SPIM & MIPS

Department of Information and Management

Ming-Shiuan Chen

Computer Organization and Structure 2015

TAs

TA

- Name : 陳明軒 (包子)

  Ming-Shiuan Chen

- Gender : Male

- Email :

  intere2960@cmlab.csie.ntu.edu.tw

TA

- Name : 廖以圻 (Chi-Chi)

  Chi-Chi Lao

- Gender : Male

- Email :

  chichi@cmlab.csie.ntu.edu.tw

Outline

- Assembly Language
- SPIM
- MIPS
- Homework 2

Assembly Language

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

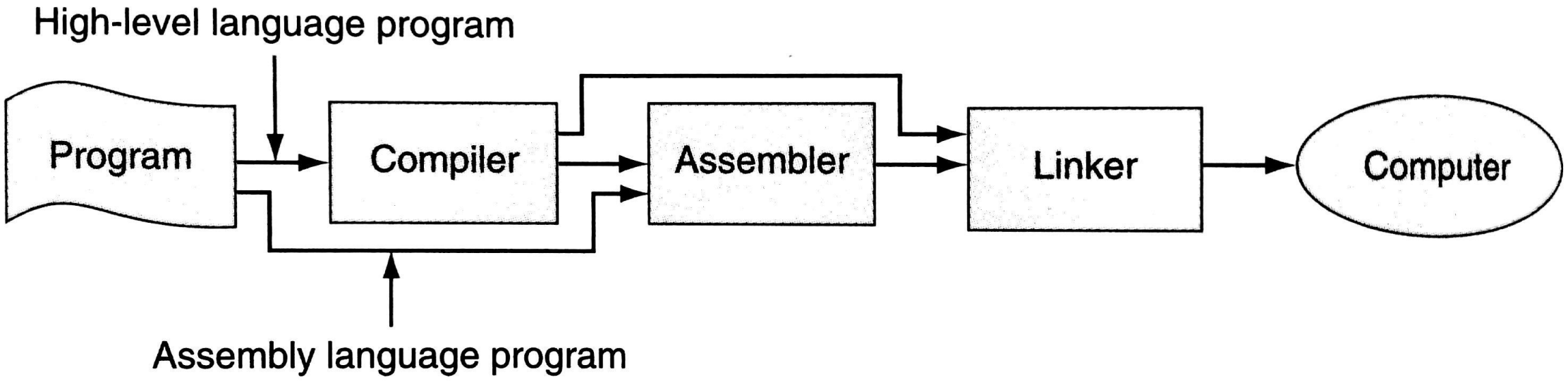


FIGURE B.1.6 Assembly language either is written by a programmer or is the output of a compiler.

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

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

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

spim mips simulator

Brought to you by: jameslarus

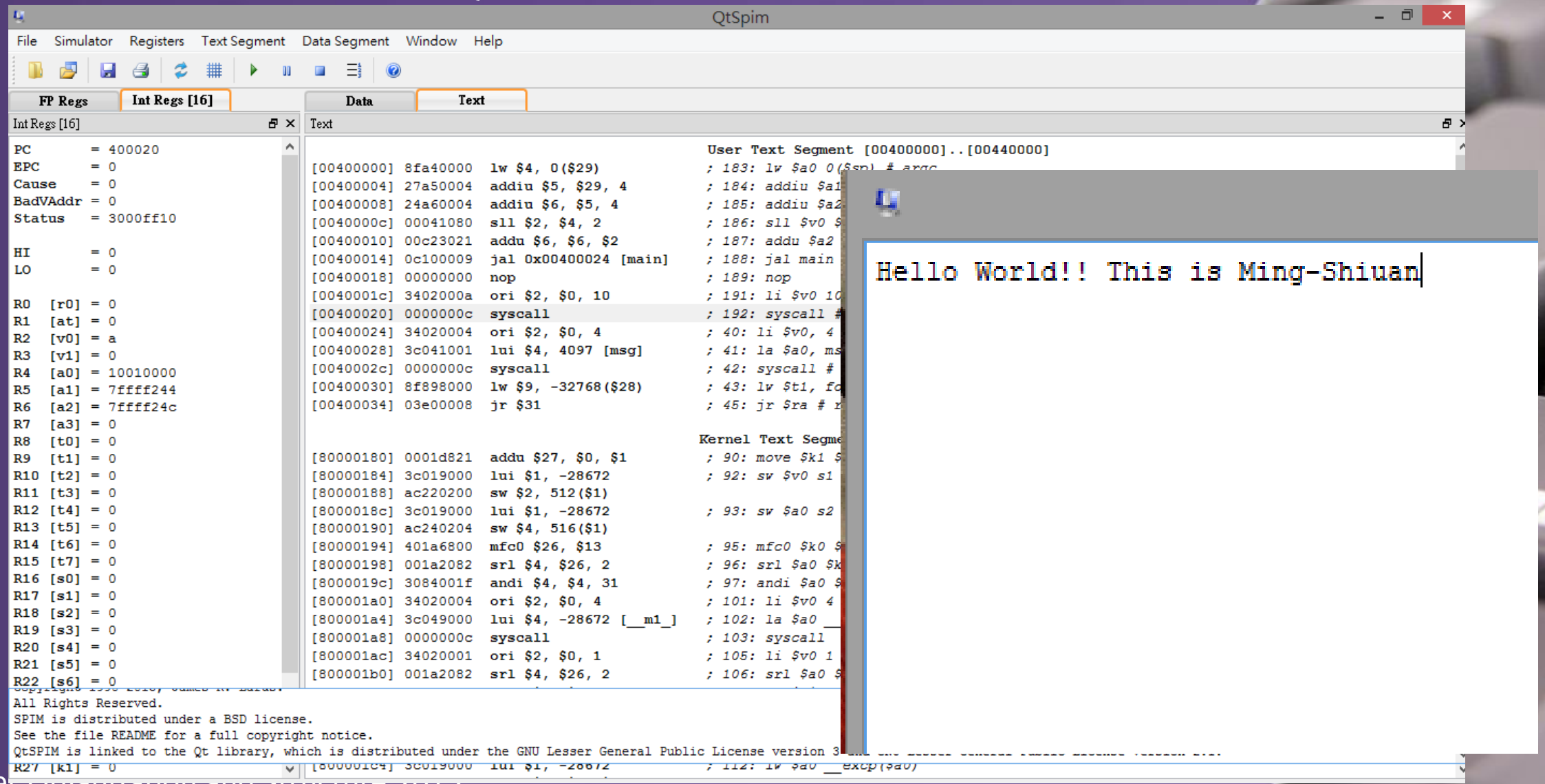
Summary | Files | Reviews | Support | Wiki | Code | Tickets ▾

