

SPIM & MIPS

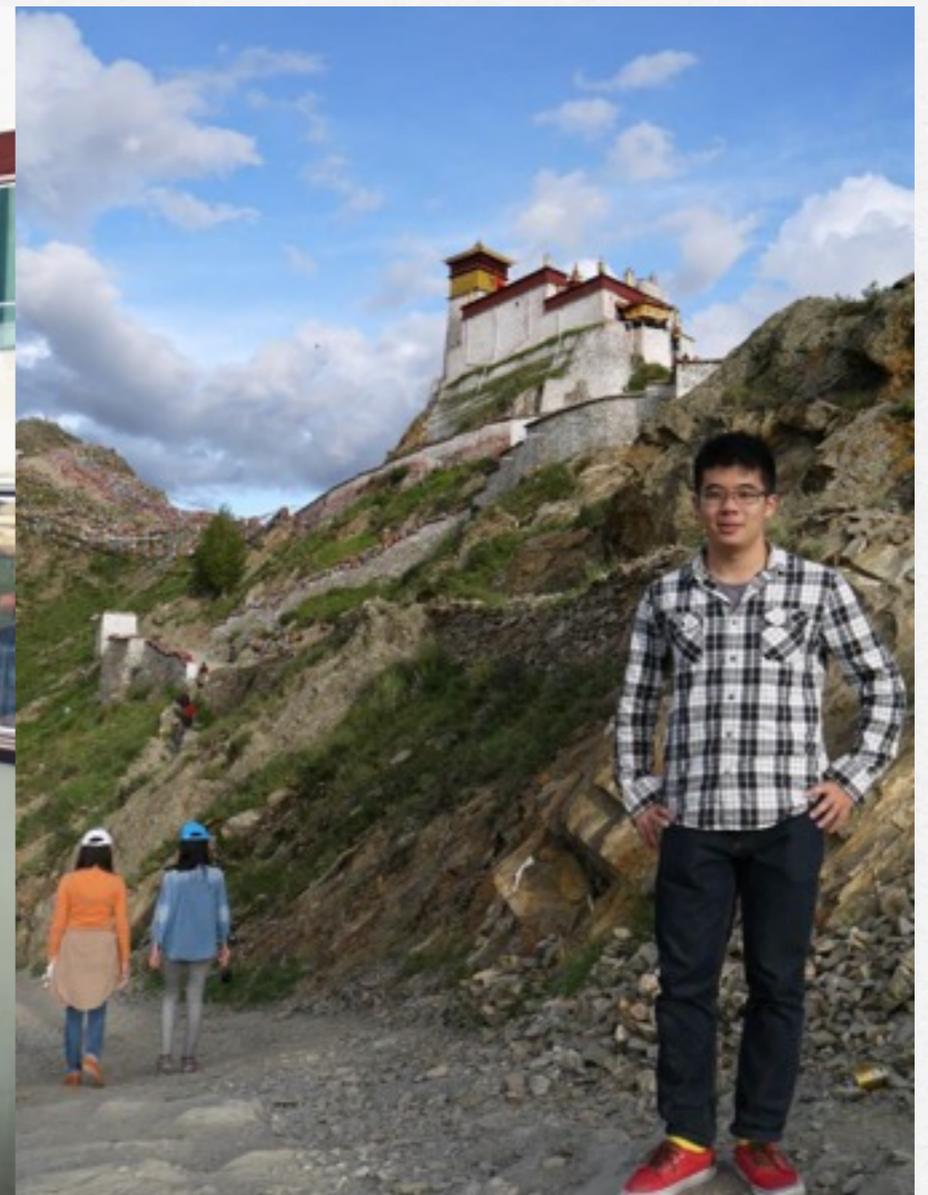
- Department of Information and Management**

Chi-Chi Liao

- 廖以圻
- chichi@cmlab.csie.ntu.edu.tw

TAs

- Chí-Chí
- Andí



Outline

- ❑ **Introduction to Assembly Language**
- ❑ **SPIM – Getting Started**
- ❑ **MIPS – Assembly Language Programming**
- ❑ **Homework 2 – Programming Assignment**

