Computer Organization and Structure 2012

SPIM & MIPS Programming

Department of Information Management
National Taiwan University

Outline

- Introduction to Assembly Language
- SPIM – Getting Started
- MIPS – Assembly Language Programming
- Homework 2 (Programming Assignment)

Assembly Language

Machine language

- Computer's binary encoding

Assembly language

- Symbolic representation of a computer's binary encoding

Assembler

- Translates assembly language into binary instructions

What is SPIM?

MIPS32 simulator that reads and executes assembly language program written for SPIM.

SPIM does NOT execute binary program.

References of SPIM

- Official website of SPIM: http://spimsimulator.sourceforge.net/
- Assemblers, Linkers, and the SPIM Simulator:http://pages.cs.wisc.edu/~larus/HP_AppA.pdf
- MIPS Instruction Reference:http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html

QtSPIM - Installation

SPIM: A MIPS32 Simulator

James Larus
spim@larusstone.org

Contents

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 iterface 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

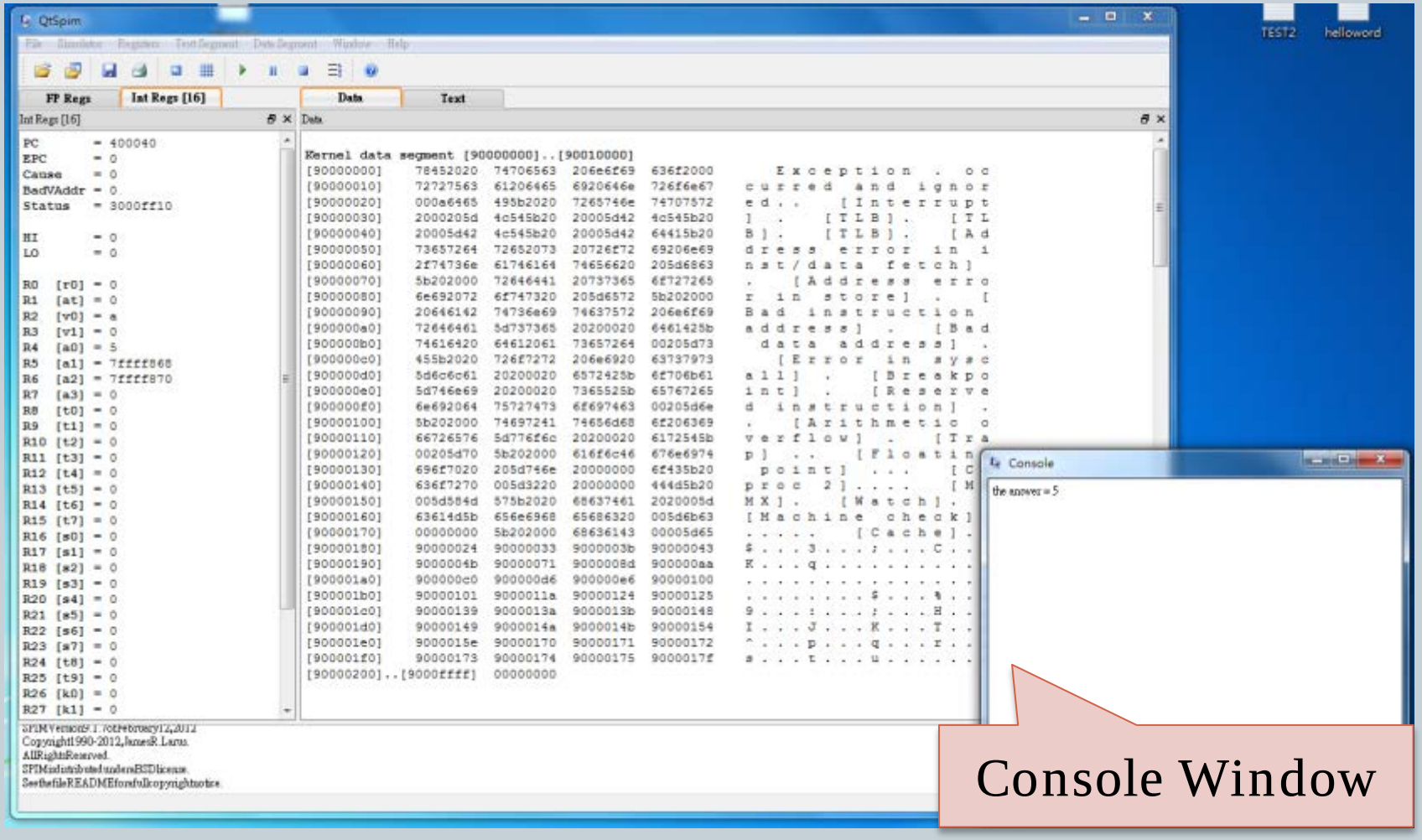
Home

Name	Modified	Size	Downloads
QtSpim_9.1.7_Windows.zip		MB	774
qtspim_9.1.6_linux32.deb	2012-02-06	1.1 MB	69
qtspim_9.1.6_linux64.deb	2012-02-06	1.1 MB	54
QtSpim_9.1.5_Windows.zip	2012-01-03	21.0 MB	9
qtspim_9.1.5_linux64.deb	2012-01-03	1.1 MB	5
qtspim_9.1.5_linux32.deb	2012-01-03	1.1 MB	10
PCSpim_9.1.4.zip	2011-09-04	5.7 MB	163
readme.txt	2011-07-27	686 Bytes	18
QtSpim_9.0.3_mac.dmg		MB	114
Totals: 9 Items		69.7 MB	1,216

Windows

Mac

QtSPIM - Screenshot



MIPS Assembly Language

Operation Code (Opcode)

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

MIPS Assembly Language

- **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 Language

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, underbars (_), 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.

Example – Hello World

C vs MIPS

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

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

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

Example – Hello World

MIPS Architecture

Put Static Data Here

Put Your Code Here

```
.data
Mystr:     .asciiz "Hello World¥n"
Yourint:   .word 75
Hisarray:  .word 100, 100, 100
           .word 20 , 40 , 60
           .word 1  , 2  , 3
.text
......
```

Example – Hello World

MIPS Architecture

Put Static Data Here

Put Your Code Here

```
.data

.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 $v0 services.

MIPS System Calls - Example

Output: "the answer = 5"

```
.data
    str:
    .asciiz "the answer = "
.text
    li $v0, 4          # system call code for print_str
    la $a0, str        # address of string to print
    syscall            # print the string
    li $v0, 1          # system call code for print_int
    li $a0, 5          # integer to print
    syscall            # print it
    li $v0, 10         #system call code for exit
    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.

How to Execute My Program Using QtSPIM?