Looking for the latest version? **Download QtSpim_9.1.12_Windows.exe (31.5 MB)**

Home

Name	Modified	Size	Downloads / Week
qtspim_9.1.16_linux32.deb	2015-08-26	29.7 MB	71
QtSpim_9.1.16_mac.mpkg.zip	2015-08-26	28.7 MB	248
qtspim_9.1.16_linux64.deb	2015-08-26	28.1 MB	137
QtSpim_9.1.16_Windows.exe	2015-08-26	34.3 MB	1,014
QtSpim_9.1.15_mac.mpkg.zip	2015-04-26	28.7 MB	128
QtSpim_9.1.13.mac.mpkg.zip	2014-02-05	33.1 MB	2
QtSpim_9.1.12_Windows.exe	2013-12-17	31.5 MB	547
qtspim_9.1.12_linux32.deb	2013-12-14	1.1 MB	10
qtspim_9.1.12_linux64.deb	2013-12-14	1.1 MB	120
qtspim_9.1.9_linux64.deb	2013-01-23	1.1 MB	3

QtSPIM Screenshot



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

MIPS

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)

Arthmetic 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

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 VS MIPS

```c
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
```

MIPS— 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
......
```

MIPS— 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 ystem services.

$v0

MIPS System Calls - Example

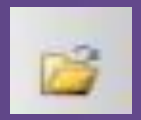
Output: "the answer = 5"

```
.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.

Execute Program in

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

Homework 2

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)

Find out all prime numbers
Tower of Hanoi
Greatest Common Divisor

- One bonus program : Binary Search

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:

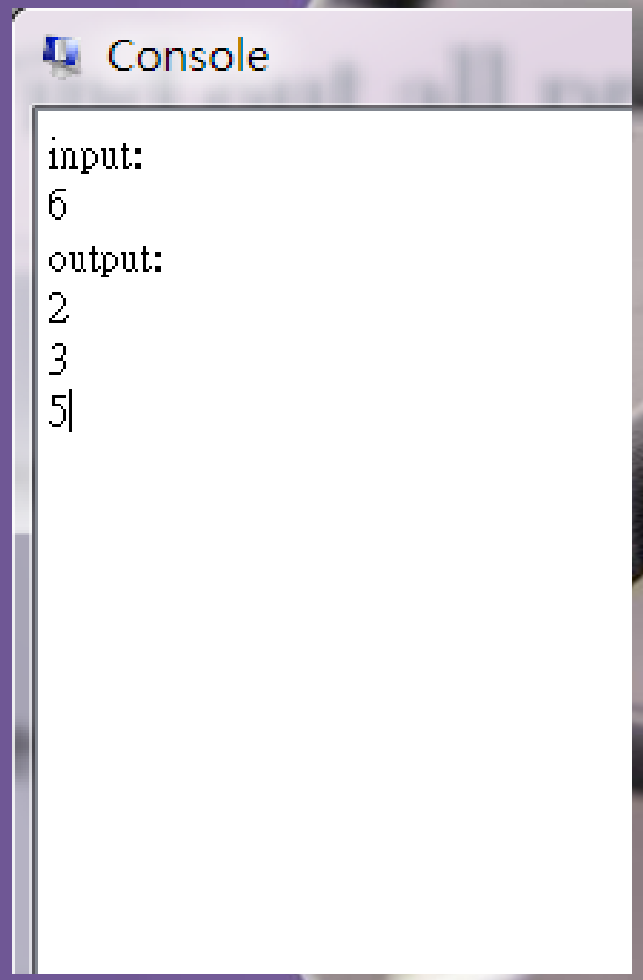
  a positive integer n>1

- Output:

  All prime numbers which is smaller than n

- Requirements:

  1. Print the correct answer.
  2. Can do many calculations iteratively
  3. The file name is FindPrime.s



Problem 2: Tower of Hanoi

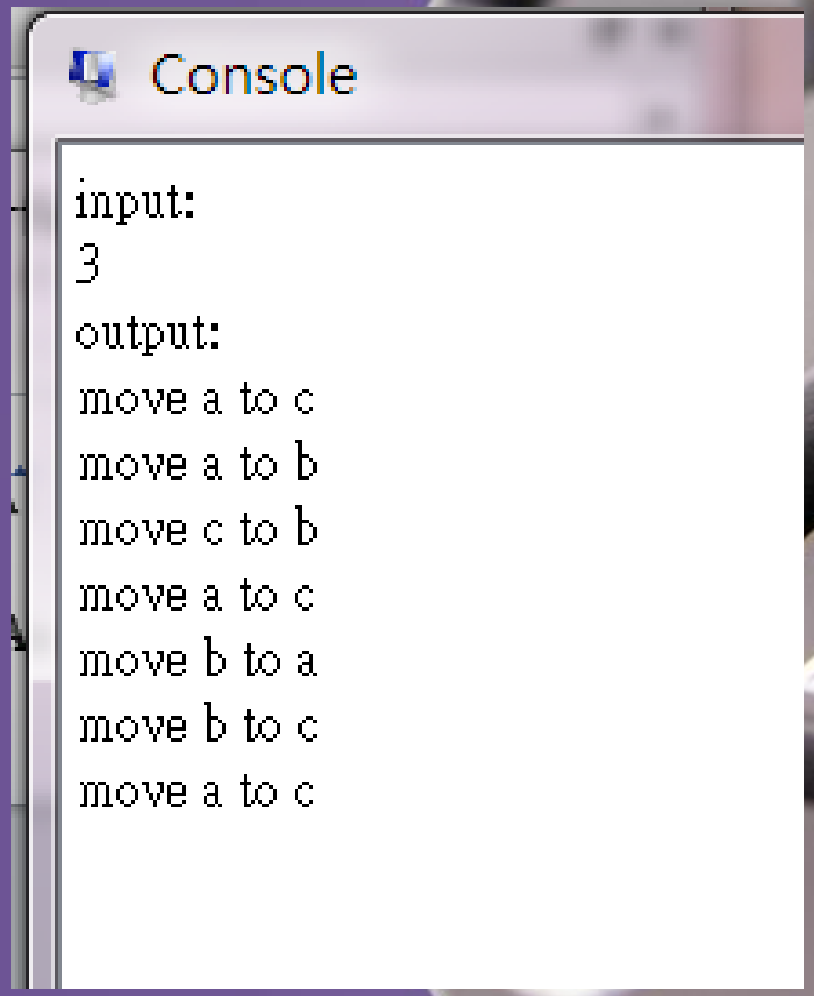
- A hanoi tower with 3 rods A,B,C and n disks. Move all the disk from A to C
- Input:

  a positive integers n, $1<n\leq 5$
- Output:

  Print all the steps
- Requirements:

  Print the correct step.

  The file name is Hanoi.s



Problem 3: GCD

- Input :

  two positive integers x, y > 1

- Output :

  the greatest common divisor of x and y. Output gcd(x, y)

- Requirements:

  1. Print the correct answer
  2. Can do many calculations iteratively
  3. The file name is GCD.s

Optional: Binary Search

- Input:

  1. n positive integers, where n < 8
  2. value to search

- Output:

  1.If value can be find, print “Find it.”.

  2. If value not in the sequence,print “The value is not in this sequence.”.

- Requirements:

  Print the correct answer.

  The file name is Binary_Search.s

Submisssion

- Deadline: 11:59 PM, Monday, Oct. 19, 2015
- You must submit at least the following files:
  - FindPrime.s
  - Hanoi.s
  - GCD.s
  - Binary_Search.s (Optional)
  - (Your student id)_hw2_document.pdf
- **The attach filename should be like b03xxxxxx.zip**
- **Email your zipped file to TA:**

  intere2960@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) Fir. 8:00~10:00

  Ming-Shiuan Chen(陳明軒) Tus. 13:00~15:00

- **Contact us if you have any problem**

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

  包子: intere2960@cmlab.csie.ntu.edu.tw

Thank You For Your Attention