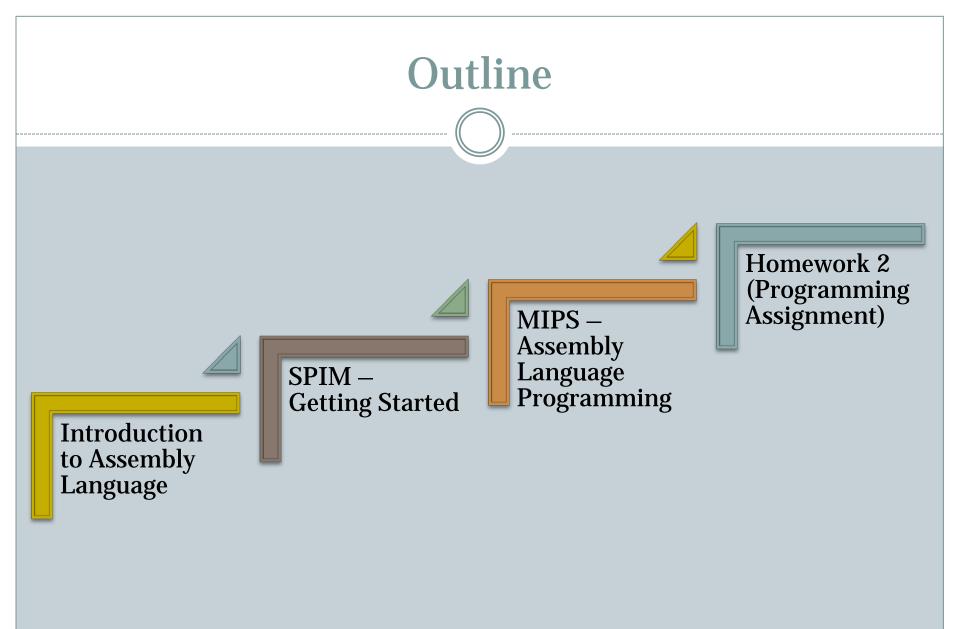
#### **Computer Organization and Structure 2012**



# SPIM & MIPS Programming

Department of Information Management National Taiwan University



## **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: <u>http://spimsimulator.sourceforge.net/</u>
- Assemblers, Linkers, and the SPIM Simulator:<u>http://pages.cs.wisc.edu/~larus/HP\_A</u> ppA.pdf
- MIPS Instruction

Reference:<u>http://www.mrc.uidaho.edu/mrc/peopl</u> e/jff/digital/MIPSir.html

### **QtSPIM - Installation**

#### SPIM: A MIPS32 Simulator

James Larus spim@larusstone.org

#### Contents

- Older Versions of SPIM
- Further Information
- <u>Changes to SPIM</u>
- 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 <u>not</u> execute binary (compiled) programs.

Spim implements almost the entire MIPS32 assemblerextended 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 <u>older programs *xspim* and *PCSpim*</u> provide window interfaces for these systems as well.



#### What's New?

*QtSpim* is a new user interface for *Spim* built on the <u>Qt UI</u> <u>framework</u>. 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).

