EE 3613: Computer Organization
Chapter 2: Instruction Set Architecture
Introduction – 3/3

Avinash Karanth
Department of Electrical Engineering & Computer Science
Ohio University, Athens, Ohio 45701
E-mail: karanth@ohio.edu
Website:
http://oucsace.cs.ohiou.edu/~avinashk/classes/ee461a/ee461a.htm

1

Course Administration

- SPIM (or Qt-SPIM) tutorial today in-class

2

MIPS ISA So Far

Category	Instr	Op Code	Example	Meaning
Arithmetic (R & I format)	add	0 and 32	add $s1, $s2, $s3	$s1 = $s2 + $s3
	subtract	0 and 34	sub $s1, $s2, $s3	$s1 = $s2 - $s3
	add immediate	8	addi $s1, $s2, 6	$s1 = $s2 + 6
	or immediate	13	ori $s1, $s2, 6	$s1 = $s2 v 6
Data Transfer (I format)	load word	35	lw $s1, 24($s2)	$s1 = Memory($s2+24)
	store word	43	sw $s1, 24($s2)	Memory($s2+24) = $s1
	load byte	32	lb $s1, 25($s2)	$s1 = Memory($s2+25)
	store byte	40	sb $s1, 25($s2)	Memory($s2+25) = $s1
	load upper imm	15	lui $s1, 6	$s1 = 6 * 2^{16}
Cond. Branch (I & R format)	br on equal	4	beq $s1, $s2, L	if ($s1==$s2) go to L
	br on not equal	5	bne $s1, $s2, L	if ($s1 !=$s2) go to L
	set on less than	0 and 42	slt $s1, $s2, $s3	if ($s2<$s3) $s1=1 else $s1=0
	set on less than immediate	10	slti $s1, $s2, 6	if ($s2<6) $s1=1 else $s1=0
Uncond. Jump (J & R format)	jump	2	j 2500	go to 10000
	jump register	0 and 8	jr $t1	go to $t1
	jump and link	3	jal 2500	go to 10000; $ra=PC+4

3

Register Organization

Name	Register Number	Usage	Preserved on Call
$zero	0	Constant Value of 0	n.a.
$v0-$v1	2-3	Values for results and expressions	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved	Yes
$t8-$t9	24-25	More Temporaries	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	Yes

- What about Register 1?

4

Other Instructions

- See handout posted on the class website

5

Example - 1

- Write a MIPS code to move 100 signed 16-bits signed from memory location A (0x00001000) to memory location B (0x00002000)

```
        addi $s1, $zero, 4096       \\ load the address A into $s1
        addi $s2, $zero, 8192       \\ load the address B into $s2
        addi $t0, $zero, 100        \\ load the counter into $t0
LOOP:   lh $s3, 0($s1)              \\ load halfword into $s3
        sh $s3, 0($s2)              \\ store halfword from $s3
        addi $s1, $s1, 2            \\ adjust the address for A
        addi $s2, $s2, 2            \\ adjust the address for B
        subi $t0, $t0, 1            \\ reduce the counter
        bneq $t0, $zero, LOOP       \\ test the loop: $t0 - $zero = ?
```

6

Example - 2

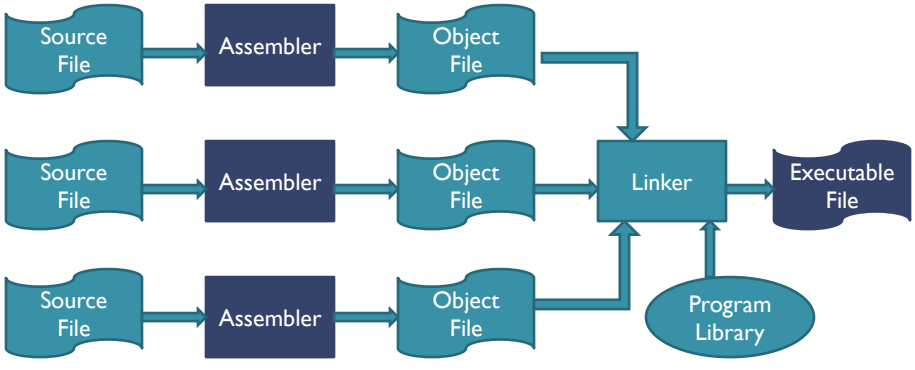
- Write a MIPS code to move 100, 4-byte numbers from memory location A (0x00001000) to memory location B (0x00002000) if the number is positive; else to memory location C (0x00003000)

```
          addi $s1, $0, 4096        \\ load the address A into $s1
          addi $s2, $0, 8192        \\ load the address B into $s2
          addi $s3, $0, 12288       \\ load the address C into $s3
          addi $t0, $0, 100         \\ load the counter into $t0

LOOP:     lw $s4, 0($s1)            \\ load word into $s3
          addi $s1, $s1, 4          \\ loop adjust for address A
          stl $t1, $s4, $0          \\ $t1 = 1 if $s3 < $0
          beq $t1, $0, POSITIVE     \\ Comparing $t1 = $zero ?
          sw $s4, 0($s3)            \\ NEGATIVE, so load into C
          addi $s3, $s3, 4          \\ loop adjust for address C
          j LOOPADJUST
POSITIVE:
          sw $s4, 0($s2)            \\ POSITIVE, so load into B
          addi $s2, $s2, 4          \\ loop adjust for address B
LOOPADJUST:
          subi $t0, $t0, 1
          bneq $t0, $zero, LOOP     \\ test the loop: $t0 - $zero = ?
```

