SPIM & MIPS Programming

Presented by TAs
Sheng-Jie L., Chun-Liang L., Ko-Yuan C.

National Taiwan University
Department of Information Management

October 7, 2008
Computer Organization and Structure

Outline

- SPIM - Getting Start
- MIPS Assembly Language Programming
- Programming Assignment (HW#2)

SPIM

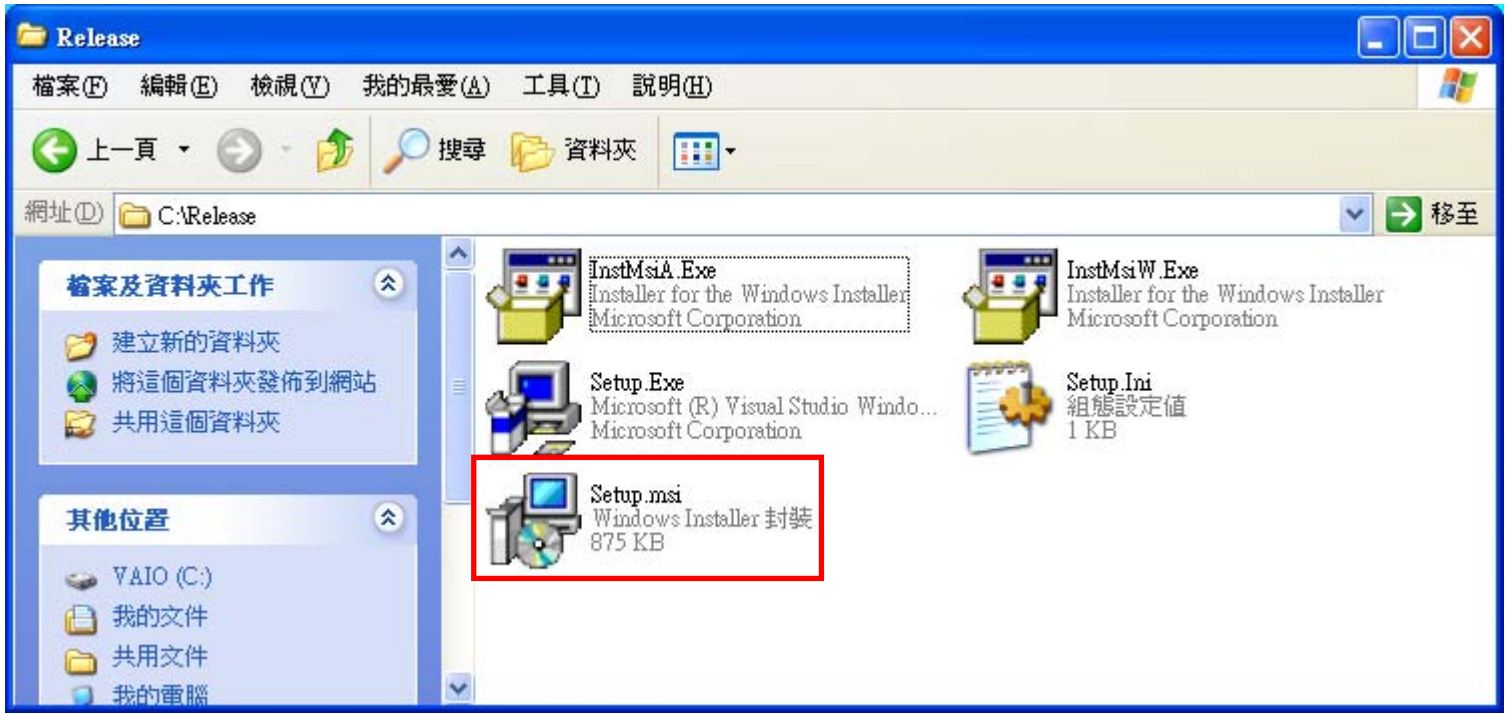
- SPIM is a **MIPS32 simulator** that reads and executes assembly language program written for SPIM.
- Platform - Unix, Linux, Mac OS X, and Microsoft Windows
- The homepage of SPIM:
  - http://pages.cs.wisc.edu/~larus/spim.html