Write your own assembly program, and save it as .s file

Simulator → Reinitialize Simulator

Open your .s file

Simulator → Clear Registers

Simulator → Run / Continue

Homework

Homework

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

Find out all prime numbers

Mutual prime

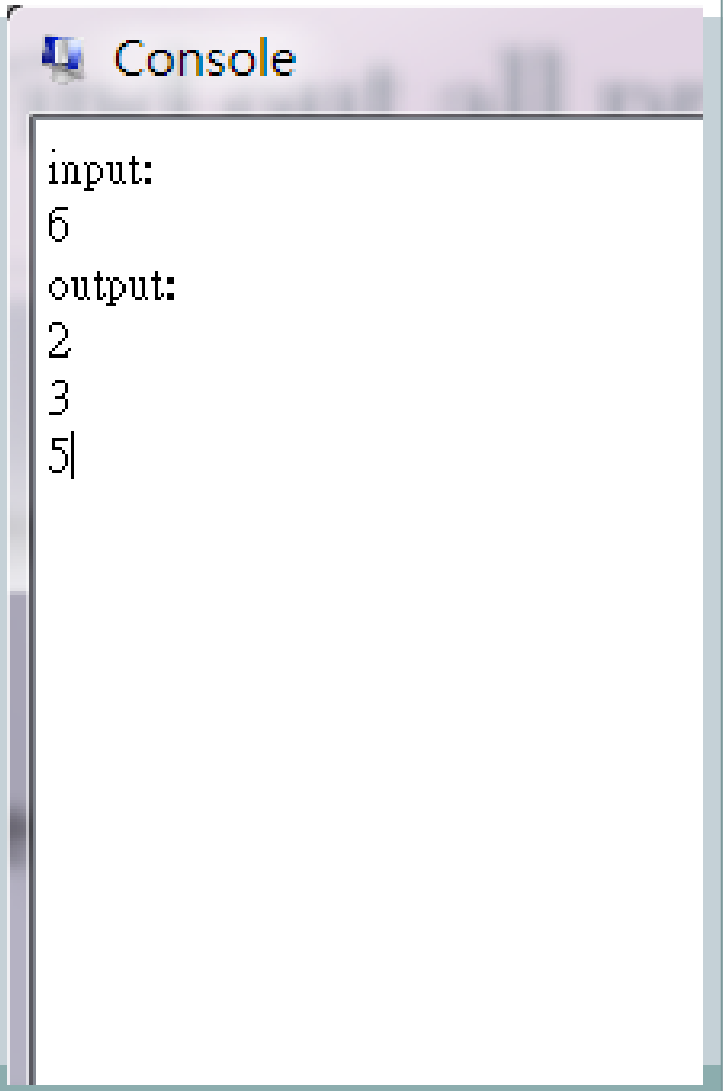
Tower of Hanoi

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

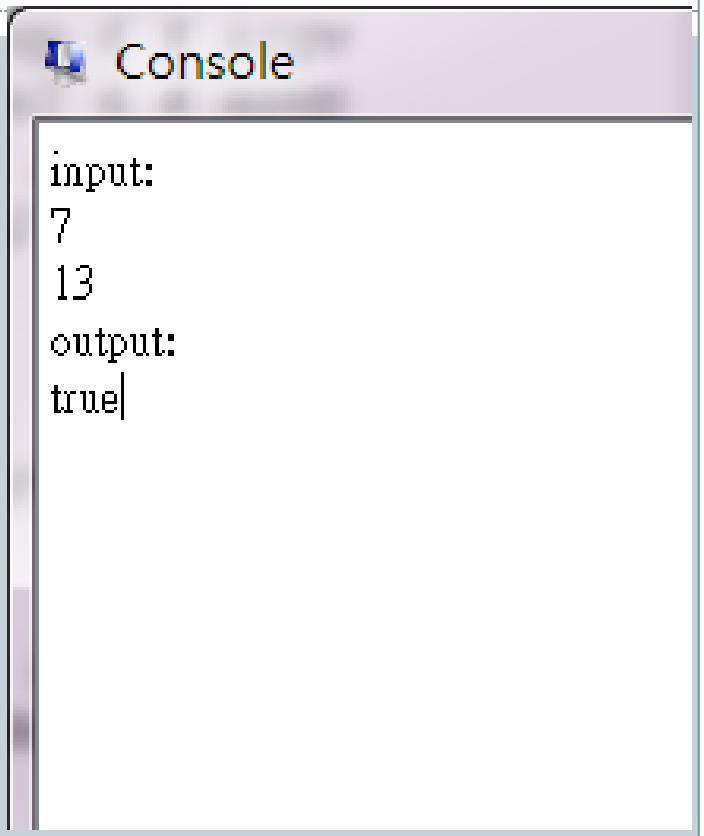
Problem 1. Find out all prime numbers

- **Input:**
  - a positive integer n>1
- **output:**
  - All prime numbers which is smaller than n
- **Requirements:**
  - Print the correct answer.
  - Can do many calculations iteratively
  - The file name is **FindPrime.s**