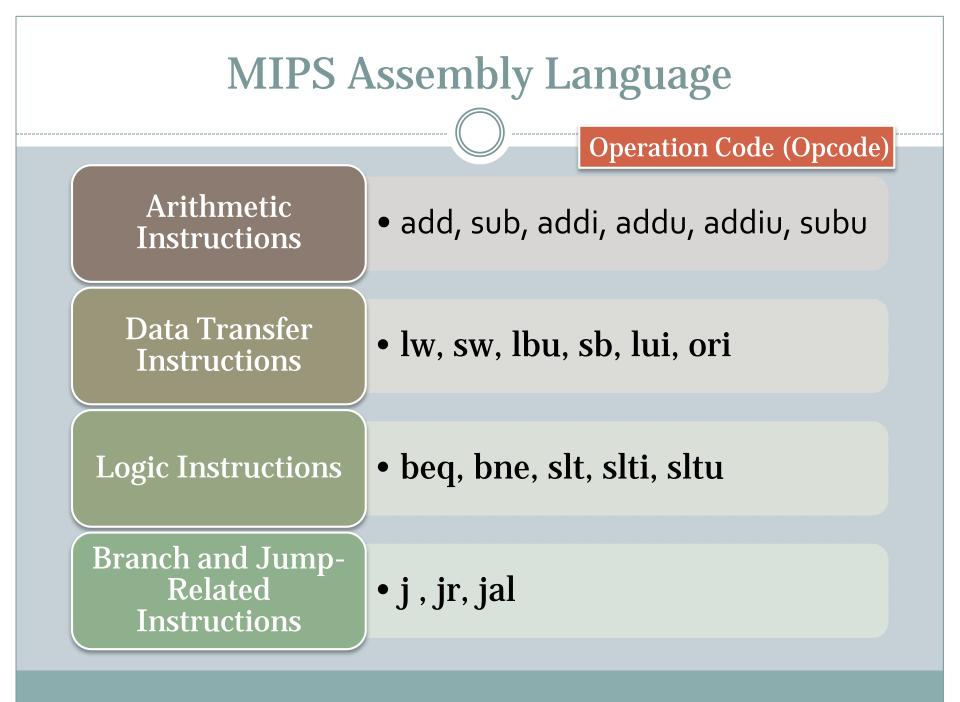
Spim has moved to SourceForge! The source code for all version of Spim are in an SVN repository and compiled version are available for download. There is also a bug tracker and discussion forum. Spim is an open source project, so please join in and contribute.

### **QtSPIM - Installation**

Home				2
Name 🕈	Modified +	Size +	Downloads +	
QtSpim_9.1.7_Windows.zip	Windo	WS MB	774	0
qtspim_9.1.6_linux32.deb	2012-02-06	1.1 MB	69	0
qtspim_9.1.6_linux64.deb	2012-02-06	1.1 MB	54	0
QtSpim_9.1.5_Windows.zip	2012-01-03	21.0 MB	9	0
qtspim_9.1.5_linux64.deb	2012-01-03	1.1 MB	5	0
qtspim_9.1.5_linux32.deb	2012-01-03	1.1 MB	10	0
PCSpim_9.1.4.zip	2011-09-04	5.7 MB	163	0
readme.txt	2011-07-27	686 Bytes	18	0
QtSpim_9.0.3_mac.dmg	Mae	C <sub>IB</sub>	114	0
Totals: 9 Items		69.7 MB	1,216	

