Name _____

Computer Architecture
EE 4720
Midterm Examination Two

Wednesday, 19 November 2003, 10:40–11:30 CST

Problem 1	_____	(50 pts)
Problem 2	_____	(50 pts)
Exam Total	_____	(100 pts)

Alias _____

Good Luck!

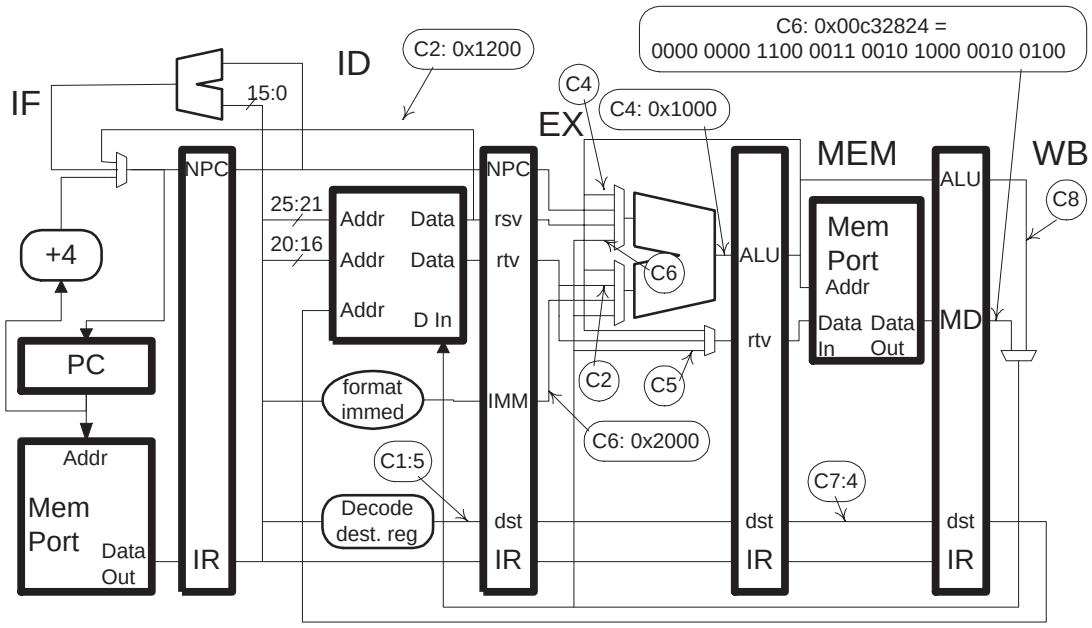
Problem 1: In the diagram below some wires are labeled with cycle numbers and values that will then be present. For example, C6:9 indicates that at cycle 6 the pointed-to wire will hold an 9. Other wires are labeled just with cycle numbers, indicating that the wire is used at that cycle. There are no stalls during the execution of the code.[50 pts]

☐ Write a program consistent with these labels.

☐ Show the address of every instruction.

☐ Show every register number that can be determined and use `r10`, `r11`, etc. for other register numbers.

☐ One immediate value and two register numbers are found in interesting ways.



Insn Addr	Cycle: 0	1	2	3	4	5	6	7	8
0x1000:	IF	ID	EX	ME	WB				
		IF	ID	EX	ME	WB			
			IF	ID	EX	ME	WB		
				IF	ID	EX	ME	WB	
					IF	ID	EX	ME	WB
	Cycle: 0	1	2	3	4	5	6	7	8

Problem 2: On the next page is a MIPS pipeline that implements the post-increment load instruction from Homework 4. In the code below the post-increment loads load the data at the address in `r2` into destination register `r1`. They then store the address +4 (the address of the next word) in `r2`.

The implementation on the next page uses a register file with only one write port, just like in Homework 4. The loaded value gets written back in the post-increment load's writeback cycle, the incremented address normally gets written back in the writeback cycle of the next instruction that does not itself perform write-back. That may be the very next instruction, or several instructions later. Or never (see the cases below). [50 pts]

(*a*) Design the control hardware for the five **bold** multiplexors (one in EX, two in WB, and two for the new pipeline latch) so that the implementation correctly executes as many cases below as possible with as few stall cycles as possible. Do not worry about exceptions.

(*b*) Design the logic to generate the stall signal needed for at least one case below and for similar cases. Stall in as few cases as possible.

```
# Pipeline segments provided for convenience.  In at least
# one case stalls will be necessary, AND THOSE STALLS ARE NOT SHOWN.
# In examples focus on r1-r9.

# Case 1:  Ordinary case, use WB of branch.
lw r1, (r2)+             IF ID EX ME WB
add r12, r11, r12           IF ID EX ME WB
bne r15,r16 SOMEWHERE          IF ID EX ME WB
add r12, r11, r12                 IF ID EX ME WB

# Case 2:  Register r2 not needed.
lw r1, (r2)+             IF ID EX ME WB
add r12, r11, r12           IF ID EX ME WB
add r2, r14, r15               IF ID EX ME WB
add r18, r2, r19                  IF ID EX ME WB

# Case 3:  Bypass needed.
lw r1, (r2)+             IF ID EX ME WB
add r12, r11, r12           IF ID EX ME WB
add r12, r11, r12              IF ID EX ME WB
add r12, r11, r2                  IF ID EX ME WB
bne r15,r16 SOMEWHERE                IF ID EX ME WB
add r12, r11, r12                        IF ID EX ME WB

# Case 4:  Notice that base address registers different. (r2, r4)
lw r1, (r2)+             IF ID EX ME WB
lw r3, (r4)+                IF ID EX ME WB
add r12, r11, r12              IF ID EX ME WB
add r12, r11, r12                 IF ID EX ME WB
bne r15,r16 SOMEWHERE                IF ID EX ME WB
add r12, r11, r12                        IF ID EX ME WB

# Case 5:  Make sure this works correctly.
lw r1, (r2)+             IF ID EX ME WB
add r12, r11, r12           IF ID EX ME WB
add r12, r11, r12              IF ID EX ME WB
add r2, r13, r14                  IF ID EX ME WB
add r18, r2, r19                     IF ID EX ME WB
```

Problem 2, continued:

- The control logic can be in any stage, not just ID.
- The logic is relatively simple, it should easily fit on the page.