Download SPIM

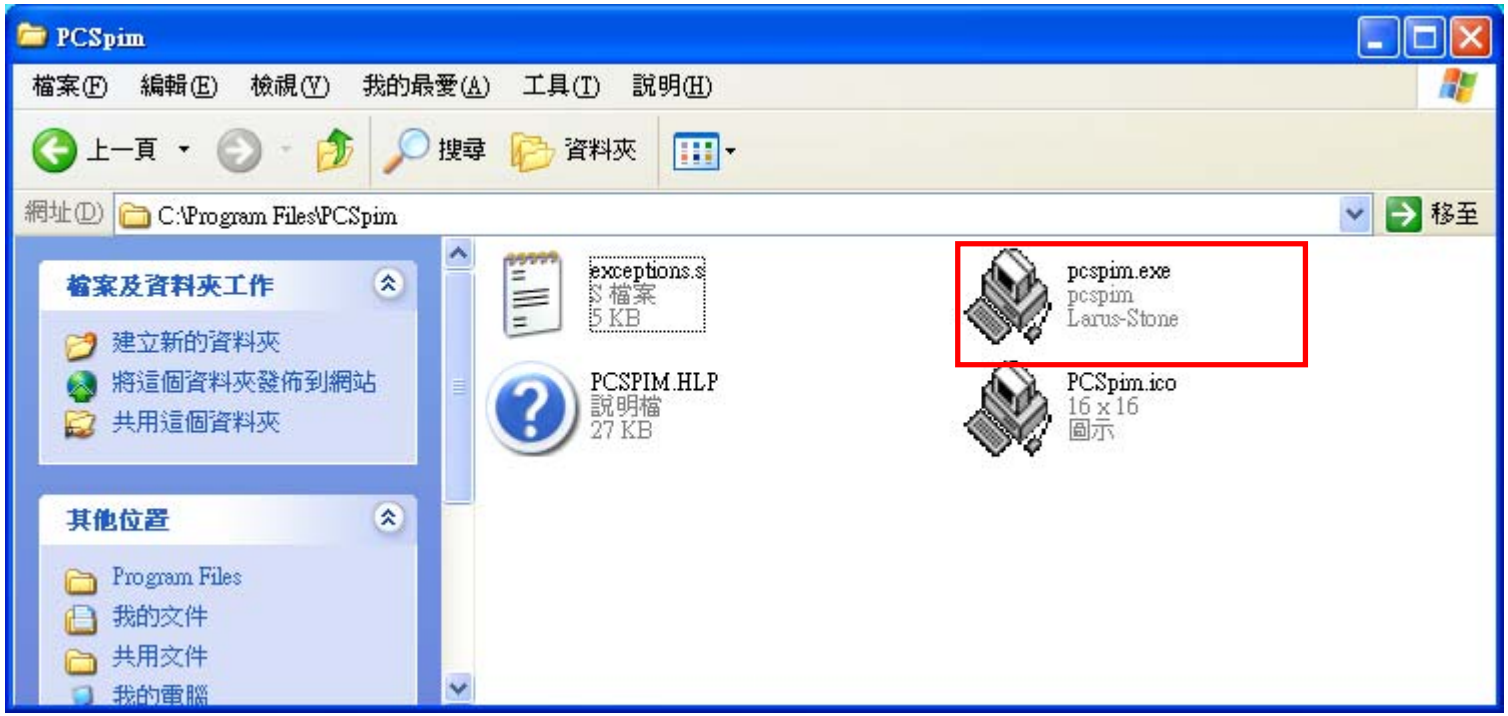
Downloading SPIM

Platform	Program	Form	File
Unix or Linux system Mac OS X	*spim* *xspim*	Source code	http://www.cs.wisc.edu/~larus/SPIM/spim.tar.Z or http://www.cs.wisc.edu/~larus/SPIM/spim.tar.gz
Linux	*spim* *xspim*	Binary RPM for Fedora	http://www.cs.wisc.edu/cbi/downloads/
Microsoft Windows (Windows NT, 2000, XP)	*spim* *PCSpim*	Executable	http://www.cs.wisc.edu/~larus/SPIM/pcspim.zip
(spim 7.0 and later versions no longer run on Windows 95/98. Use version 6.5 or earlier.)		Source code	http://www.cs.wisc.edu/~larus/SPIM/pcspim_src.zip