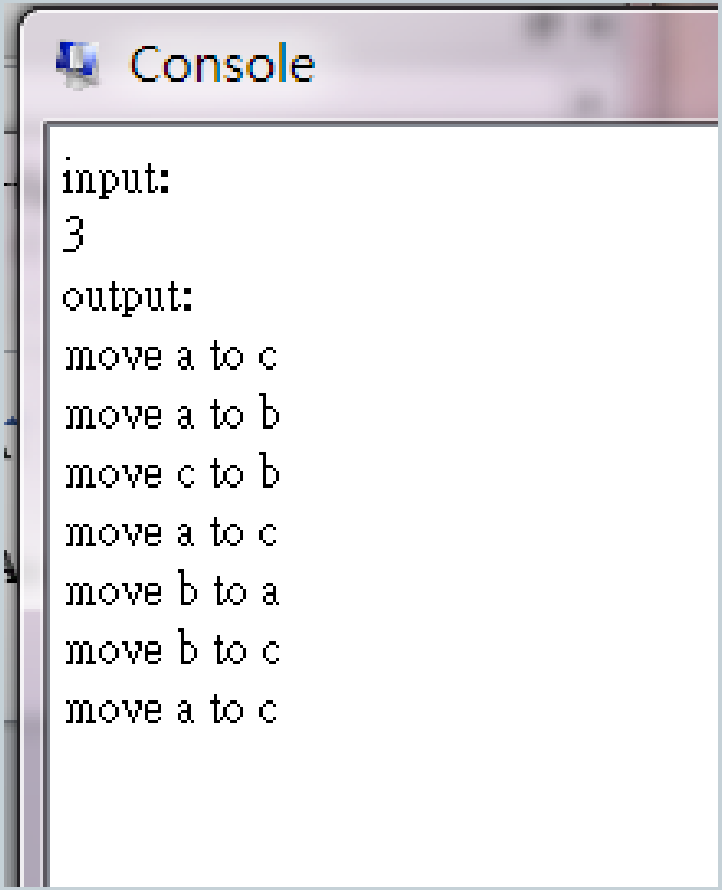
Problem 2: Mutual prime

- **Input:**
  - two positive integers x,y>1
- **output:**
  - If gcd(x, y)=1 ,output True
  - otherwise output False
- **Requirements:**
  - Print the correct answer.
  - Can do many calculations iteratively
  - The file name is **MutualPrime.s**



Problem 3:Tower of Hanoi

- **A hanoi tower with 3 rods A,B,C and n disks**
- **Move all the disk from A to C**
- **Input:**
  - a positive integers n, 1<n≤5
- **Output:**
  - Print all the steps
- **Requirements:**
  - Print the correct step.
  - The file name is **Hanoi.s**



Submission

- Deadline:
- You must submit at least the following files:
  - **FindPrime.s**
  - **MutualPrime.s**
  - **Hanoi.s**
  - **(Your student id)_hwX_document.doc/pdf/docx**
- Please put all your files in a directory named by your student id in uppercase, and then compress it into one zipped file. (e.g. zip/tgz ...)
- The file name should be like B00xxxxxx.zip.
- Email your zipped file to TA

Grading Guidelines

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

Contact Information

- TA Hours @ 管五 503-C
  - Chien-Wen Jung(榮芊雯) Wed. 15:00~17:00
  - Hsiao-Ching You (尤筱青) Wed. 13:00~15:00
  - Shao-Chi Chen (陳少祁) Fri. 10:00~12:00
- Contact us if you have any problem
  - 榮芊雯 winnie87@cmlab.csie.ntu.edu.tw
  - 尤筱青 hcyou@cmlab.csie.ntu.edu.tw
  - 陳少祁auberon@cmlab.csie.ntu.edu.tw

Deadline

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

Thank you for Your Attention

ANY QUESTIONS?