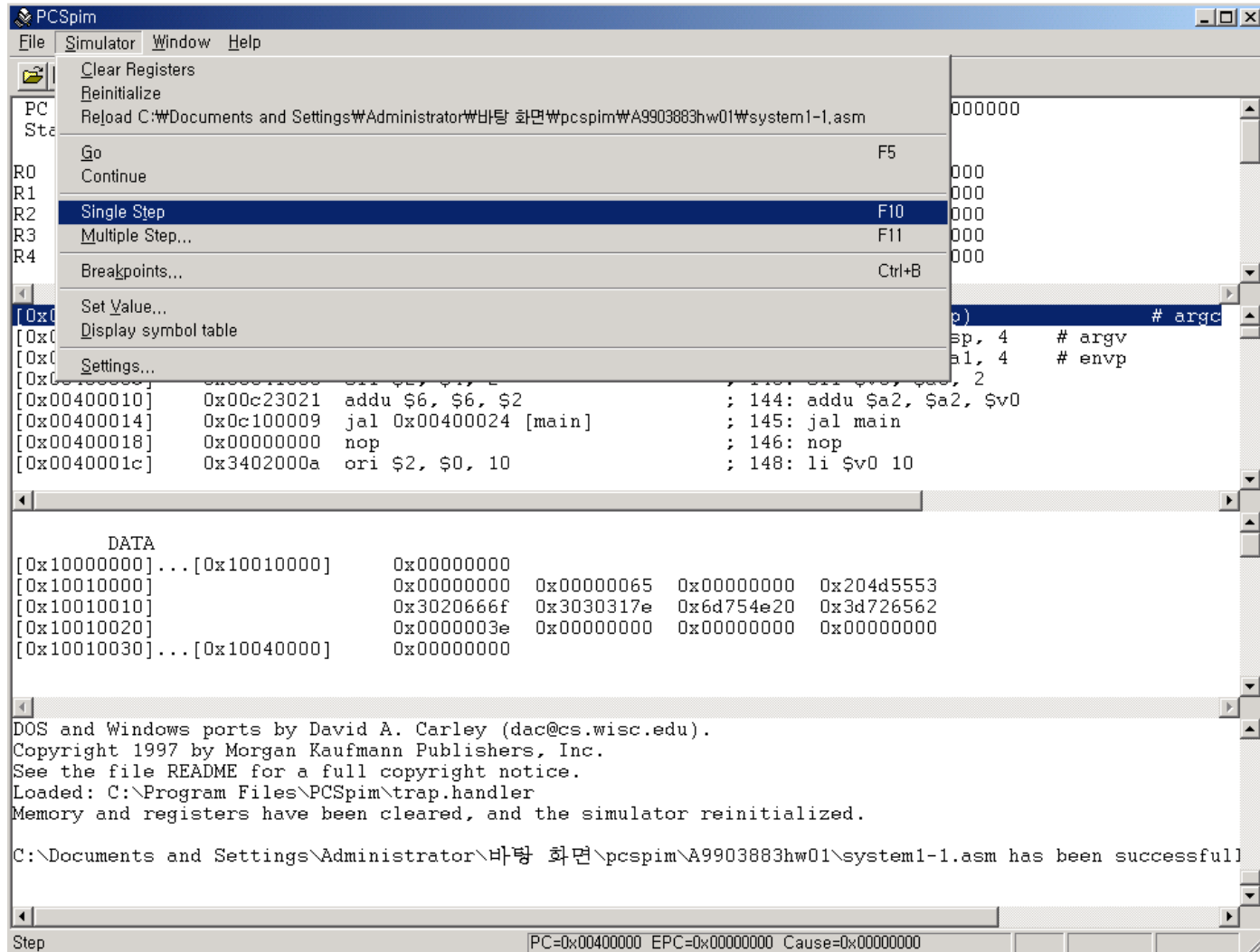
# Function - Load




# Function - Go



# Function – Single step



# Function – Single step



One step

```
PCSpim
File Simulator Window Help

PC = 00400004 EPC = 00000000 Cause = 00000000 BadVAddr= 00000000
Status = 00000000 HI = 00000000 LO = 00000000

General Registers
R0 (r0) = 00000000 R8 (t0) = 00000000 R16 (s0) = 00000000 R24 (t8) = 00000000
R1 (at) = 00000000 R9 (t1) = 00000000 R17 (s1) = 00000000 R25 (t9) = 00000000
R2 (v0) = 00000000 R10 (t2) = 00000000 R18 (s2) = 00000000 R26 (k0) = 00000000
R3 (v1) = 00000000 R11 (t3) = 00000000 R19 (s3) = 00000000 R27 (k1) = 00000000
R4 (a0) = 00000000 R12 (t4) = 00000000 R20 (s4) = 00000000 R28 (gp) = 10008000

[0x00400000] 0x8fa40000 lw $4, 0($29) ; 140: lw $a0, 0($sp) # argc
[0x00400004] 0x27a50004 addiu $5, $29, 4 ; 141: addiu $a1, $sp, 4 # argv
[0x00400008] 0x24a60004 addiu $6, $5, 4 ; 142: addiu $a2, $a1, 4 # envp
[0x0040000c] 0x00041080 sll $2, $4, 2 ; 143: sll $v0, $a0, 2