### **QtSPIM - Screenshot**

g QtSpim		
File Elizable Registers Test Segment Date Deg	nt Wadow Help	TEST2 helloword
	. = .	
The second second		
FP Regs Int Rogs [16]	Data Text	
d Regs [16] 8 ×	inter contract of the second se	7 ×
C = 400040 *		·
EPC = 0	Rernel data segment [90000000][90010000]	
Cause = 0	[9000000] 78452020 74706563 206e6f69 636f2000 Exception . oc	
MadVAddr = 0	[90000010] 72727563 61206665 6920646e 726f6e67 curred and ignor	
Status = 3000ff10	[90000020] 000a6665 493b2020 7265766 74707572 ed., [Interrupt	£
	[90000030] 20002054 4c545b20 20005442 4c545b20 ] . [T L B ] . [T L	
II = 0	[90000040] 20005d42 4c545b20 20005d42 64415b20 B ] . [T L B ] . [ A d	
0 = 0	[90000050] 73657264 72652073 2072672 69206669 dresserrorin i	
	[90000060] 2f74736e 61746164 74656620 20546863 nst/datafetch]	
80 [r0] = 0	[90000070] 5b202000 72646441 20737365 6f127265 . [Addresserro	
11 [at] = 0	[9000080] 6e632072 6f747320 20546572 5b202000 r in store] . [	
12 [v0] = a	[9000090] 20646142 74736e69 74637572 206e6f69 Bad instruction	
3 [v1] = 0	[900000a0] 72646461 5d737365 20200020 6461425b address] . [Bad	
(4 [a0] = 5	[900000b0] 74616420 64612061 73657264 00205d73 data address] .	
35 [a1] = 7ffff868	[900000c0] 45552020 726£7272 206e6920 63737973 [Error in sysc	
<pre>k6 [a2] = 7ffff870 #</pre>	[900000d0] 5d6c6c61 20200020 6572425b 6f706b61 all], [Breakpo	
27 [a3] = 0	[900000e0] 5d746e69 20200020 7365525b 65767265 1 n t ] . [Reserve	
R8 [t0] = 0	[900000f0] 6e692064 75727473 6f697463 00205d6e d instruction] .	
19 [t1] = 0	[90000100] 5b202000 74697241 74656d68 6f206369 . [Arithmetic o	
10 [t2] = 0	[90000110] 66726576 5d776f6c 20200020 6172545b verflow] . [Tra	
11 [t3] = 0	[90000120] 00205dT0 5b202000 616f6c46 676e6974 p ] . [Flostin] to Console	— D — X
k12 [t4] = 0	[90000130] 69617020 20507468 20000000 61435520 point] [C	
213 [t5] = 0	[90000140] 636f7270 005d3220 20000000 444d5b20 proc 2] [M the approx = 5	
814 [t6] = 0	[90000150] 005d584d 575b2020 68637461 2020005d MX]. [Watch].	
k15 [t7] = 0	[90000160] 63614d5b 65666968 65686320 005d6b63 [Machine check]	
16 [s0] = 0	[90000170] 00000000 5b202000 68636143 00005d65 [Cache].	
17 [s1] = 0	[90000180] 90000024 90000033 9000003b 90000043 \$ 3 ; C	
18 [s2] = 0	[90000190] 900004b 9000071 900008d 90000aa Kq	
819 [s3] = 0	[900001a0] 900000c0 900000d6 900000e6 90000100	
20 [s4] = 0	[900001b0] 90000101 9000011a 90000124 90000125	
21 [85] = 0	[900001c0] 90000139 9000013a 9000013b 90000148 9 : ; H	
222 [s6] = 0	[900001d0] 90000149 9000014a 9000014b 90000154 IJKT	
23 [s7] = 0	[900001e0] 9000015e 90000170 90000171 90000172 ^pqr.	
24 [t8] = 0	[900001f0] 90000173 90000174 90000175 9000017f atu	
25 [t9] = 0	[30000200][30001111] 0000000	
$k_{26} [k_0] = 0$		
27 [k1] = 0 +		
PIN Veraons 1. Kothebroary12,4012 Jogynight1990-2012, Inneeß, Larus. IlRightsRearved. PUndrithsburd undernBSD licenze. sethefileREADMEforsfulkopynightsotare.	Consol	e Window



### **MIPS Assembly Language**

### × MIPS Registers and Usage Convention

- × \$zero constant 0
  - expression of a function
    - argument 1~4
    - temporary registers
    - save registers
    - stack pointer
    - frame pointer
    - return address

X ...

