SPIM & MIPS

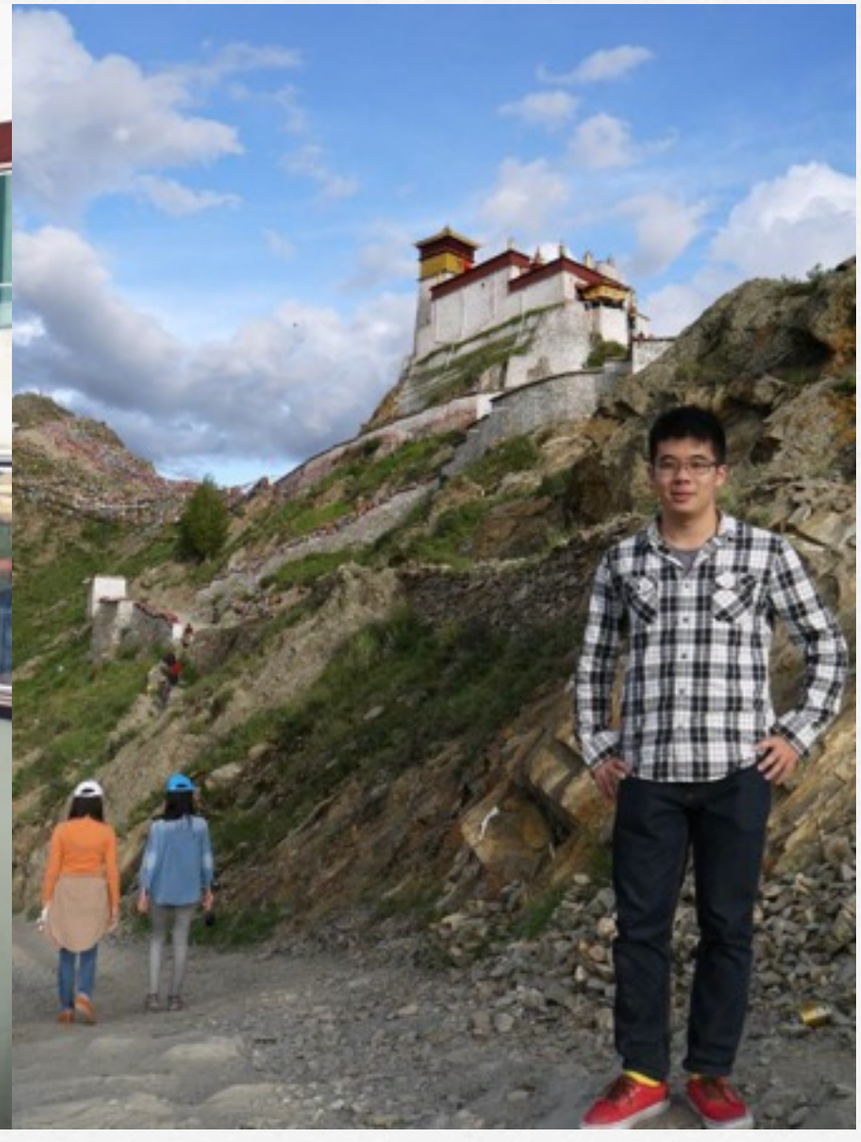
- Department of Information and Management