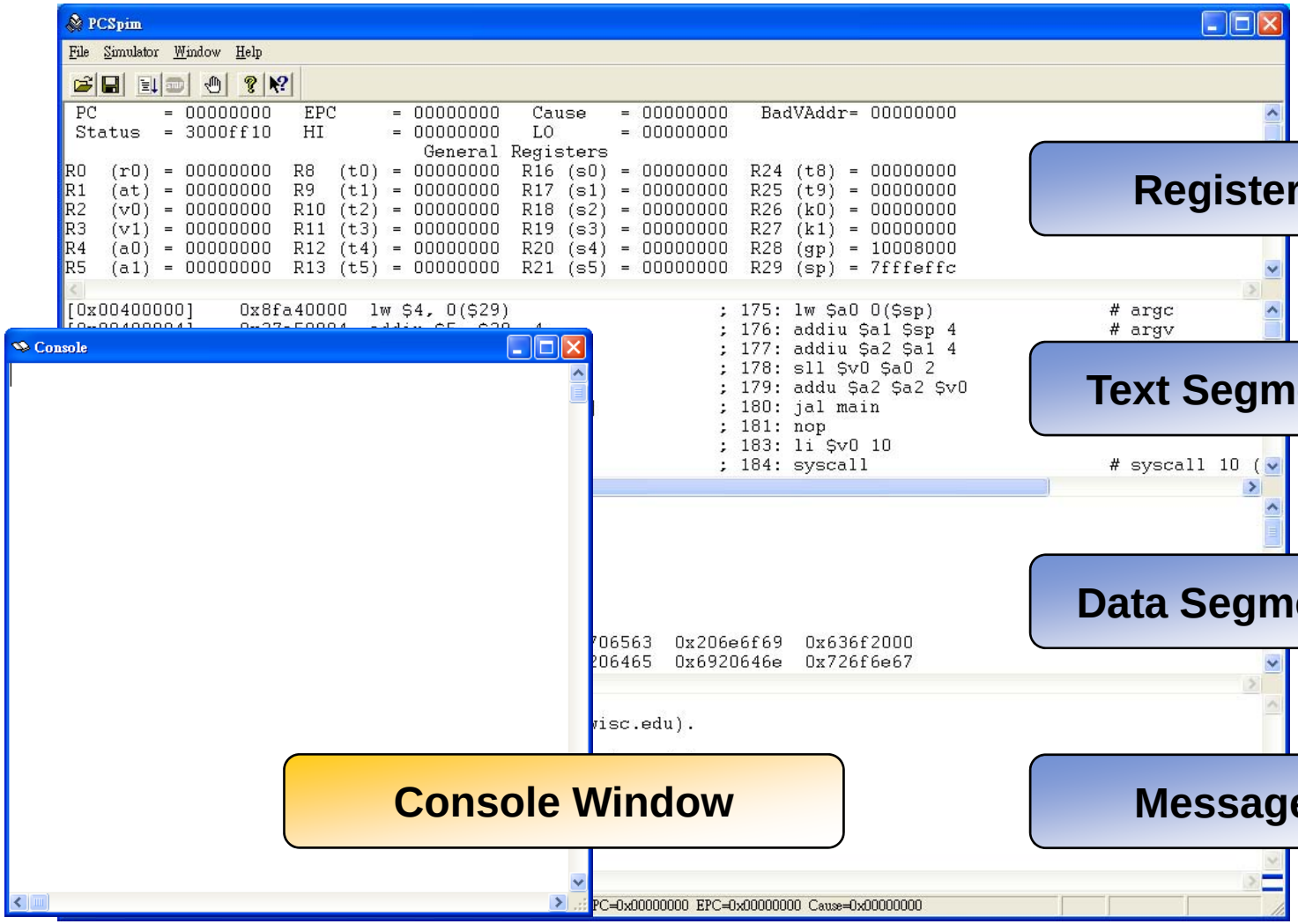
Install SPIM



Start SPIM


PCSpim
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
資料夾
網址(D) C:\Program Files\PCSpim
移至
檔案及資料夾工作
建立新的資料夾
將這個資料夾發佈到網站
共用這個資料夾
其他位置
Program Files
我的文件
共用文件
我的電腦
exceptions.s
S 檔案
5 KB
PCSPIM.HLP
說明檔
27 KB
pcspim.exe
pcspim
Larus-Stone
PCSpim.ico
16 x 16
圖示