[0x00400010] 0x00c23021 addu $6, $6, $2 ; 144: addu $a2, $a2, $v0
[0x00400014] 0x0c100009 jal 0x00400024 [main] ; 145: jal main
[0x00400018] 0x00000000 nop ; 146: nop
[0x0040001c] 0x3402000a ori $2, $0, 10 ; 148: li $v0 10

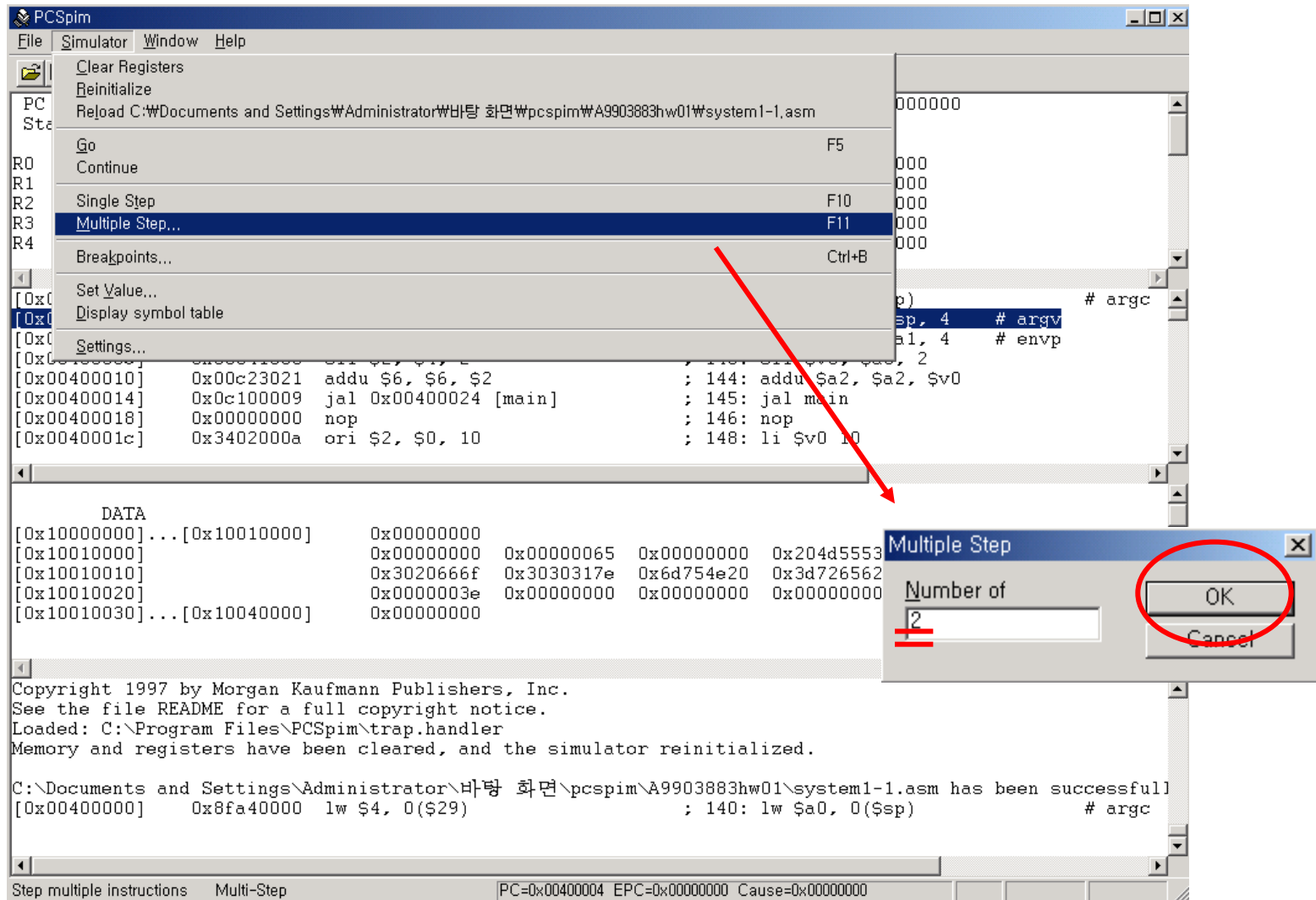
DATA
[0x10000000]...[0x10010000] 0x00000000
[0x10010000] 0x00000000 0x00000065 0x00000000 0x204d5553
[0x10010010] 0x3020666f 0x3030317e 0x6d754e20 0x3d726562
[0x10010020] 0x0000003e 0x00000000 0x00000000 0x00000000
[0x10010030]...[0x10040000] 0x00000000

Copyright 1997 by Morgan Kaufmann Publishers, Inc.
See the file README for a full copyright notice.
Loaded: C:\Program Files\PCSpim\trap.handler
Memory and registers have been cleared, and the simulator reinitialized.

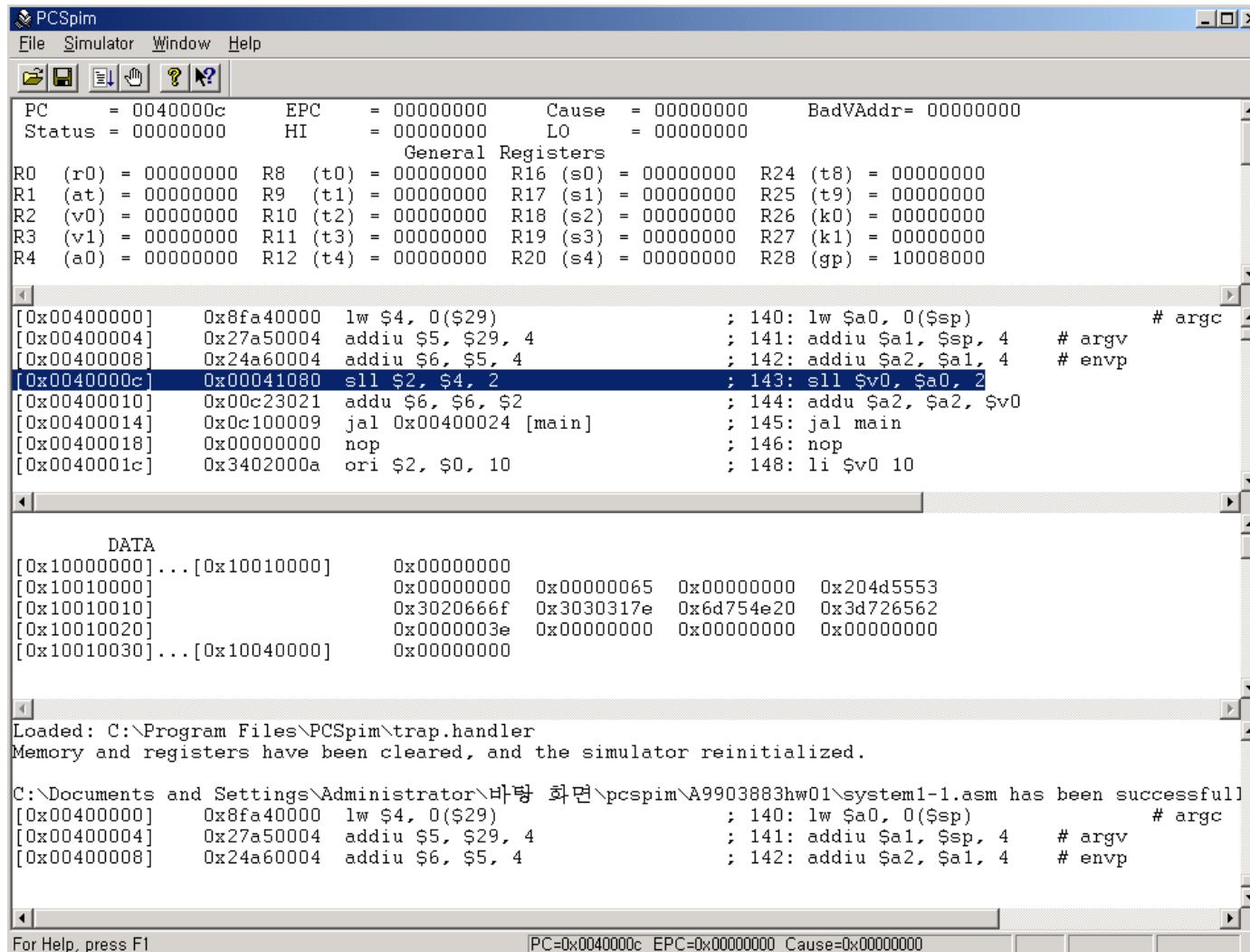
C:\Documents and Settings\Administrator\바탕 화면\pcspim\A9903883hw01\system1-1.asm has been successful
[0x00400000] 0x8fa40000 lw $4, 0($29) ; 140: lw $a0, 0($sp) # argc

For Help, press F1 PC=0x00400004 EPC=0x00000000 Cause=0x00000000
```