Chi-Chi Liao

- 廖以圻
- chichi@cmlab.csie.ntu.edu.tw

TAs

- Chi-Chi
- Andi



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

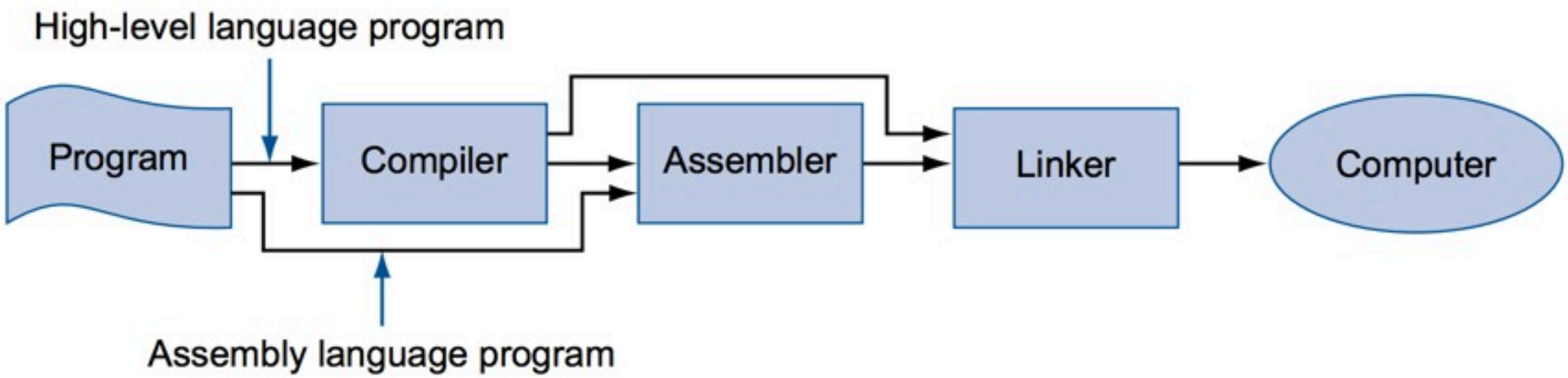


FIGURE A.1.6 Assembly language either is written by a programmer or is the output of a compiler.

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.