Screenshot


PCSpim
File Simulator Window Help
PC = 00000000 EPC = 00000000 Cause = 00000000 BadVAddr= 00000000
Status = 3000ff10 HI = 00000000 LO = 00000000
General Registers
R0 (r0) = 00000000 R8 (t0) = 00000000 R16 (s0) = 00000000 R24 (t8) = 00000000
R1 (at) = 00000000 R9 (t1) = 00000000 R17 (s1) = 00000000 R25 (t9) = 00000000
R2 (v0) = 00000000 R10 (t2) = 00000000 R18 (s2) = 00000000 R26 (k0) = 00000000
R3 (v1) = 00000000 R11 (t3) = 00000000 R19 (s3) = 00000000 R27 (k1) = 00000000
R4 (a0) = 00000000 R12 (t4) = 00000000 R20 (s4) = 00000000 R28 (gp) = 10008000
R5 (a1) = 00000000 R13 (t5) = 00000000 R21 (s5) = 00000000 R29 (sp) = 7fffeffc
[0x00400000] 0x8fa40000 lw $4, 0($29) ; 175: lw $a0 0($sp) # argc
; 176: addiu $a1 $sp 4 # argv
; 177: addiu $a2 $a1 4
; 178: sll $v0 $a0 2
; 179: addu $a2 $a2 $v0
; 180: jal main
; 181: nop
; 183: li $v0 10
; 184: syscall # syscall 10 (
706563 0x206e6f69 0x636f2000
206465 0x6920646e 0x726f6e67
visc.edu).
Console
PC=0x00000000 EPC=0x00000000 Cause=0x00000000
Register Window
Text Segment Window
Data Segment Window
Console Window
Message Window

MIPS Assembly Language

- Operations code (Opcode)
  - add, sub, addi, addu, addiu, subu
  - lw, sw, lbu, sb, lui, ori
  - beq, bne, slt, slti, sltu
  - 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
  - $sp stack pointer
  - $fp frame pointer
  - $ra return address
  - …

Data Types

- .word, .half - 32/16 bit integer
- .byte - 8 bit integer
- .ascii, .asciiz - string
- .double, .float - floating point

- .space n
  - Allocate n bytes of space in the current segment (which must be the data segment in SPIM).

System Call

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	

An Example: Hello World

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

Writing MIPS - Overview

```
.data
```

Put static data here.

```
.text
```

Put all your code here.

Writing MIPS - Data Segment

- [LABEL] [DATA TYPE] [VALUE]

```
  .data
Mystr: .asciiz “Hello, I’m a TA.¥n”
Yourint: .word 90
Hisarray:    .word 100, 100, 100
             .word 20 , 40 , 60
             .word 1  , 2  , 3
  .text
..........
```