# Function – Multiple step



# Function – Multiple step



PCSpim

File Simulator Window Help

PC = 0040000c EPC = 00000000 Cause = 00000000 BadVAddr= 00000000  
Status = 00000000 HI = 00000000 LO = 00000000

General Registers

R0 (r0) = 00000000	R8 (t0) = 00000000	R16 (s0) = 00000000	R24 (t8) = 00000000
R1 (at) = 00000000	R9 (t1) = 00000000	R17 (s1) = 00000000	R25 (t9) = 00000000
R2 (v0) = 00000000	R10 (t2) = 00000000	R18 (s2) = 00000000	R26 (k0) = 00000000
R3 (v1) = 00000000	R11 (t3) = 00000000	R19 (s3) = 00000000	R27 (k1) = 00000000
R4 (a0) = 00000000	R12 (t4) = 00000000	R20 (s4) = 00000000	R28 (gp) = 10008000

[0x00400000] 0x8fa40000 lw \$4, 0(\$29) ; 140: lw \$a0, 0(\$sp) # argc  
[0x00400004] 0x27a50004 addiu \$5, \$29, 4 ; 141: addiu \$a1, \$sp, 4 # argv  
[0x00400008] 0x24a60004 addiu \$6, \$5, 4 ; 142: addiu \$a2, \$a1, 4 # envp  
[0x0040000c] 0x00041080 sll \$2, \$4, 2 ; 143: sll \$v0, \$a0, 2  
[0x00400010] 0x00c23021 addu \$6, \$6, \$2 ; 144: addu \$a2, \$a2, \$v0  
[0x00400014] 0x0c100009 jal 0x00400024 [main] ; 145: jal main  
[0x00400018] 0x00000000 nop ; 146: nop  
[0x0040001c] 0x3402000a ori \$2, \$0, 10 ; 148: li \$v0 10

DATA

[0x10000000] ... [0x10010000]	0x00000000
[0x10010000]	0x00000000
[0x10010010]	0x00000065 0x00000000 0x204d5553
[0x10010020]	0x3020666f 0x3030317e 0x6d754e20 0x3d726562
[0x10010030] ... [0x10040000]	0x0000003e 0x00000000 0x00000000 0x00000000

Loaded: C:\Program Files\PCSpim\trap.handler  
Memory and registers have been cleared, and the simulator reinitialized.