× \$sp

× \$fp

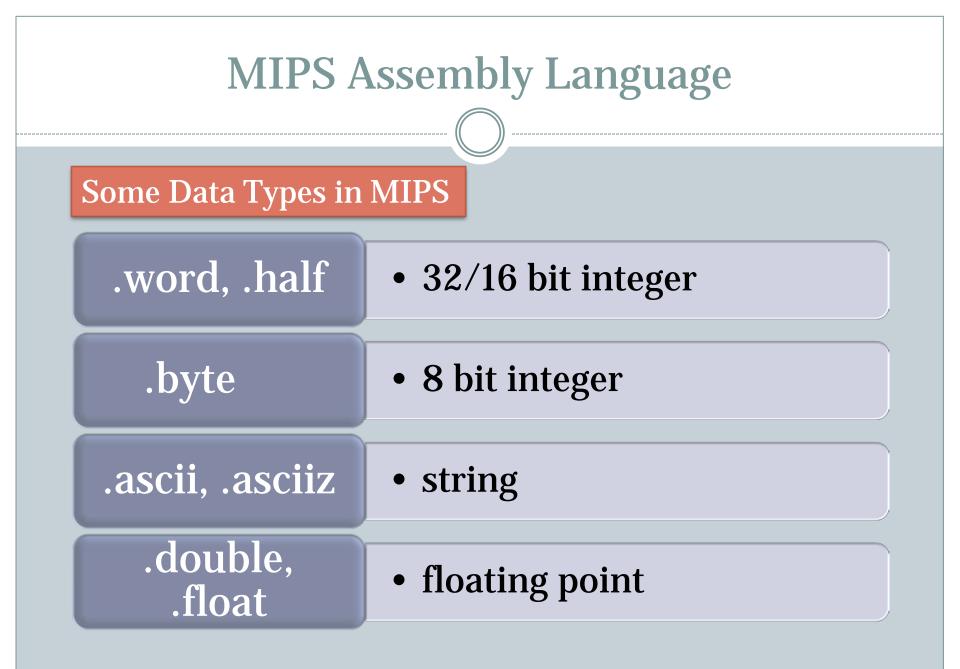
× \$ra

× \$v0, \$v1

× \$a0~\$a3

× \$t0~\$t9

× \$s0~\$s7



### **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

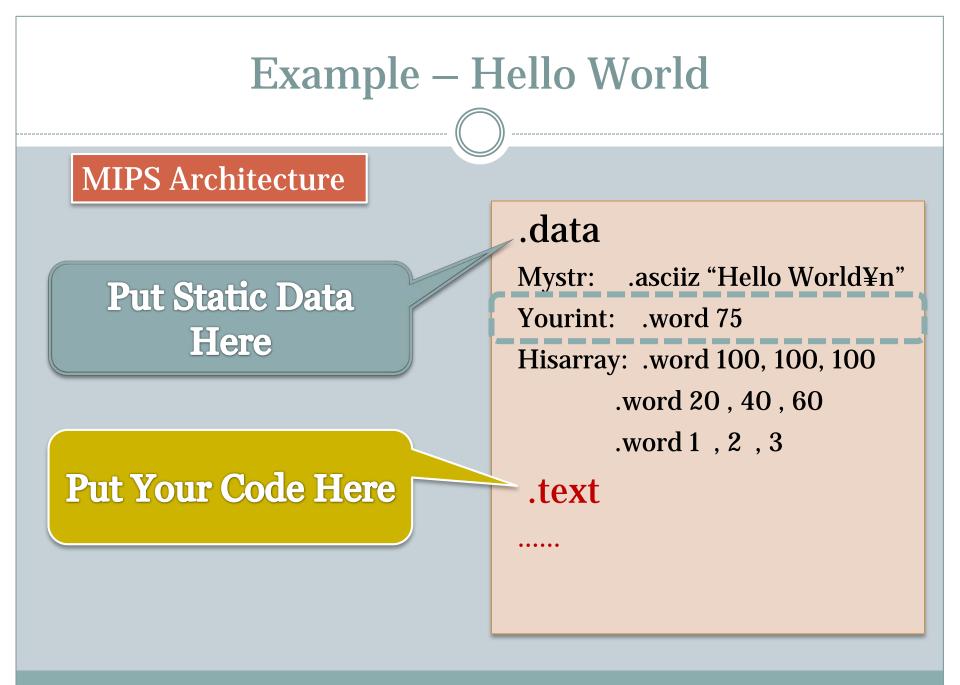
### C vs MIPS

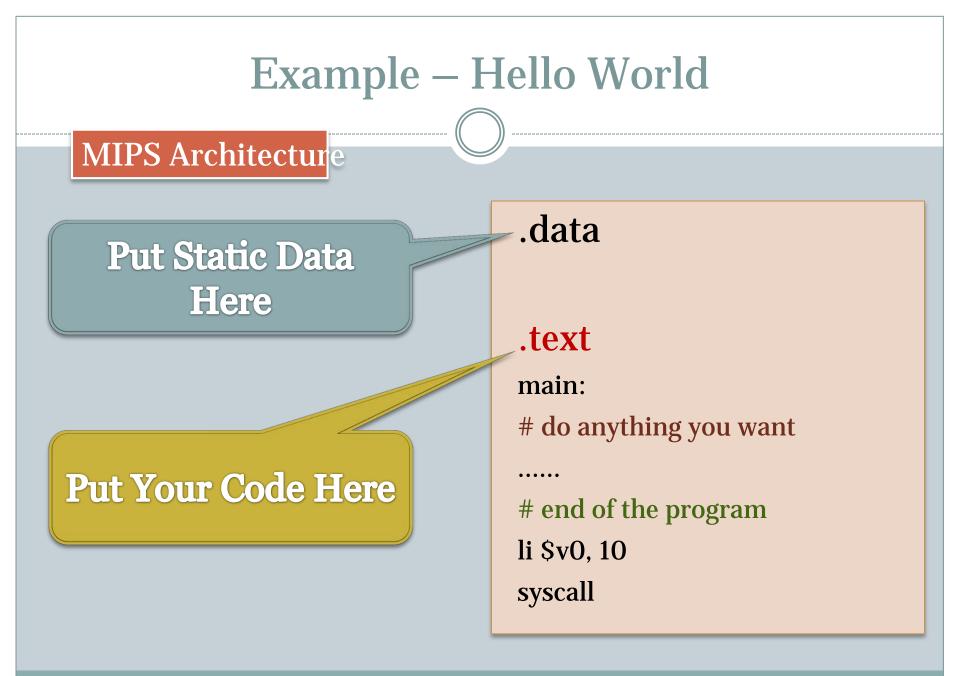
int main()

```
printf("Hello World¥");
return 0;
```

.data Mystr: .asciiz "Hello World¥n"

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





### **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	<pre>\$a0 = filename (string), \$a1 = flags, \$a2 = mode</pre>	file descriptor (in \$a0)
read	14	<pre>\$a0 = file descriptor, \$a1 = buffer, \$a2 = length</pre>	num chars read (in \$a0)
write	15	<pre>\$a0 = file descriptor, \$a1 = buffer, \$a2 = length</pre>	num chars written (in \$a0)
close	16	\$a0 = file descriptor	
exit2	17	\$a0 = result	

FIGURE A.9.1 Sv0 i 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  $\rightarrow$  Reinitialize Simulator  $\blacksquare$ 

Open your .s file 📷

Simulator  $\rightarrow$  Clear Registers

Simulator  $\rightarrow$  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** 

- 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