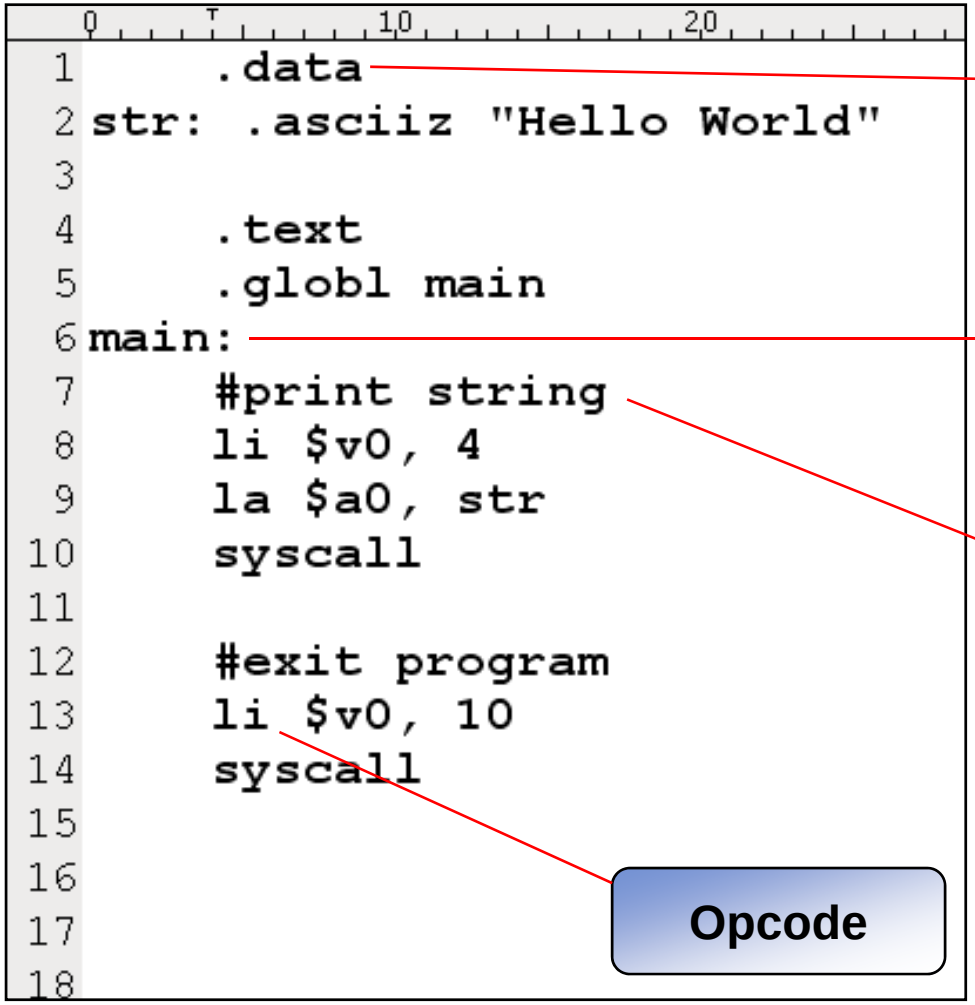
Writing MIPS - Text Segment

```
  .data
..........

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

Assembler Syntax

- **Comments** in assembler files begin with a sharp-sign (#).
- **Identifers** are a sequence of alphanumeric characters, underbars (_), and dots (.) that do not begin with a number.
- **Opcodes** for instructions are reserved words that are not valid identifers.
- **Labels** are declared by putting them at the beginning of a line followed by a colon.

Assembler Syntax (cont.)



Homework (L)

Homework (L)

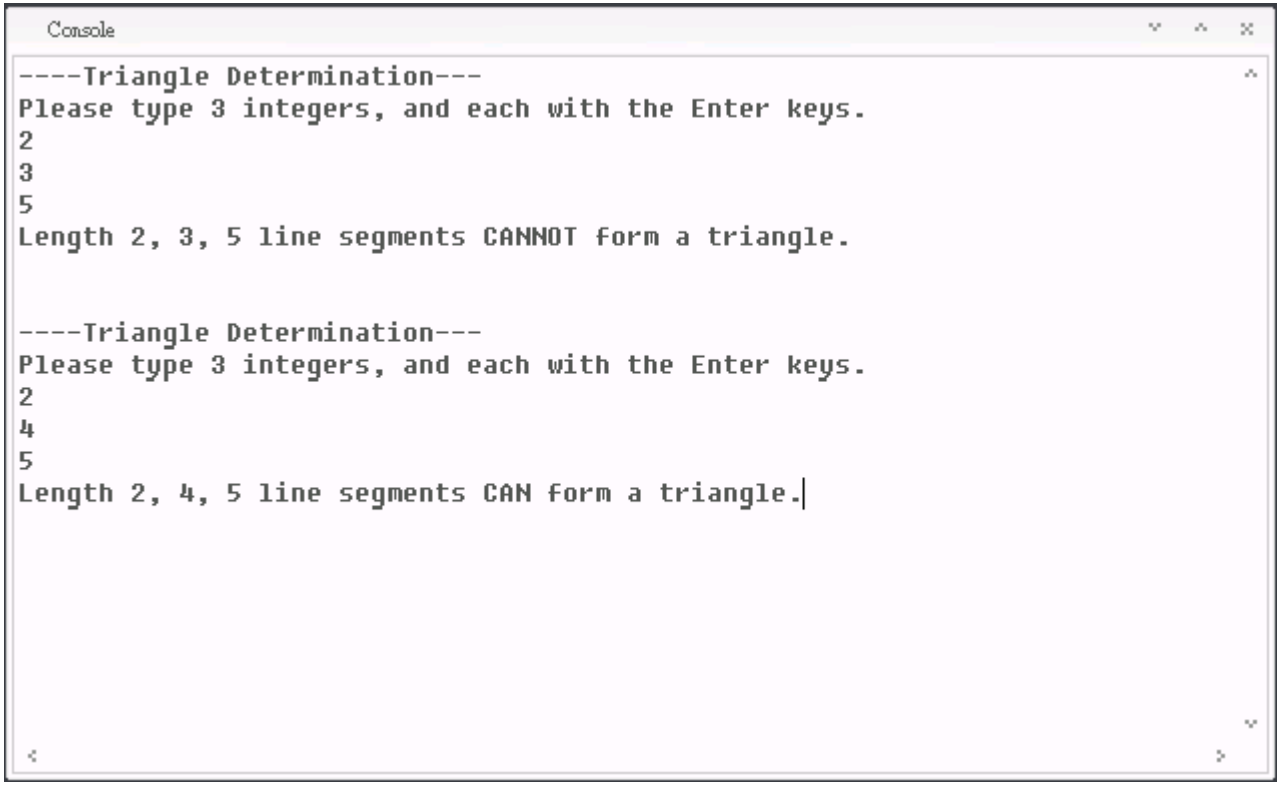
- This is an individual assignment.
- Write the following three programs in MIPS assembly language. (Must run correctly on SPIM)
  - 1. Triangle determination.
  - 2. Variation of Fibonacci sequence.
  - 3. Finding largest prime.

Problem 1.
Triangle Determination

- We will give you three positive numbers as the length for each straight line segment, and your job is to write a program to determine whether these three line segments can form a triangle or not.
- If input is a negative integer, please show an error message.
- The file name is triangle.s

Problem 1. (cont.) Triangle Determination

- Sample Output



Problem 2.
Variation of Fibonacci sequence