Assembly Language

□ Introduction

Assembly language

- **Assembly language**

Symbolic representation of a computer's binary encoding

Assembler

Translates assembly language into binary instructions



- **Machine code**

Computer's binary encoding

Assembly Language

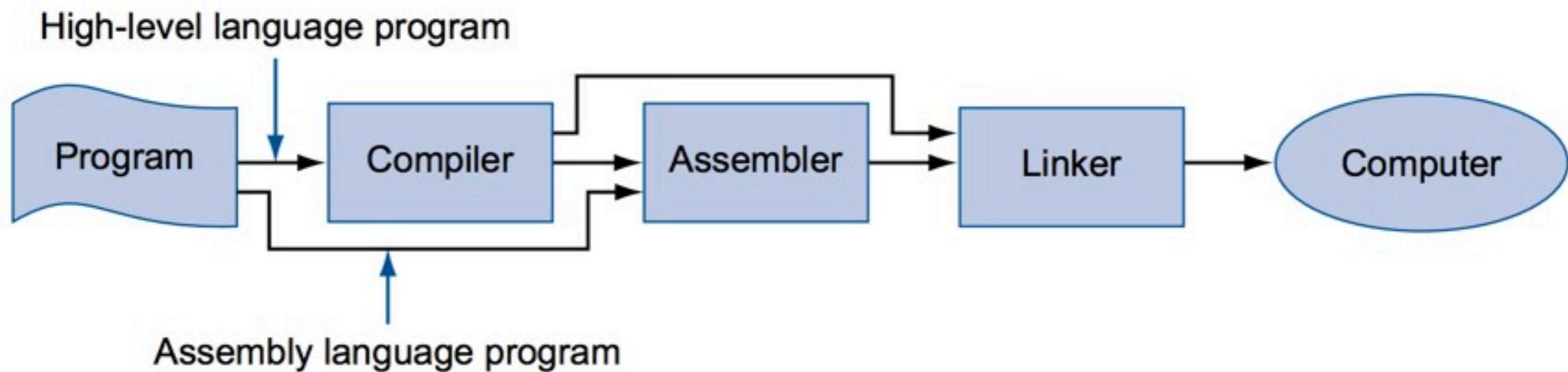


FIGURE A.1.6 Assembly language either is written by a programmer or is the output of a compiler.

Why Assembly

- ❑ **A low level language**
the code and syntax is much closer to the computer's processor
- ❑ **Direct hardware manipulation**
device drivers, low-level embedded systems, and real-time systems
- ❑ **Speed optimization**
performance and efficiency

*To write in assembly is to understand exactly **how** the processor and memory **work together** to "make things happen".*



Sometimes to debug a higher-level language, you have to review the resulting assembly language.

SPIM

- A MIPS32 Simulator**

What is SPIM

- ❑ **MIPS32 Simulator**
reads and executes assembly language program
written for MIPS 32-bit architecture
- ❑ **SPIM does not execute binary
programs**
provides a simple debugger and minimal set of
operating system services
- ❑ **SPIM implements both a terminal**

QtSPIM Installation

SPIM MIPS Simulator

spimsimulator.sourceforge.net

SPIM: A MIPS32 Simulator

James Larus
spim@larusstone.org

Contents

- [Older Versions of SPIM](#)
- [Further Information](#)
- [Changes to SPIM](#)
- [Copyright](#)

Spim is a self-contained simulator that runs MIPS32 programs. It reads and executes assembly language programs written for this processor. *Spim* also provides a simple debugger and minimal set of operating system services. *Spim* does not execute binary (compiled) programs.

Spim implements almost the entire MIPS32 assembler-extended instruction set. (It omits most floating point comparisons and rounding modes and the memory system page tables.) The MIPS architecture has several variants that differ in various ways (e.g., the MIPS64 architecture supports 64-bit integers and addresses), which means that *Spim* will not run programs for all MIPS processors.

Spim comes with complete source code and documentation.

Spim implements both a terminal and windows interfaces. On Microsoft Windows, Linux, and Mac OS X, the *spim* program offers a simple terminal interface and the *QtSpim* program provides the windowing interface. The [older programs xspim and PCSpim](#) provide window interfaces for these systems as well.

[Download SPIM](#)

What's New?

QtSpim is a new user interface for *Spim* built on the [Qt UI framework](#). Qt is cross-platform, so the same user interface and same code will run on Windows, Linux, and Mac OS X (yeah!). Moreover, the interface is clean and up-to-date (unlike the archaic X windows interface).

