February 1

- Today's tune brought to you courtesy of Kalena Kelly.
- Maxima demonstration.
- READ THE BOOK!
- If you're not getting it let me know IN CLASS.
- Assignment 4 on the web. START EARLY

- Slides corrected after class

Execution Example

Program Counter

200

Memory(32 bits)

200	1000110100001001 0000000000000000	**LW $9, 0($8)**
204	0000000100100111 0100100000100000	**ADD $9,$9,$7**
208	1010110100001001 0000000000001000	**SW $9, 8($8)**
212		

Memory(32 bits)

112	**8**
116	**13**
120	**21**
124	**34**
128	**55**
132	**89**

Registers (32 bits)

6	**1234**
7	**23**
8	**120**
9	**-314159**
10	**316**

Instruction Register (32 bits)

R

op	rs	rt	rd	shft	func
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

I

op	rs	rt	offset
6 bits	5 bits	5 bits	16 bits

Execution Example: Fetch(200)

Program Counter

200

Memory

200	1000110100001001 0000000000000000	**LW $9, 0($8)**
204	0000000100100111 0100100000100000	**ADD $9,$9,$7**
208	1010110100001001 0000000000001000	**SW $9, 8($8)**
212		

Memory

112	**8**
116	**13**
120	**21**
124	**34**
128	**55**
132	**89**

Registers

6	**1234**
7	**23**
8	**120**
9	**-314159**
10	**316**

Instruction Register

R						
I	**100011**	**01000**	**01001**	**0000000000000000**		
	35	**8**	**9**	**0**		

Execution Example: Execute(200)

Program Counter

204

Memory

200	1000110100001001 0000000000000000	**LW $9, 0($8)**
204	0000000100100111 0100100000100000	**ADD $9,$9,$7**
208	1010110100001001 0000000000001000	**SW $9, 8($8)**
212		

Memory

112	**8**
116	**13**
120	**21**
124	**34**
128	**55**
132	**89**

Registers

6	**1234**
7	**23**
8	**120**
9	**21**
10	**316**

Instruction Register

R

I

100011	**01000**	**01001**	**0000000000000000**
35	**8**	**9**	**0**

Execution Example: Fetch(204)

Program Counter

204

Memory

200	1000110100001001 0000000000000000	**LW $9, 0($8)**
→ 204	0000000100100111 0100100000100000	**ADD $9,$9,$7**
208	1010110100001001 0000000000001000	**SW $9, 8($8)**
212		

Memory

112	**8**
116	**13**
120	**21**
124	**34**
128	**55**
132	**89**

Registers

6	**1234**
7	**23**
8	**120**
9	**21**
10	**316**

Instruction Register

R	**000000**	**01001**	**00111**	**01001**	**00000**	**100000**
	0	**9**	**7**	**9**	**0**	**32**
I						

Execution Example: Execute(204)

Program Counter

208

Memory

200	1000110100001001 0000000000000000	**LW $9, 0($8)**
→ 204	0000000100100111 0100100000100000	**ADD $9,$9,$7**
208	1010110100001001 0000000000001000	**SW $9, 8($8)**
212		

Memory

112	**8**
116	**13**
120	**21**
124	**34**
128	**55**
132	**89**

Registers

6	**1234**
7	**23**
8	**120**
9	**44**
10	**316**

Instruction Register

R	**000000**	**01001**	**00111**	**01001**	**00000**	**100000**
	0	**9**	**7**	**9**	**0**	**32**
I						

Execution Example: Fetch(208)

Program Counter

208

Memory

200	1000110100001001 0000000000000000	**LW $9, 0($8)**
204	0000000100100111 0100100000100000	**ADD $9,$9,$7**
→ 208	1010110100001001 0000000000001000	**SW $9, 8($8)**
212		

Memory

112	**8**
116	**13**
120	**21**
124	**34**
128	**55**
132	**89**

Registers

6	**1234**
7	**23**
8	**120**
9	**44**
10	**316**

Instruction Register

R						
I	**101011**	**01000**	**01001**	**0000000000001000**		
	43	**8**	**9**	**8**		

Execution Example: Execute(208)