- We all understand the well-known Fibonacci sequence, defined by the following recurrence relation.

$$F_n = \begin{cases} 0, if\ n = 0 \\ 1, if\ n = 1 \\ F_{n-1} + F_{n-2}, if\ n > 1 \end{cases}$$

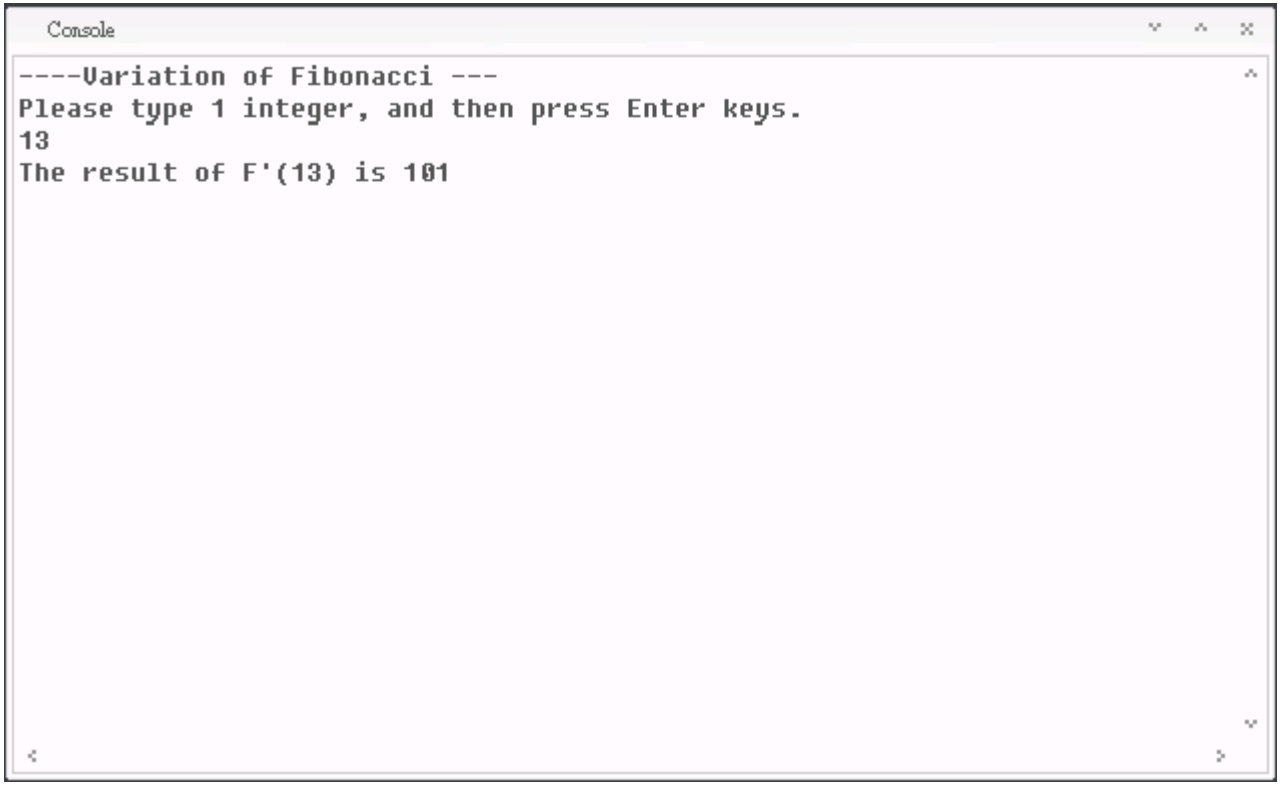
- Now we make some variation on the original Fibonacci sequence with the new definition as follow.

$$F'_n = \begin{cases} 0, \ if\ n = 0 \\ 1, \ if\ n = 1 \\ 2, \ if\ n = 2 \\ F_{n-1} + F_{n-3}, \ if\ n > 2 \end{cases}$$

- Your job is to implement a recursive function to solve the problem above.

Problem 2. (cont.) Variation of Fibonacci sequence

- If input is a negative integer, please show an error message.
- The file name is VarFibonacci.s
- Sample Output



Problem.3
Prime number finding

- Implement a program to find the *largest prime number* which is smaller than or equal to a user-defined number

- For example, if user input a positive number 30, and your function should find out the largest prime number within the range [0, 30], which is 23.

- Here we introduce the *Sieve of Eratosthenes algorithm* to solve this problem. The most efficient way to find all of the small primes (say all those less than 10,000,000) is by using the *Sieve of Eratosthenes(ca 240 BC)*

Problem.3 (cont.)
Prime number finding