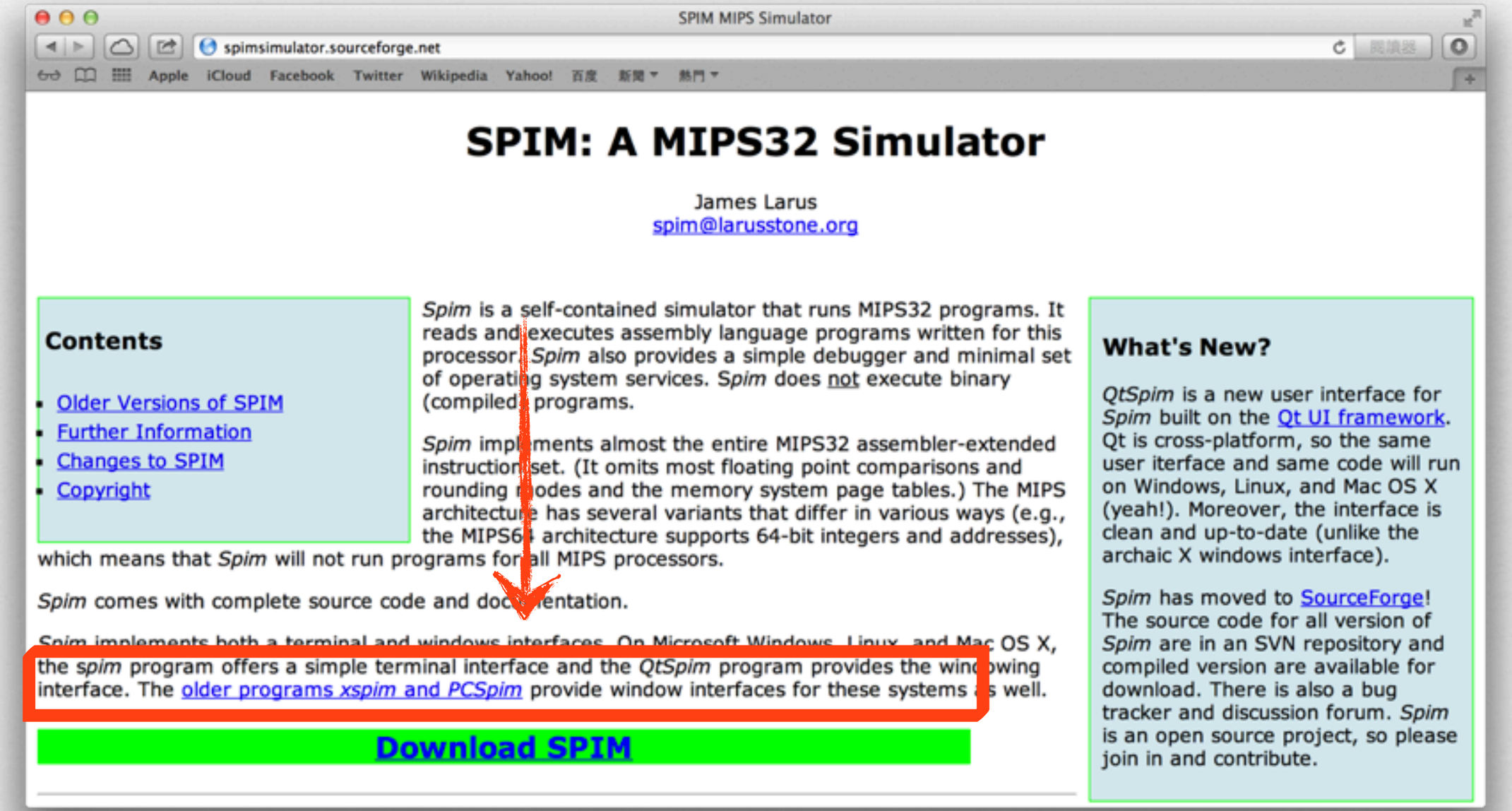
SPIM

- A MIPS32 Simulator

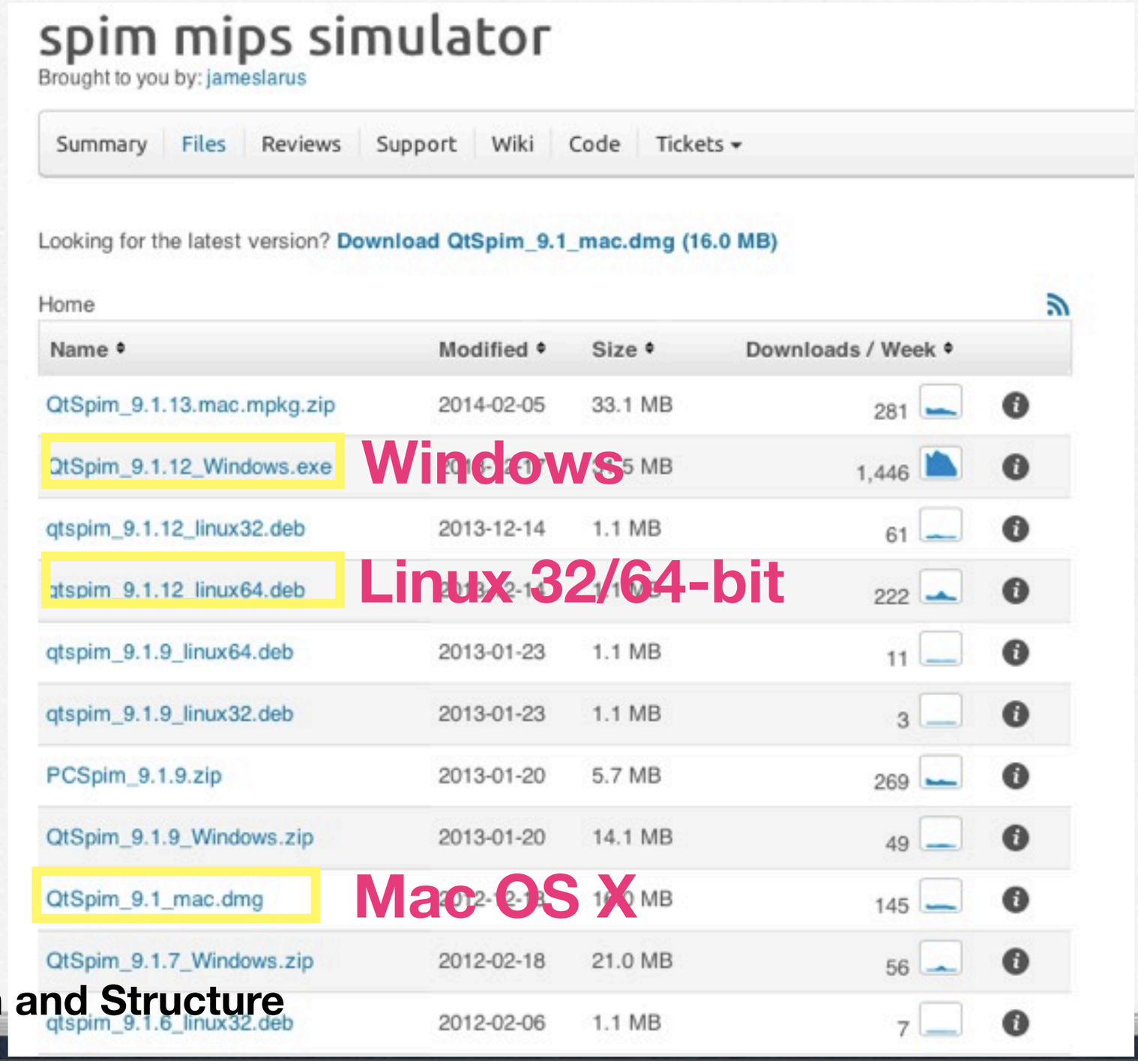
What is SPIM

- **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



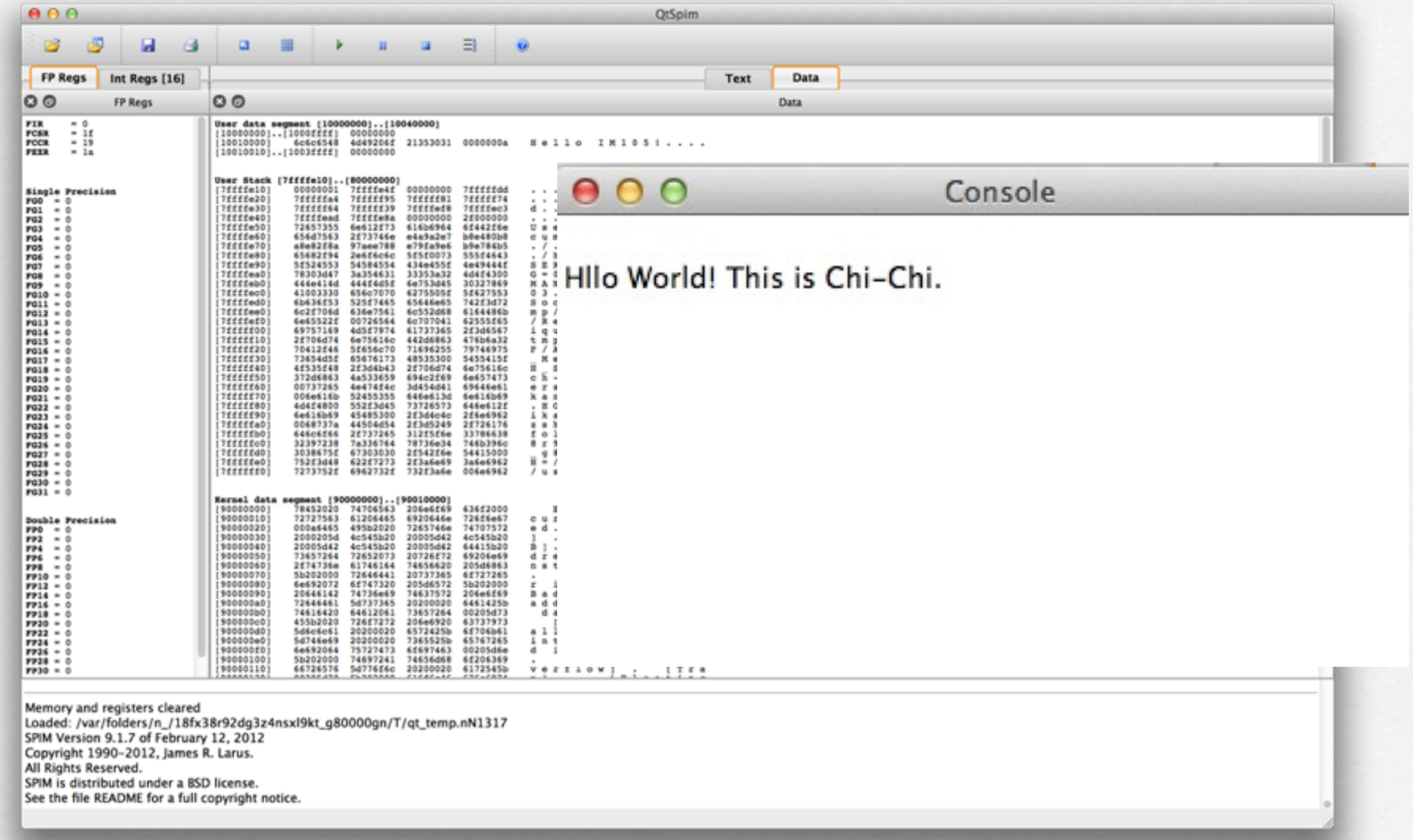
QtSPIM Installation



Computer Organization and Structure

2013

QtSPIM Screenshot



Computer Organization and Structure

2013

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	MIPS

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

```
