

+ Instruction Set for Today...

3

Purpose	Instruction
Add	add dest, src1, src2
Multiply	mul dest, src1, src2
Add Immediate (Literal)	addi dest, src, imm
Branch (Goto)	b label
Branch if equal	beq src1, src2, label
Branch if less than	blt src1, src2, label
Load immediate	li dest, imm

+ A Simple MIPS Program...

4

```
.text
.globl __start
__start:
li $t1, 9
li $t2, 8
add $t3, $t1, $t2
```

+ Quiz: Question 1

5

- Join up the following Java keywords and symbols with the correct type of MIPS instruction.

Java	MIPS Instruction Group
break	Arithmetic and Logical
static final	Constant-Manipulating
Not Applicable	Comparison
<, ==, >	Branch and Jump
Not Applicable	Load
+, <<, %	Store
(double)	Data Movement
throw	Floating Point
.clone()	Exception and Trap

+ Quiz: Question 2

6

- Join up the following Java keywords and symbols with the closest matching MIPS instruction.

Java	MIPS Instruction Group
==	rol
+	rem
.clone()	add
<	slt
	div
%	seq
<<	break
throw	move
/	or

+ Quiz: Question 3

7

- We want to store the literal 4 042 312 908 (a 32 bit number) into a 32 bit register.
 - The binary for this is 11110000 11110000 11001100 11001100.
 - Hint 1: 1111000011110000 = 61680
 - Hint 2: 1100110011001100 = 52428
 - Let's ignore endian issues for now...
 - Remember we can't store it in one go.
 - Erase two of the following lines...

```
lui $1, 61680
lui $1, 52428
ori $2, $1, 52428
addi $2, $1, 52428
```

+ Exercise 1

8

- Write a MIPS assembly language fragment that computes the following:
 - $17 * 9 + 18$
 - $17^2 + 9$

+ Exercise 2

9

- Write a MIPS assembly language fragment that implements the following java blocks:
 - ```
if(a == 9){
 b = a + 1;
}else{
 b = a - 1;
}
```
  - ```
while( b < 19 ){  
    b += 2;  
}
```
 - ```
for(a = 0; a < 10; a++){
 b += 2;
}
```

## + Exercise / Walkthrough 3

10

- Write a program that, given an ASCII string, will detect if it is a palindrome or not.
  - A palindrome is a sentence that reads the same backwards and forwards.
  - Laurent's example: Was it a rat I saw?
  - Only pay attention to letters, ignore the rest.
- It may help to write the program in Java first!
- For simplicity, I'll walk through a basic solution.
  - It does not filter out special characters.

```
1: #####
2: #####
3: # DOES A STRING READ BACKWARDS?
4: # A Partial and Non-Optimal Solution
5: # TONY CHUNG '09
6: #####
7: #####
8:
9:
10: # The strings
11: .data
12: str1: .asciiz "itsti" # Input
13: str2: # Working space
14:
15: .text
16: .globl __START
17: __START:
18: #####
19: #####
20: ## PHASE 1: GET LENGTH OF STRING
21: #####
22:
23: # t0: counter
24: # t1: pointer
25: # t2: byte holder
26:
27: # AIM: Keep counting until byte is equal to zero
28:
29: #init count
30: li $t0, 0
31: #get addr of string
32: la $t1, str1
33:
34: test1:
35: #while not last char
36:
37: #get byte
38: lb $t2, ($t1)
39:
40: #test for null
41: beqz $t2, storei
42:
43: loop1:
44: #increment pointer and counter
45: addi $t0, $t0, 1
46: addi $t1, $t1, 1
47: b test1
48:
49: #####
50: ## PHASE 2: COPY INVERSE STRING
51: #####
52:
53: # t0: counter (preset to length)
54: # t1: pointer for string 1 (preset to last char + 1)
55: # t3: size of string
56: # t4: pointer for string 2
57: # t5: temporary byte holder
58:
59: storei:
60: #copy counter to size
61: move $t3, $t0
62:
63: #get position of first char in str2
64: la $t4, str2
65:
```

```
66: test2:
67: #while not first character
68: beqz $t0, compare
69:
70: loop2:
71: #decrement string 1 pointer (do first as +1)
72: addi $t1, $t1, -1
73:
74: #copy current character to new position
75: lb $t5, ($t1)
76: sb $t5, ($t4)
77:
78: #increment string 2 pointer
79: addi $t4, $t4, 1
80:
81: #decrement counter
82: addi $t0, $t0, -1
83:
84: b test2
85:
86:
87: #####
88: ## PHASE 3: COMPARE EACH CHAR
89: #####
90:
91: # t0: counter
92: # t1: pointer for string 1
93: # t3: size of string (preset)
94: # t4: pointer for string 2
95: # t6: byte from string 1
96: # t7: byte from string 2
97:
98: compare:
99:
100: #goto start of both strings and reset counter
101: la $t1, str1
102: la $t4, str2
103: li $t0, 0
104:
105:
106: test3:
107: #while not at end
108: beq $t0, $t3, equal
109:
110: loop3:
111:
112: #get chars
113: lb $t6, ($t1)
114: lb $t7, ($t4)
115:
116: #increment pointers and counters (do before test!)
117: addi $t1, $t1, 1 #string 1
118: addi $t4, $t4, 1 #string 2
119: addi $t0, $t0, 1 #counter
120:
121: #test (continue if equal, else not equal)
122: beq $t6, $t7, test3
123:
124:
125: b notequal
126:
127:
128: #####
129: ## PHASE 4: OUTPUT RESULT
130: #####
```

```
131:
132: # a0: result (1 for true, 0 for false)
133: # v0: syscall command
134:
135:
136: equal:
137: li $a0, 1
138: li $v0, 1
139: syscall
140:
141: li $v0, 10
142: syscall
143:
144: notequal:
145: li $a0, 0
146: li $v0, 1
147: syscall
148:
149: li $v0, 10
150: syscall
```