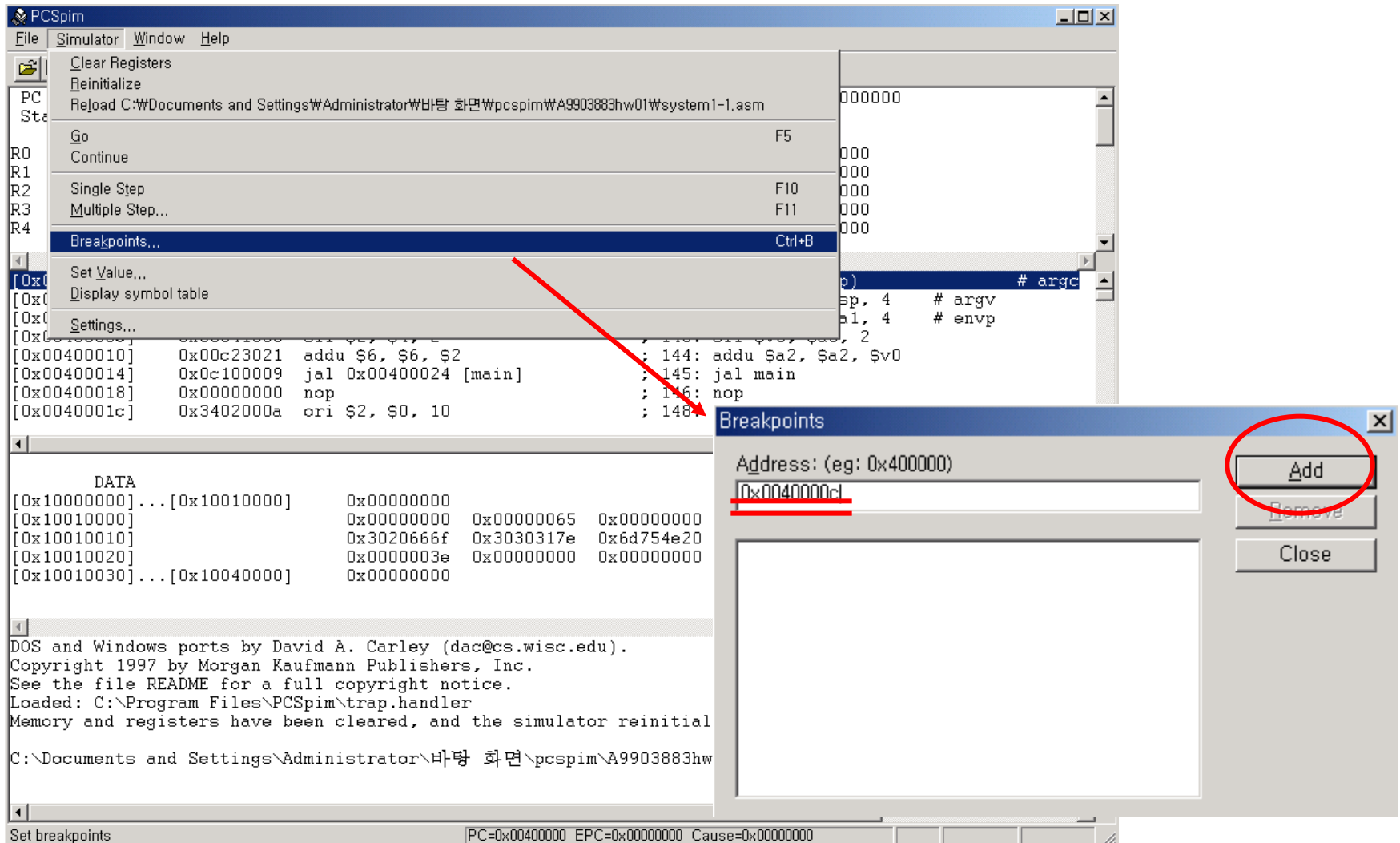
C:\Documents and Settings\Administrator\바탕 화면\pcspim\A9903883hw01\system1-1.asm has been successfully loaded.

[0x00400000] 0x8fa40000 lw \$4, 0(\$29) ; 140: lw \$a0, 0(\$sp) # argc  
[0x00400004] 0x27a50004 addiu \$5, \$29, 4 ; 141: addiu \$a1, \$sp, 4 # argv  
[0x00400008] 0x24a60004 addiu \$6, \$5, 4 ; 142: addiu \$a2, \$a1, 4 # envp