 .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

MIPS

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

Put your code here

```
.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

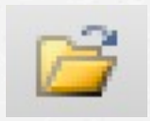
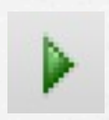
```
.data
str: .asciiz “The answer = ”

.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, x1, y1, x2, y2, x3, y3, and each with the enter key:

x1 (input)
y1 (input)
x2 (input)
y2 (input)
x3 (input)
y3 (input)
The area is:(Your answer here.)

Requirements:

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

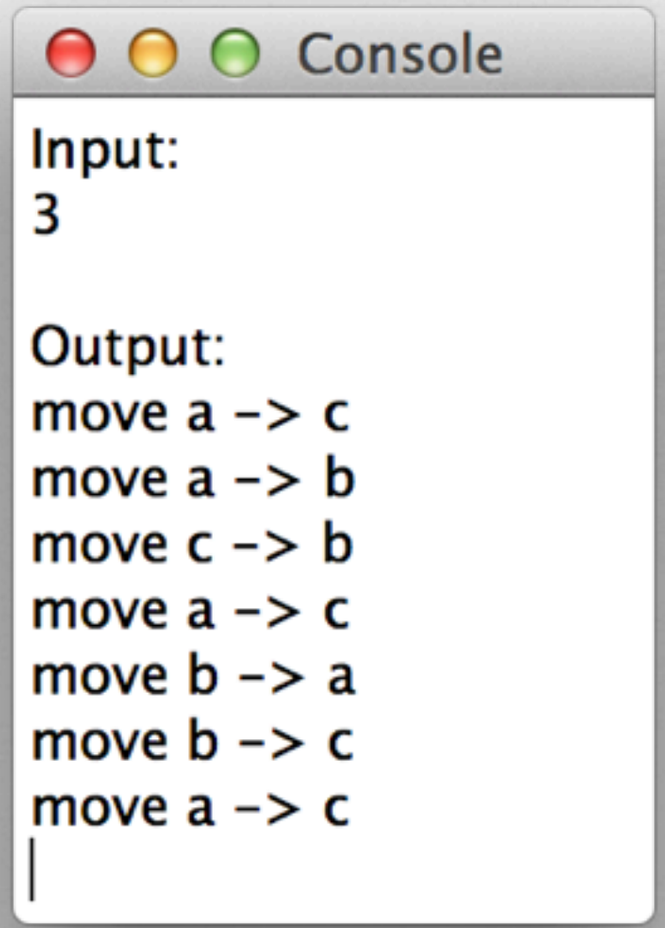
Tower of Hanoi

- 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 ≤ n ≤ 5

- **Output :**
  Print all the steps

- **Requirements:**
  1. Print the correct steps
  2. The file name is Hanoi.s



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 n1, n2, n3, n4, n5... 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