Spim has moved to [SourceForge!](#) The source code for all version of *Spim* are in an SVN repository and compiled version are available for download. There is also a bug tracker and discussion forum. *Spim* is an open source project, so please join in and contribute.

QtSPIM Installation

spim mips simulator

Brought to you by: jameslarus

Summary Files Reviews Support Wiki Code Tickets ▾

Looking for the latest version? [Download QtSpim_9.1_mac.dmg \(16.0 MB\)](#)

Home

Name	Modified	Size	Downloads / Week
QtSpim_9.1.13_mac.mpkg.zip	2014-02-05	33.1 MB	281
QtSpim_9.1.12_Windows.exe	2013-12-17	1.5 MB	1,446
qtspim_9.1.12_linux32.deb	2013-12-14	1.1 MB	61
qtspim_9.1.12_linux64.deb	2013-12-14	1.1 MB	222
qtspim_9.1.9_linux64.deb	2013-01-23	1.1 MB	11
qtspim_9.1.9_linux32.deb	2013-01-23	1.1 MB	3
PCSpim_9.1.9.zip	2013-01-20	5.7 MB	269
QtSpim_9.1.9_Windows.zip	2013-01-20	14.1 MB	49
QtSpim_9.1_mac.dmg	2012-12-13	16.0 MB	145
QtSpim_9.1.7_Windows.zip	2012-02-18	21.0 MB	56
qtspim_9.1.6_linux32.deb	2012-02-06	1.1 MB	7

Windows

Linux 32/64-bit

Mac OS X

QtSPIM Screenshot

The screenshot displays the QtSPIM debugger interface. The main window is titled "QtSpim" and features a menu bar with options like File, Edit, View, and Help. Below the menu bar, there are tabs for "FP Regs" and "Int Regs [16]". The "FP Regs" tab is active, showing a list of floating-point registers (FP0 to FP31) and their values. The "Int Regs [16]" tab is also visible, showing integer registers. The main display area is divided into two panes: "Text" and "Data". The "Text" pane shows the user data segment and user stack, with the text "Hello IM105!..." visible. The "Data" pane shows the kernel data segment. A "Console" window is overlaid on the main display, showing the output "Hllo World! This is Chi-Chi." The bottom of the window contains a status bar with the following text: "Memory and registers cleared", "Loaded: /var/folders/n_/18fx38r92dg3z4nsxl9kt_g80000gn/T/qt_temp.nN1317", "SPIM Version 9.1.7 of February 12, 2012", "Copyright 1990-2012, James R. Larus.", "All Rights Reserved.", "SPIM is distributed under a BSD license.", and "See the file README for a full copyright notice."

Ref. of SPIM

- ❑ **Official website of SPIM**
<http://spimsimulator.sourceforge.net/>
- ❑ **Assemblers, Linkers, and SPIM Simulator**
http://spimsimulator.sourceforge.net/HP_AppA.pdf
- ❑ **MIPS Instruction Reference**
<http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html>

MIPS

- ❑ **Microprocessor without Interlocked Pipeline Stages**

MIPS memory layout

- ❑ **MIPS 32-bit CPU (all registers are 32 bits wide)**
accessible memory range: 0x00000000–0xFFFFFFFF
- ❑ **Memory holds both instructions (text) and data**
If a program is loaded into SPIM, its .text segment is automatically placed at 0x00400000, its .data segment at 0x10000000

MIPS Assembly

Operation Code(Opcode)

- **Arithmetic Instructions**
add, sub, addi, addu, addiu, subu
- **Data Transfer Instructions**
lw, sw, lbu, sb, lui
- **Logic Instructions**
beq, bne, slt, slti, sltu
- **Branch and Jump- Related Instructions**
j , jr, jal

MIPS Assembly

MIPS Registers and Usage Convention

\$zero	constant 0
\$v0, \$v1	expression of a function
\$a0 ~ \$a3	argument 1~4
\$t0 ~ \$t9	temporary registers
\$s0 ~ \$s7	save registers
\$sp	stack pointer
\$fp	frame pointer
\$ra	return address
...	...

MIPS Assembly

Some data types in MIPS

.word, .half

32/16 bit integer

.byte

8 bit integer

.ascii, .asciiz

string

.double, .float

floating point

Assembler Syntax

- ❑ **Comment : (#)**
Everything from the sharp sign to the end of the line is ignored
- ❑ **Identifier : (A sequence of alphanumeric characters, _ , and .)**
Identifier are a sequence of alphanumeric characters, underscores (_), and dots (.) that do not begin with a number
- ❑ **Instruction Opcode**
Instruction opcodes are reserved words that are not valid identifiers
- ❑ **Label**
Labels are declared by putting them at the beginning of a line followed by a colon.