For example, to find all the primes less than or equal to 30, first list the numbers from 2 to 30.

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

The first number 2 is prime, so keep it and cross out its multiples , so the red numbers are not prime.

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

The first number left is 3, so it is the first odd prime. Keep it and cross out all of its multiples. We know that all multiples less than 9 (i.e. 6) will already have been crossed out, so we can start crossing out at $3^2=9$.

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Problem.3 (cont.)
Prime number finding

Now the first number left is 5, the second odd prime. So keep it also and cross out all of its multiples (all multiples less than $5^2=25$ have already been crossed out, and in fact 25 is the only multiple not yet crossed out).

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

The next number left, 7, is larger than the square root of 30, so there are no multiples of 7 to cross off that haven't already been crossed off (14 and 28 by 2, and 21 by 3), and therefore the sieve is complete!!

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

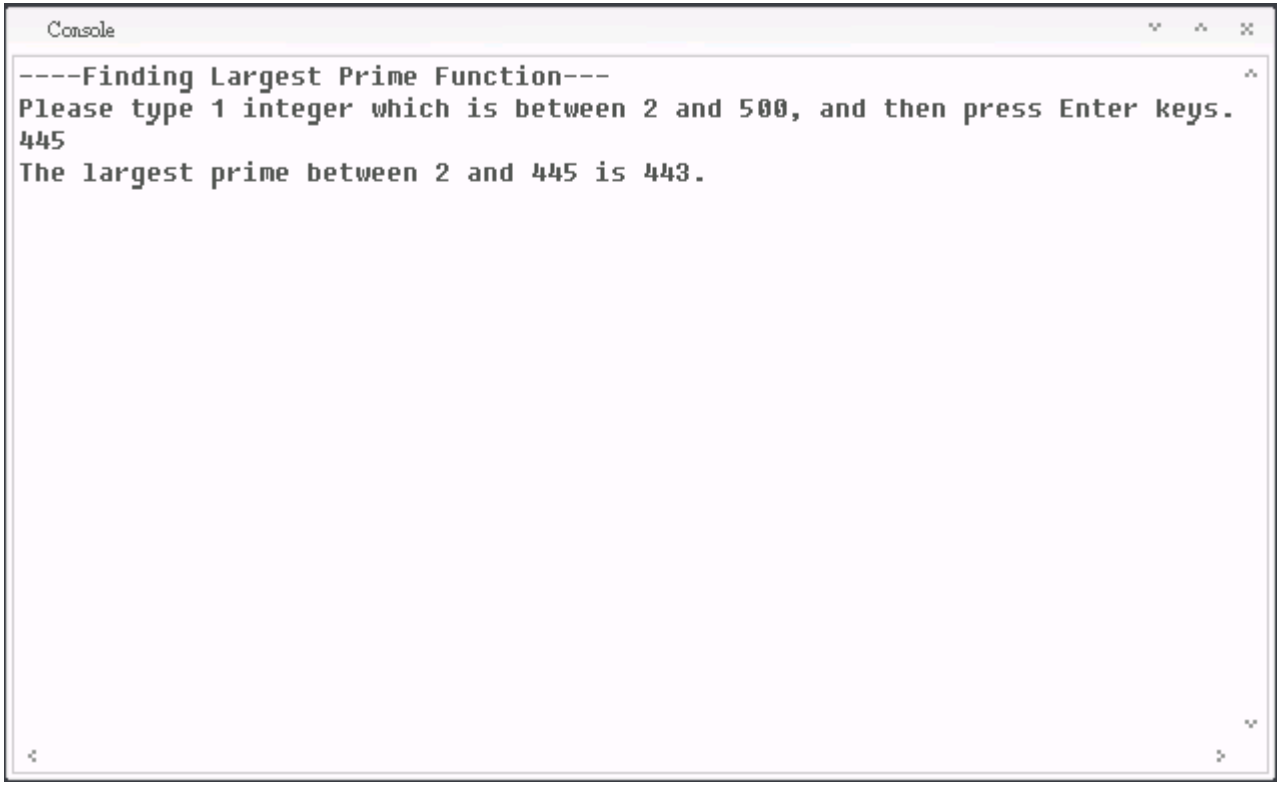
Problem.3(cont.) Prime number finding

- C++ implementation