Program Counter

212

Memory

200	1000110100001001 0000000000000000	**LW $9, 0($8)**
204	0000000100100111 0100100000100000	**ADD $9,$9,$7**
→ 208	1010110100001001 0000000000001000	**SW $9, 8($8)**
212		

Memory

112	**8**
116	**13**
120	**21**
124	**34**
128	**44**
132	**89**

Registers

6	**1234**
7	**23**
8	**120**
9	**44**
10	**316**

Instruction Register

R						
I	**101011**	**01000**	**01001**	**0000000000001000**		
	43	**8**	**9**	**8**		

Control

- Decision making instructions
  - alter the control flow,
  - i.e., change the "next" instruction to be executed

- MIPS conditional branch instructions:

```
bne $t0, $t1, Label
beq $t0, $t1, Label
```

- Example: if (i==j) h = i + j;

```
        bne $s0, $s1, Label
        add $s3, $s0, $s1
Label: ....
```

Control

- MIPS unconditional branch instructions:

```
j  label
```

- Example:

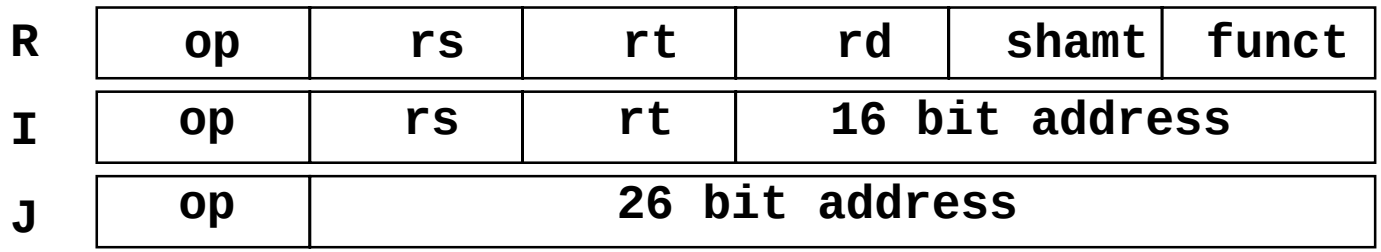
```
if (i!=j)              beq $s4, $s5, Lab1
    h=i+j;             add $s3, $s4, $s5
else                   j Lab2
    h=i-j;             Lab1:  sub $s3, $s4, $s5
                       Lab2:  ...
```

So far:

- Instruction — Meaning

```
add $s1,$s2,$s3   $s1 = $s2 + $s3
sub $s1,$s2,$s3   $s1 = $s2 – $s3
lw $s1,100($s2)   $s1 = Memory[$s2+100]
sw $s1,100($s2)   Memory[$s2+100] = $s1
bne $s4,$s5,L     Next instr. is at Label if $s4 != $s5
beq $s4,$s5,L     Next instr. is at Label if $s4 = $s5
j Label           Next instr. is at Label
```

- Formats:



Control Flow

- We have: beq, bne, what about Branch-if-less-than?
- New instruction:

```
slt $t0, $s1, $s2
```

```
if  $s1 < $s2 then
   $t0 = 1
else
   $t0 = 0
```

- Can use this instruction to build "`blt $s1, $s2, Label`"
  — can now build general control structures
- Note that the assembler needs a register to do this,
  — there are policy of use conventions for registers

Policy of Use Conventions

Name	Register number	Usage
`$zero`	0	the constant value 0
`$v0-$v1`	2-3	values for results and expression evaluation
`$a0-$a3`	4-7	arguments
`$t0-$t7`	8-15	temporaries
`$s0-$s7`	16-23	saved
`$t8-$t9`	24-25	more temporaries
`$gp`	28	global pointer
`$sp`	29	stack pointer
`$fp`	30	frame pointer
`$ra`	31	return address