For Help, press F1

PC=0x0040000c EPC=0x00000000 Cause=0x00000000

# Function - breakpoint



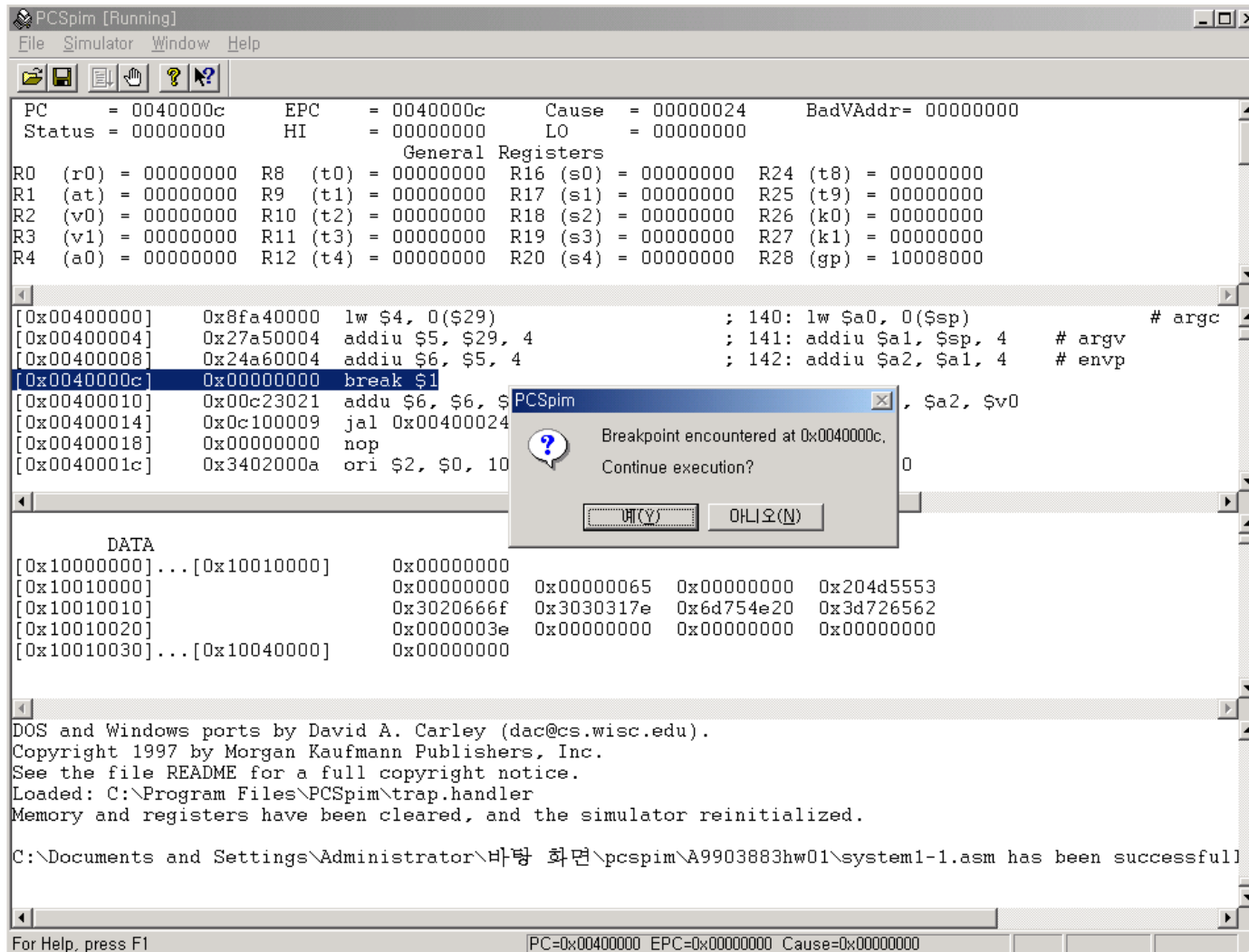


# Function - breakpoint

The image shows the PCSpim simulator window. The top menu bar includes File, Simulator, Window, and Help. Below the menu is a toolbar with icons for file operations and simulation control. The main window is divided into several sections:

- Registers:** Displays the current state of the processor registers. PC = 00400000, EPC = 00000000, Cause = 00000000, BadVAddr = 00000000. Status = 00000000, HI = 00000000, LO = 00000000. General Registers R0 through R28 are listed with their values.
- Assembly Code:** Shows the assembly code being executed. A red box highlights the instruction at address 0x0040000c: `break $1`. The code includes instructions for loading arguments, adding offsets to pointers, and jumping to the `main` function.
- DATA:** Displays the memory layout, showing addresses and corresponding data values.
- Log/Status:** At the bottom, it shows the current PC, EPC, and Cause values, along with a message indicating that the simulator has been reinitialized.

# Function - breakpoint



# Example

## □ Sum

```
.text                                # text section
.globl main                          # call main by SPIM

main: la    $t0, value               # load address 'value' into $t0
      lw    $t1, 0($t0)              # load word 0(value) into $t1
      lw    $t2, 4($t0)              # load word 4(value) into $t2
      lw    $t3, 8($t0)              # load word 8(value) into $t3

Loop: beqz   $t2, End                 # t2가 0이면 End로 가고 아니면 다음으로 내려간다
      addi   $t1, $t1, 1              # t1의 값을 하나씩 증가시킨다
      add    $t3, $t3, $t1            # t3에 t1을 더해 t3에 넣으며 같은 누적한다
      addi   $t2, $t2, -1             # t2의 값을 하나씩 감소시킨다
      j      Loop                    # Loop로 이동시켜 루프를 돌린다

End:  sw     $t3, 8($t0)              # store word $t3 into 8($t0)


      li     $v0, 4
      la     $a0, msg1
      syscall

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

.data                                # data section
value: .word 0, 100, 0               # data for addition
msg1:  .asciiz "SUM of 0~100 Number=>"
```

# Function - Go

---



```
Console
SUM of 0~100 Number=>5050
```

The image shows a standard Windows-style console window. The title bar at the top is blue and contains the word 'Console' next to a small icon. The main area of the window is white and contains a single line of black text: 'SUM of 0~100 Number=>5050'. The window has standard Windows controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side. The bottom of the window shows a greyed-out status bar with navigation arrows.

# Function - Go

[0x00400050]	0x34020004	ori \$2, \$0, 4			
[0x00400054]	0x3c011001	lui \$1, 4097 [msg1]			
[0x00400058]	0x3424000c	ori \$4, \$1, 12 [msg1]			
[0x0040005c]	0x0000000c	syscall			
[0x00400060]	0x34020001	ori \$2, \$0, 1			
[0x00400064]	0x000b2021	addu \$4, \$0, \$11			
[0x00400068]	0x0000000c	syscall			
[0x0040006c]	0x03e00008	jr \$31			

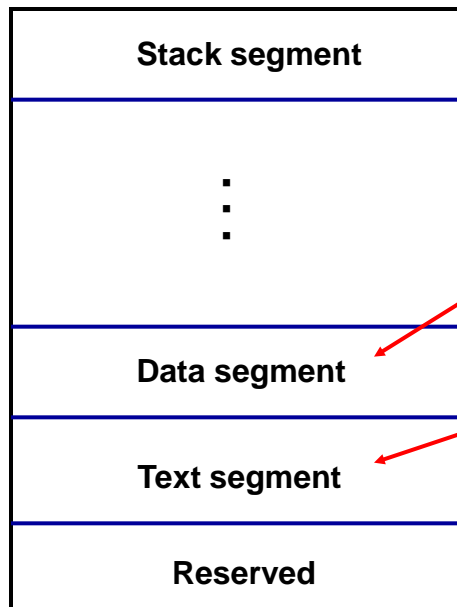
Define upper 2 bytes

Define lower 2 bytes

0x7FFFFFFF

0x10000000

0x00400000



Data that  
Instruction  
operate on

Instruction

# Directives

---

- ❑ **.text** – Indicates that following items are stored in the user text segment
- ❑ **.globl** *sym* – Declare that symbol *sym* is global and can be referenced from other files
- ❑ **.data** – Indicates that following data items are stored in the data segment

# Data Types

---

- ❑ **.word, .half** - 32/16 bit integer
- ❑ **.byte** - 8 bit integer (similar to 'char' type in C)
- ❑ **.ascii, .asciiz** - string (asciiz is null terminated)
  - Strings are enclosed in double-quotes("")
  - Special characters in strings follow the C convention
    - `newline(\n)`, `tab(\t)`, `quote(\")`
- ❑ **.double, .float** - floating point

