# BONUS SLIDES

You are responsible for the material contained on the following slides, though we may not have enough time to get to them in lecture.

They have been prepared in a way that should be easily readable.

### Agenda

- I-Format
  - Branching and PC-Relative Addressing
- Administrivia
- J-Format
- Pseudo-instructions
- Bonus: Assembly Practice
- Bonus: Disassembly Practice

### **Assembly Practice**

- Assembly is the process of converting assembly instructions into machine code
- On the following slides, there are 6-lines of assembly code, along with space for the machine code
- For each instruction,
  - 1) Identify the instruction type (R/I/J)
  - 2) Break the space into the proper fields
  - Write field values in decimal
  - 4) Convert fields to binary
  - 5) Write out the machine code in hex
- Use your Green Sheet; answers follow

# **Code Questions**

Addr Inst	ruction	Material from past lectures:	
800 Loop	: sll \$t1,\$s3,2	What type of C variable is probably stored in \$s6?	
804 addu	\$t1,\$t1,\$s6	Write an equivalent C loop using	
808 lw	\$t0,0(\$t1)	a→\$s3, b→\$s5, c→\$s6. Define variable types (assume they are initialized somewhere) and feel	
812 beq	\$t0,\$s5, Exit	free to introduce other variables as you like.	
816 addi	u \$s3,\$s3,1		
820 ј	Loop	In English, what does this loop do?	
Exit	:		

# **Code Answers**

Addr	Instru	action	Material from past lectures:	
800	Loop:	sll \$t1,\$s3,2	What type of C variable is probably stored in \$s6? int * (or any pointer)	
804	addu	\$t1,\$t1,\$s6		
808	lw	\$t0,0(\$t1)	Write an equivalent C loop using a→\$s3, b→\$s5, c→\$s6. Define variable types (assume they are initialized somewhere) and feel	
812	beq	\$t0,\$s5, Exit	free to introduce other variables as you like.	
816	addiu	\$s3,\$s3,1	int a,b,*c; /* values initialized */ while(c[a] != b) a++;	
820	j	Loop	In English, what does this loop do? Finds an entry in array c that	
	Exit:	Spring 2012 Locture	matches b.	

## **Assembly Practice Question**

```
Addr
       Instruction
      Loop: sl<u>l</u> $t1,$s3,2
 0.08
              $t1,$t1,$s6
 804
       addu
              $t0,0($t1)
 808
       lw
 812
              $t0,$s5, Exit
      beq
       addiu $s3,$s3,1
 816
 820
              Loop
```

Exit:

# Assembly Practice Answer (1/4)

### Addr Instruction

```
Loop: sll $t1,$s3,2
008
  opcode
                                  shamt
                                          funct
                     rt
                            rd
             rs
            $t1,$t1,$s6
    addu
804
  opcode
                                  shamt
                                          funct
                    rt
             rs
                            rd
            $t0,0($t1)
808
    lw
  opcode
                                immediate
             rs
                    rt
            $t0,$s5, Exit
812
     beq
  opcode
                                immediate
                    rt
             rs
    addiu $s3,$s3,1
816
  opcode
                                immediate
             rs
                    rt
820
            Loop
  opcode
                      target address
     Exit:
```

### Assembly Practice Answer (2/4)

#### Instruction Addr Loop: sll \$t1,\$s3,2 800 R: 19 9 804 addu \$t1,\$t1,\$s6 R: 22 33 9 \$t0,0(\$t1) 808 lw 35 \$t0,\$s5, Exit 812 beq 816 addiu \$s3,\$s3,1 19 820 Loop 200

Exit:

### Assembly Practice Answer (3/4)

### Addr Instruction

```
Loop: sll $t1, $s3,2
800
                                   00010
   00000
                                           00000
            00000
                   10011
                           01001
            $t1,$t1,$s6
     addu
804
   00000
            01001
                   10110
                           01001
                                   00000
                                           100001
            $t0,0($t1)
808
     ] w
    00011
                            0000
                                 0000
                                       0000
                                             0000
                   01000
            $t0,$s5, Exit
812
     beq
   000100
                   10101
                            0000
                                 0000
                                       0000
                                             0010
     addiu $s3,$s3,1
   001000
                   10011
                            0000
                                 0000
                                       0000
                                             0001
820
            Loop
   000010
            00 0000
                     0000
                           0000
                                 0000
                                       1100
     Exit:
```

### Assembly Practice Answer (4/4)

```
Addr Instruction
800 Loop: sll $t1,$s3,2
R:
    0x 0013 4880
804 addu $t1,$t1,$s6
R: 0x 0136 4821
808 lw $t0,0($t1)
l: 0x 8D28 0000
812 beg $t0,$s5, Exit
l: 0x 1115 0002
816 addiu $s3,$s3,1
l: 0x 2273 0001
820 j Loop
J: 0x 0800 00C8
     Exit:
```

### Agenda

- I-Format
  - Branching and PC-Relative Addressing
- Administrivia
- J-Format
- Pseudo-instructions
- Bonus: Assembly Practice
- Bonus: Disassembly Practice

### **Disassembly Practice**

- Disassembly is the opposite process of figuring out the instructions from the machine code
- On the following slides, there are 6-lines of machine code (hex numbers)
- Your task:
  - 1) Convert to binary
  - 2) Use opcode to determine format and fields
  - 3) Write field values in decimal
  - 4) Convert fields MIPS instructions (try adding labels)
  - 5) Translate into C (be creative!)
- Use your Green Sheet; answers follow

### Disassembly Practice Question

### Address Instruction

0x00400000

. . .