```
unsigned int Eratosthenes(unsigned int n)
{
        unsigned int Sieve[500];
        Sieve[0] = Sieve[1] = 0;
        for(unsigned int i = 2 ; i < 500; i++)
                Sieve[i] = 1;

        for (unsigned int i = 2; i * i < 500; i++)
                if (Sieve[i] == 1)
                        for (unsigned int j = (i + i); j < 500; j += i)
                                Sieve[j] = 0;

        for (unsigned int j = 499; j > 1; j--)
                if (Sieve[j] == 1)
        return j;

return 0;
}
```

Problem.3(cont.) Prime number finding

- If input *n* is a negative integer or *n* >= 500, please show an error message.
- The file name is FindPrime.s
- Sample Output



Submission

- Deadline: Tuesday October 28, 2008 11:59 PM
- You must submit at least the following files.
  - Triangle.s
  - VarFibonacci.s
  - FindPrime.s
  - (Your student id)_hw2_document.doc/pdf
- Please put all your files in a directory named by your student id in lowercase, and then compress it into one zipped file. (e.g. zip/tgz …)
- You should email your zipped file HW#2 to TAs before the deadline. The file name will be like b96xxxxxx.zip.

Grading Guidelines

Description	For Each Problem
Program compiles without errors	10%
Program executes correctly	60%
Documentation and description of the program	10%
Demo and Presentation	10%
Implement Detail	10%