# System calls

## ❑ System Calls (syscall)

- OS-like services

## ❑ Method

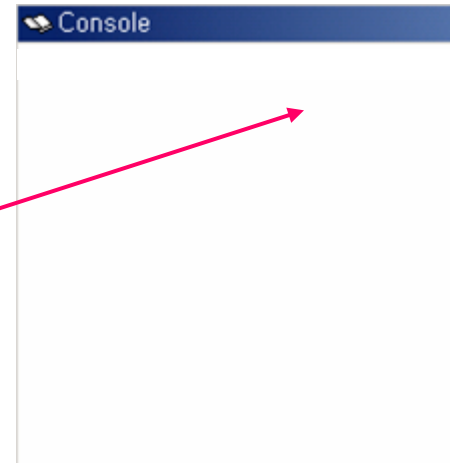
- Load system call code into register \$v0
- Load arguments into registers \$a0...\$a3
- After call, return value is in register \$v0

### Example)

```
li    $v0, 4    # print string
la    $a0, msg1
syscall
```

```
li    $v0, 1    # print integer
move  $a0, $t3
syscall
```

◆ msg1 = "SUM of 0~100 Number=>" , \$t3 = 5050



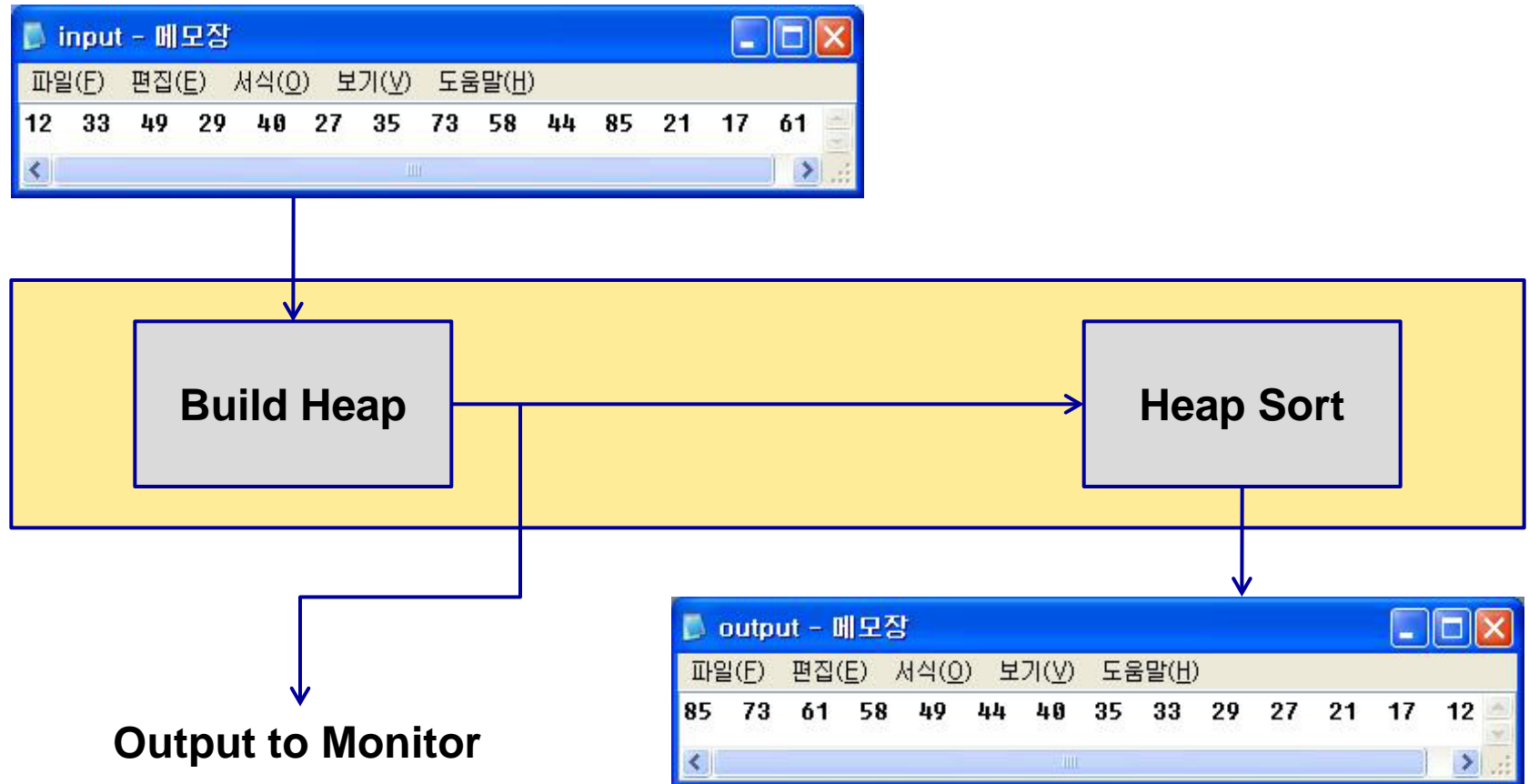


# System calls

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 \$a0)
open	13	\$a0 = filename (string), \$a1 = flags, \$a2 = mode	file descriptor (in \$v0)
read	14	\$a0 = file descriptor, \$a1 = buffer, \$a2 = length	num chars read (in \$v0)
write	15	\$a0 = file descriptor, \$a1 = buffer, \$a2 = length	num chars written (in \$v0)
close	16	\$a0 = file descriptor	
exit2	17	\$a0 = result	