0x00001025

0x0005402A

0x11000003

0x00441020

0x20A5FFFF

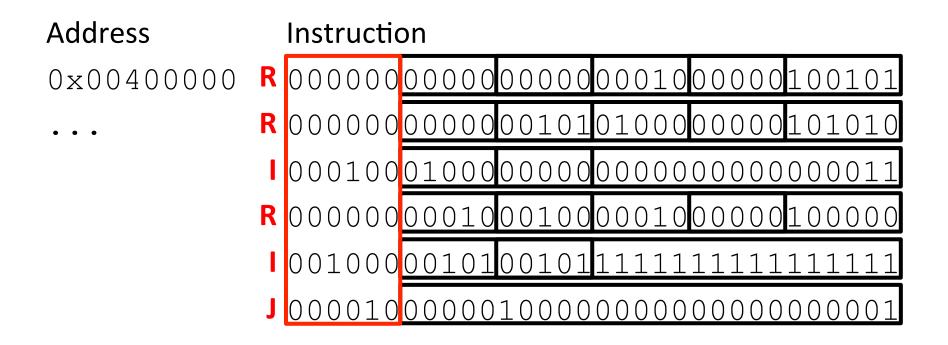
0x08100001

### Disassembly Practice Answer (1/9)

### 

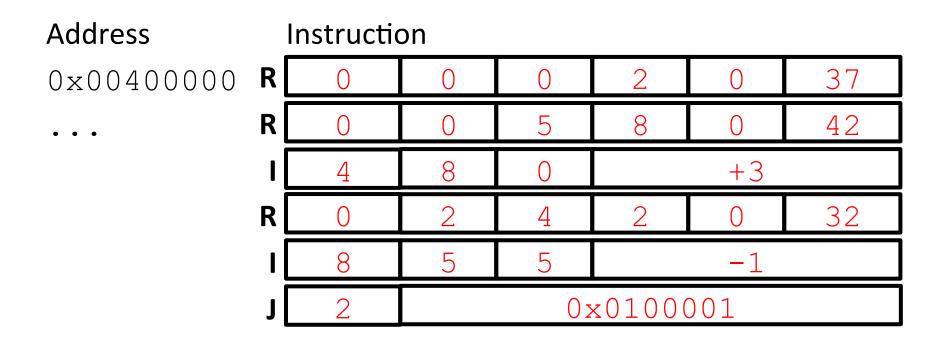
### 1) Converted to binary

### Disassembly Practice Answer (2/9)



- 2) Check opcode for format and fields...
  - 0 (R-Format), 2 or 3 (J-Format), otherwise (I-Format)

# Disassembly Practice Answer (3/9)



- 3) Convert to decimal
  - Can leave target address in hex

# Disassembly Practice Answer (4/9)

Address	Instruc	ction
0x00400000	or	\$2,\$0,\$0
0x00400004	slt	\$8,\$0,\$5
0x00400008	beq	\$8,\$0,3
0x0040000C	add	\$2,\$2,\$4
0x00400010	addi	\$5,\$5,-1
0x00400014	j	0x0100001
0x00400018		

4) Translate to MIPS instructions (write in addrs)

# Disassembly Practice Answer (5/9)

Address	Instruc	ction	
0x00400000	or	\$v0,\$0,\$0	
0x00400004	slt	\$t0,\$0,\$a1	
0x00400008	beq	\$t0,\$0,3	
0x0040000C	add	\$v0,\$v0,\$a0	
0x00400010	addi	\$a1,\$a1,-1	
0x00400014	j	0x0100001 # addr:	0x0400004
0x00400018			

- 4) Translate to MIPS instructions (write in addrs)
  - More readable with register names

### Disassembly Practice Answer (6/9)

#### Address Instruction 0x00400000 \$v0,\$0,\$0 or Loop: slt \$t0,\$0,\$a1 0x00400004 0x00400008 beq \$t0,\$0,Exit add \$v0,\$v0,\$a0 $0 \times 0.040000$ C 0x00400010 addi \$a1,\$a1,-1 j 0x00400014 Loop 0x00400018 Exit:

- 4) Translate to MIPS instructions (write in addrs)
  - Introduce labels

### Disassembly Practice Answer (7/9)

```
Address Instruction

or $v0,$0,$0 # initialize $v0 to 0

Loop: slt $t0,$0,$a1 # $t0 = 0 if 0 >= $a1

beq $t0,$0,Exit # exit if $a1 <= 0

add $v0,$v0,$a0 # $v0 += $a0

addi $a1,$a1,-1 # decrement $a1

j Loop
```

#### Exit:

- 4) Translate to MIPS instructions (write in addrs)
  - What does it do?

### Disassembly Practice Answer (8/9)

```
/* a→$v0, b→$a0, c→$a1 */
a = 0;
while(c > 0) {
   a += b;
   c--;
}
```

- 5) Translate into C code
  - Initial direct translation

### Disassembly Practice Answer (9/9)

```
/* naïve multiplication: returns m*n */
int multiply(int m, int n) {
  int p; /* product */
  for(p = 0; n-- > 0; p += m) ;
  return p;
}
```

- 5) Translate into C code
  - One of many possible ways to write this