MIPS – Hello World

C

```
int main()
{
    printf("Hello World\n");
    return 0;
}
```

MIPS

```
.data
Mystr: .asciiz "Hello World\n"

.text
main:
    li $v0, 4
    la $a0, Mystr
    syscall
    li $v0, 10
    syscall
```

MIPS – Hello World

Put static data here

Put your code here

MIPS

`.data`

```
Mystr: .asciiz "Hello World\n"
```

```
MyInteger: .word 100
```

```
MyArray: .word 1, 2, 3
```

`.text`

```
main:
```

```
    li $v0, 4
```

```
    la $a0, Mystr
```

```
    syscall
```

```
    li $v0, 10
```

```
    syscall
```

MIPS – Hello World

Put static data here

MIPS

`.data`

```
Mystr: .asciiz "Hello World\n"  
MyInteger: .word 100  
MyArray: .word 1, 2, 3
```

Put your code here

`.text`

```
main:  
    # do anything you want  
    ...  
    # end of the program  
    li $v0, 10  
    syscall
```

MIPS System Calls

- **SPIM provides a small set of **operating-system-like** services through the system call instruction**
- **A program loads the system call code into register \$v0 and arguments into registers \$a0-\$a3
(or \$f12 for floating-point values)**
- **System calls that return values put their results in register \$v0 (or \$f0 for floating-point results)**

MIPS 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 \$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 A.9.1 System services.

MIPS System Calls

MIPS

.data

```
str: .asciiz "The answer = "
```

.text

```
main:
```

```
    li $v0, 4
```

```
    la $a0, str
```

```
    syscall
```

```
    li $v0, 1
```

```
    li $a0, 5
```

```
    syscall
```

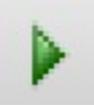
```
    li $v0, 10
```

```
    syscall
```

Service	System call code
print_int	1
print_float	2
print_double	3
print_string	4
read_int	5
read_float	6
read_double	7
read_string	8
sbrk	9
exit	10
print_char	11
read_char	12
open	13
read	14
write	15
close	16
exit2	17

FIGURE A.9.1 System services.

Execute Program in

1. Write your own assembly program, and save it as .s file
2. Simulator - Reinitialize Simulator 
3. Open your .s file 
4. Simulator - Clear Registers 
5. Simulator - Run / Continue 

Homework 2

- **Programming Assignment**

Homework 2

- This is an **individual assignment**
- **Plagiarism will be heavily punished**
- Write the following three programs in MIPS assembly language. (Must run correctly on SPIM)

Area of a Triangle

Tower of Hanoi

Bubble Sort

- One bonus program : **Variation of Fibonacci**

Documentation (20%)

- ❑ **Detailed documentation for each program is required**
- ❑ **The following parts must be included:**
 - ❑ **Your name, student ID, and email address**
 - ❑ **Explanation of the design or the flow of each program**
 - ❑ **What you've learnt from writing the programs**
- ❑ **Problems or difficulties you've encountered during writing the programs are nice to be included in the document**

Area of a Triangle

- **Introduction :**

- **A triangle is composed of three independent points. We will give you **three points on a 2D surface**, and you should calculate the area of this triangle. It's ok to be zero, but **not negative number.****

Area of A Triangle

Your file should work like this:

Please type 6 integers, x_1 , y_1 , x_2 , y_2 , x_3 , y_3 , and each with the enter key:

x_1 (input)

y_1 (input)

x_2 (input)

y_2 (input)

x_3 (input)

y_3 (input)