7

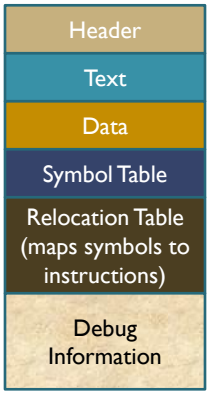
Assembly Language Programming



8

Unix Object File Format

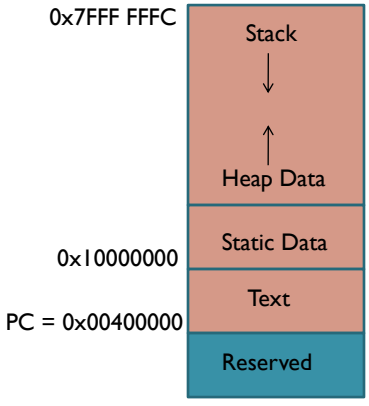
- Header (contains sizes of other parts)
- Text (machine code)
- Data (global and static data)
- Symbol table (associates addresses with external labels, lists unresolved references)
- Relocation Table (identifies instructions and data words that rely on absolute addresses)
- Debug (information contains a precise description of the way the program was compiled)



9

MIPS Layout

- CPU
  - Registers ($0 - $31)
  - ALU Unit
  - Multiply/Divide
    - Lo and Hi Registers (to store totally 64 bits)
- Coprocessor 1 (Floating Point Unit)
  - Registers ($0 - $31)
  - ALU
- Coprocessor 0 (Traps and Memory)
  - BadVAddr, Status, Cause, EPC



10

System Calls

- System calls provide a way to communicate (input/output)
- Certain system calls use specific data during the execution of the syscall (primarily $v0 and $a0)

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_string	8	$a0 = buffer, $a1 = length	
exit	10		
print_char	11	$a0 = char	
read_char	12		char (in $v0)

11

Examples of System Calls

Assembly Code	Explanation

`# Get an integer from user`
`li $v0, 5`
`syscall`
	# Load immediate system code 5 into $v0
	# System call command → get integer from
	# user returned in $v0

`# Print out the Result`
`li $v0, 1`
`move $a0, $t0`
`syscall`
	# code for print_int
	# put result in $a0
	# print out the result

12

Directives

Directives	Explanation
`.align n`	Align the next datum on a 2^n boundary (Example → .align 2 aligns the next value on a word boundary)
`.ascii str`	Store the string `str` in memory, but do not null terminate it
`.asciiz str`	Store the string `str` in memory and null terminate it
`.data <addr>`	Subsequent items are stored in the data segment. If optional argument `addr` is present, subsequent items are stored starting at addr
`.byte b1, …. bn`	Store the `n` values in successive memory locations
`.globl sym`	Declare the label `sym` is global and can be referenced from other files
`.space n`	Allocate `n` bytes of space in the current segment
`.text <addr>`	Subsequent items are put in the user text segment (instructions)
`.word w1, … wn`	Store the n 32-bit quantities in successive memory words

13

Labels

- Labels notify the program to attach a name to a data declaration or a text (program) line
  - Each data entry must be labeled a unique name or label
  - The specific name "main" MUST be the first line of text
  - Labels are always followed by semicolon
- Examples
  - string: .asciiz "Hello World\n"
  - prompt: .asciiz "Enter your name\n"

14

Hello World Program

```
# This is a simple program to print hello world
# a comment starts with a # till the end of the line
.data          # start putting stuff in the data segment
greet: .asciiz "Hello world\n"
               # declare greet to be the string
.text          #start putting stuff in the text segment
main:          # main is a label here. Names a function
li $v0, 4      # system call code for print_str(sect1.5)
la $a0, greet # address of string to print
syscall        # print the string
```

15

Example 2

- Ask the user for his name and display it back to him!
- Should look like:
  - Please Enter Your Name
  - xxxxxx
  - Your Name is
  - xxxxxx

16

Solution

```
          .data
yourname: .asciiz "Please Enter Your Name\n"
name:     .space 16
nameprint:.asciiz "Your name is\n"


          .globl main
          .text
main:

          li $v0,4
          la $a0,yourname
          syscall

          li $v0,8
          la $a0,name
          li $a1,16
          syscall

          li $v0,4
          la $a0,nameprint
          syscall

          li $v0,4
          la $a0,name
          syscall
```

17

Example 3-4

- From previous example, enter a number after name, add 100 to it and display the number back to the user!
- Sum all numbers from 1 to 10 and display the result

18