# 💵 Console input: ĥ output: 2 3 5

### **Problem 2: Mutual prime**

#### Input:

two positive integers x,y>1

### output:

- If gcd(x, y)=1,output True
- otherwise output False

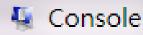
### Requirements:

- Print the correct answer.
- Can do many calculations iteratively
- The file name is MutualPrime.s

Console
nput:
, .3
utput:
rue

### **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



input:

output:

move a to c

move a to b

moveld to b

move a to c

move b to a

move b to c

move a to c

3

### Submission

- Deadline:
- You must submit at least the following files:
  - FindPrime.s
  - MutualPrime.s
  - Hanoi.s
  - o (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 **BOOxxxxxx.zip**.
- Email your zipped file to TA

Grading Guideline	S
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

- o Chien-Wen Jung(榮芊雯) Wed. 15:00~17:00
- Hsiao-Ching You (尤筱青) Wed. 13:00~15:00
- o Shao-Chi Chen (陳少祁) Fri. 10:00~12:00

### Contact us if you have any problem

- o 榮芊雯 winnie87@cmlab.csie.ntu.edu.tw
- o 尤筱青 hcyou@cmlab.csie.ntu.edu.tw
- o 陳少祁auberon@cmlab.csie.ntu.edu.tw



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

# Thank you for Your Attention

#### **ANY QUESTIONS?**