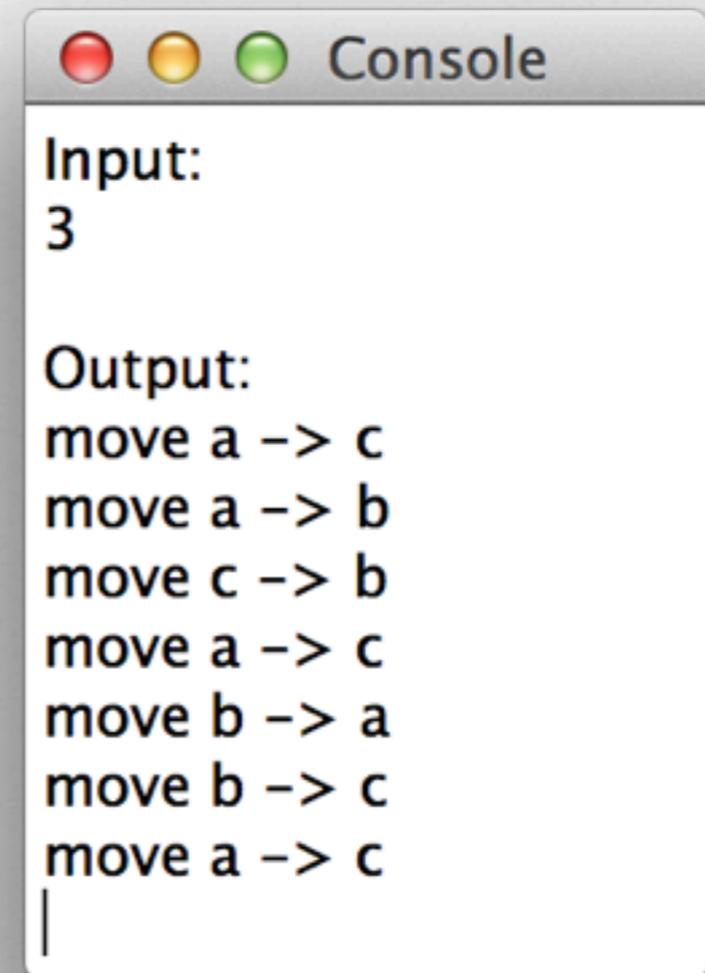
The area is:(Your answer here.)

Requirements:

1. Print the correct answer
2. The file name is **Area.s**

Tower of Hanoi

- A hanoi tower with 3 rods A,B,C and n disks
Move all the disks **from A to C**
- **Input :**
positive integer n (disks), $1 \leq n \leq 5$
- **Output :**
Print all the steps
- **Requirements:**
 1. Print the correct steps
 2. The file name is **Hanoi.s**



```
Console
Input:
3

Output:
move a -> c
move a -> b
move c -> b
move a -> c
move b -> a
move b -> c
move a -> c
|
```

Bubble Sort

- **Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order.**

Bubble Sort

Input :

n positive integers, where $n < 8$

Output :

Sorting $n_1, n_2, n_3, n_4, n_5 \dots$ in ascending order

Requirements:

1. Print the correct answer
2. The file name is **Sorting.s**

Bonus:

Variation of Fibonacci

- **Def.**
- **$F(0) = 0$**
 $F(1) = 1$
 $F(2) = 2$
 $F(n) = F(n-1) + F(n-3)$, if $n > 2$
- **So it would be: 0, 1, 2, 2, 3, 5, 7, 10, 15.....**

Bonus:

Variation of Fibonacci

Input :

positive integer n

Output :

F(n), which it is a Fibonacci number

Requirements:

1. Print the correct answer
2. The file name is **Fibonacci.s**

Submission

- **Deadline : 11:59 PM, Monday, Oct. 27, 2014**
- **You must submit at least the following files:**
 - **Area.s**
 - **Hanoi.s**
 - **Sorting.s**
 - **Fibonacci.s (optional)**
 - **(Your student id)_hw2_document.pdf**
- **Please put all your files in a directory named by your student id in **lowercase**, and then compress it into one zipped file.**
The attach filename should be like b02xxxxxx.zip.
- **Email your zipped file to TA**
chichi@cmlab.csie.ntu.edu.tw

Grading Guidelines

Description	For Each Problem
Program runs without error messages	10%
Program executes correctly	60%
Documentation and description	20%
Implementation Detail	10%

Deadline

- **Late submission policy**
 - **10% off from your total score each day**

Contact Information

□ TA Hours @ 管院一館五樓 503-C

Chi-Chi Liao (廖以圻) Thur. 14:00 ~ 15:00

Han-Chih Kuo (郭瀚智) Mon. 14:00 ~ 15:00

□ Contact us if you have any problem

Chi-Chi: chichi@cmlab.csie.ntu.edu.tw

Andi: andikan@cmlab.csie.ntu.edu.tw

**Thank You for
Your Attention**