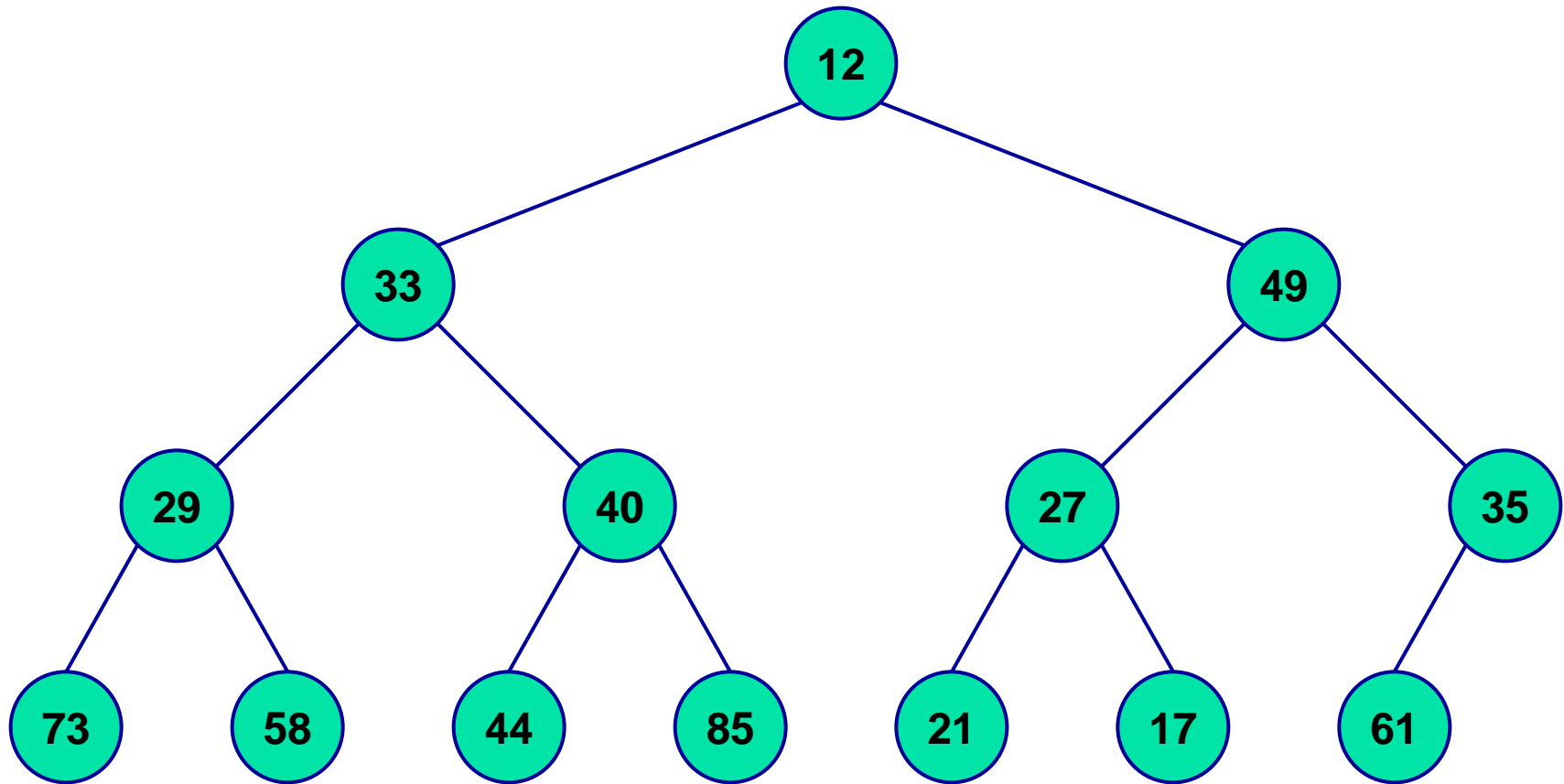
# MIPS Program Assignment

## □ Heap sort



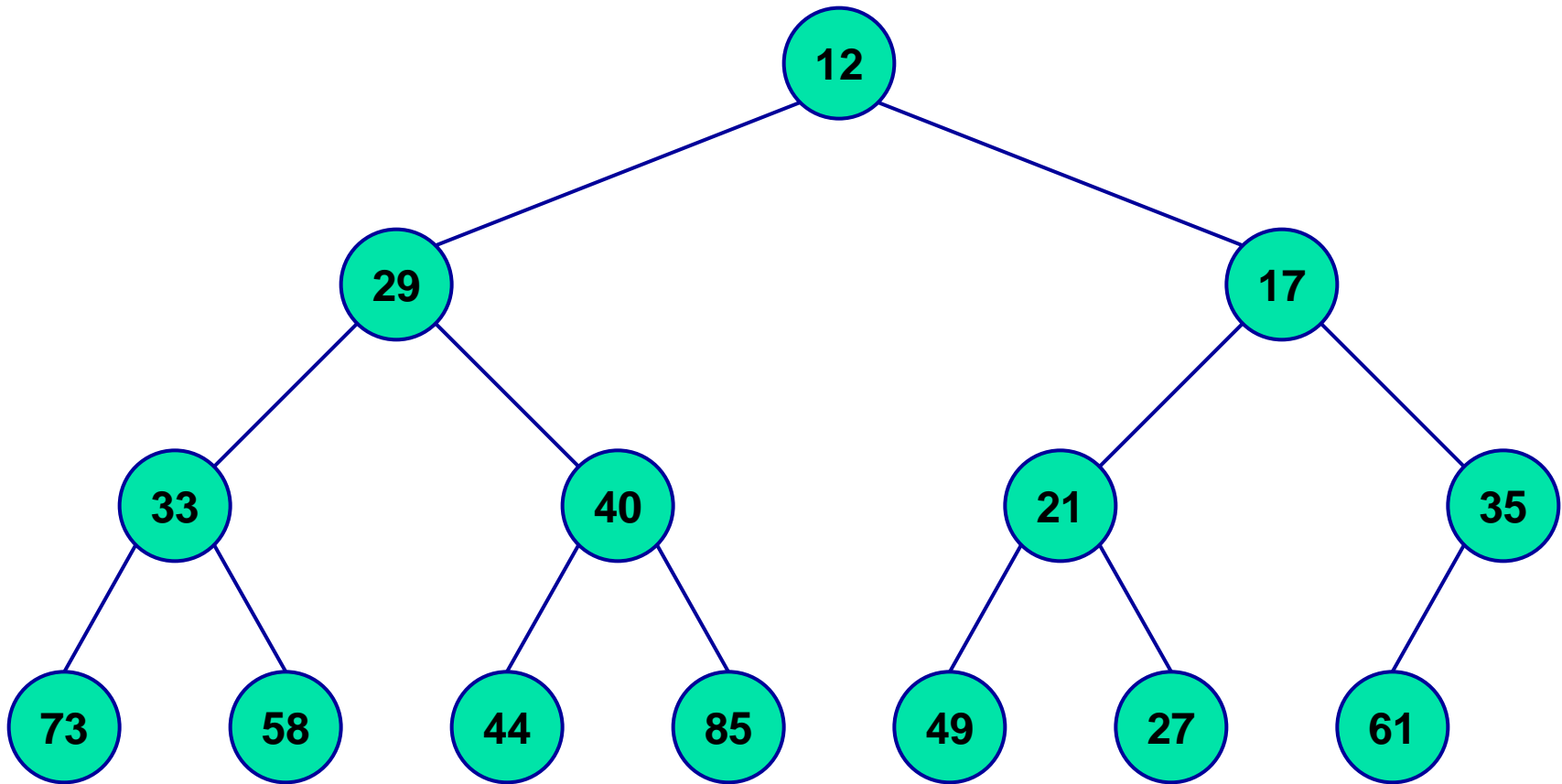
# Sequence

Sequence: {12, 33, 49, 29, 40, 27, 35, 73, 58, 44, 85, 21, 17, 61}



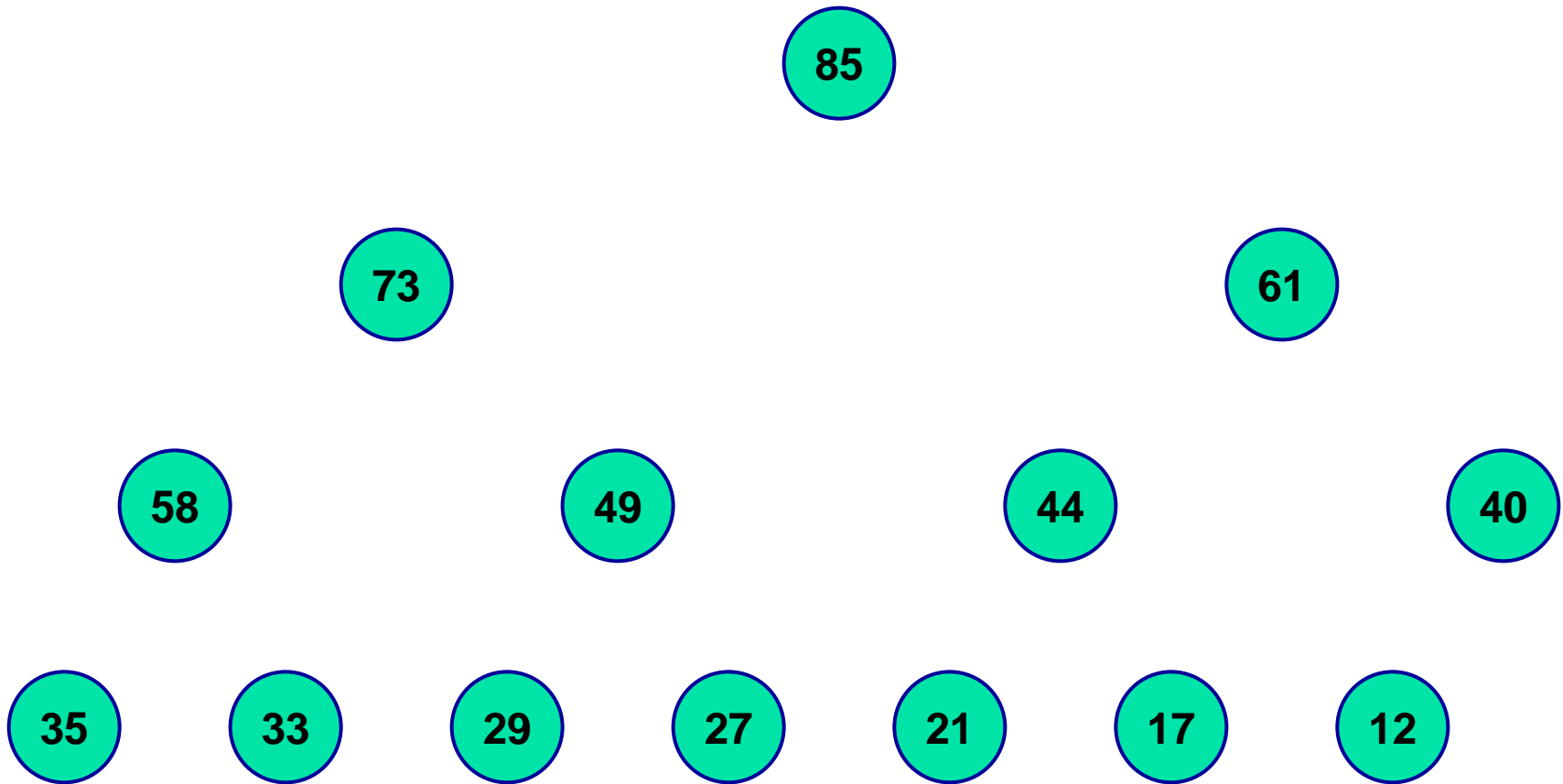
# Build Heap

After Build: {12, 29, 17, 33, 40, 21, 35, 73, 58, 44, 85, 49, 27, 61}



# Heap Sort

After Sort: {85, 73, 61, 58, 49, 44, 40, 35, 33, 29, 27, 21, 17, 12}



# MIPS Program Assignment

---

- ❑ 제출기한 : **2010.11.22(월) 11:59 PM**
- ❑ 제출항목 : **source code**, 보고서
- ❑ 제출방식 : 지정된 **FTP** 에 **upload** (추후공지)
- ❑ 평가: **total 100점, -5%/(1-day delay)**
  - **Build heap: 40점**
  - **Heap sort: 20점**
  - **File I/O: 20점**
  - **Report: 20점**

# Reference

---

- ❑ 문병로, “쉽게 배우는 알고리즘, 관계 중심의 사고법”, 한빛 미디어, **pp.93-99**
- ❑ **C source code on course homepage**