SPIM & MIPS

- Department of Information and Management

Computer Organization and Structure
2013
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

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 nodes 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 windows 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

Looking for the latest version? Download QtSpim 9.1.13.macosx.zip (16.0 MB)

<table>
<thead>
<tr>
<th>Name</th>
<th>Modified</th>
<th>Size</th>
<th>Downloads / Week</th>
</tr>
</thead>
<tbody>
<tr>
<td>QtSpim 9.1.13.macosx.zip</td>
<td>2014-02-05</td>
<td>33.1 MB</td>
<td>281</td>
</tr>
<tr>
<td>QtSpim 9.1.12.Windows.exe</td>
<td>2014-02-05</td>
<td>33.1 MB</td>
<td>1,446</td>
</tr>
<tr>
<td>QtSpim 9.1.12.Windows.exe</td>
<td>2013-12-14</td>
<td>13.5 MB</td>
<td></td>
</tr>
<tr>
<td>QtSpim 9.1.12.linux32.deb</td>
<td>2013-12-14</td>
<td>1.1 MB</td>
<td>61</td>
</tr>
<tr>
<td>QtSpim 9.1.12.linux64.deb</td>
<td>2013-12-14</td>
<td>1.1 MB</td>
<td>222</td>
</tr>
<tr>
<td>QtSpim 9.1.9.linux32.deb</td>
<td>2013-01-23</td>
<td>1.1 MB</td>
<td>3</td>
</tr>
<tr>
<td>QtSpim 9.1.9.linux64.deb</td>
<td>2013-01-23</td>
<td>1.1 MB</td>
<td>11</td>
</tr>
<tr>
<td>QtSpim 9.1.9.Windows.zip</td>
<td>2013-01-20</td>
<td>5.7 MB</td>
<td>269</td>
</tr>
<tr>
<td>QtSpim 9.1.6.linux32.deb</td>
<td>2012-02-18</td>
<td>21.0 MB</td>
<td>56</td>
</tr>
<tr>
<td>QtSpim 9.1.6.linux64.deb</td>
<td>2012-02-18</td>
<td>1.1 MB</td>
<td>7</td>
</tr>
</tbody>
</table>
QtSPIM Screenshot

Hello World! This is Chi-Chi.
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 Registers and Usage Convention

<table>
<thead>
<tr>
<th>Register</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$zero</td>
<td>constant 0</td>
</tr>
<tr>
<td>$v0, $v1</td>
<td>expression of a function</td>
</tr>
<tr>
<td>$a0 ~ $a3</td>
<td>argument 1~4</td>
</tr>
<tr>
<td>$t0 ~ $t9</td>
<td>temporary registers</td>
</tr>
<tr>
<td>$s0 ~ $s7</td>
<td>save registers</td>
</tr>
<tr>
<td>$sp</td>
<td>stack pointer</td>
</tr>
<tr>
<td>$fp</td>
<td>frame pointer</td>
</tr>
<tr>
<td>$ra</td>
<td>return address</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**MIPS Assembly**

**Some data types in MIPS**

<table>
<thead>
<tr>
<th></th>
<th>32/16 bit integer</th>
<th>8 bit integer</th>
<th>string</th>
<th>floating point</th>
</tr>
</thead>
<tbody>
<tr>
<td>.word, .half</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.ascii, .asciiz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.double, .float</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.
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
MIPS – Hello World

Put static data 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

Put your code here
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

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

**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

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 \( n_1, n_2, n_3, n_4, n_5 \ldots \) 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

<table>
<thead>
<tr>
<th>Description</th>
<th>For Each Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program runs without error messages</td>
<td>10%</td>
</tr>
<tr>
<td>Program executes correctly</td>
<td>60%</td>
</tr>
<tr>
<td>Documentation and description</td>
<td>20%</td>
</tr>
<tr>
<td>Implementation Detail</td>
<td>10%</td>
</tr>
</tbody>
</table>
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) Thu. 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