Name _____

Computer Architecture
EE 4720
Midterm Examination

Wednesday, 29 March 2006, 11:40–12:30 CST

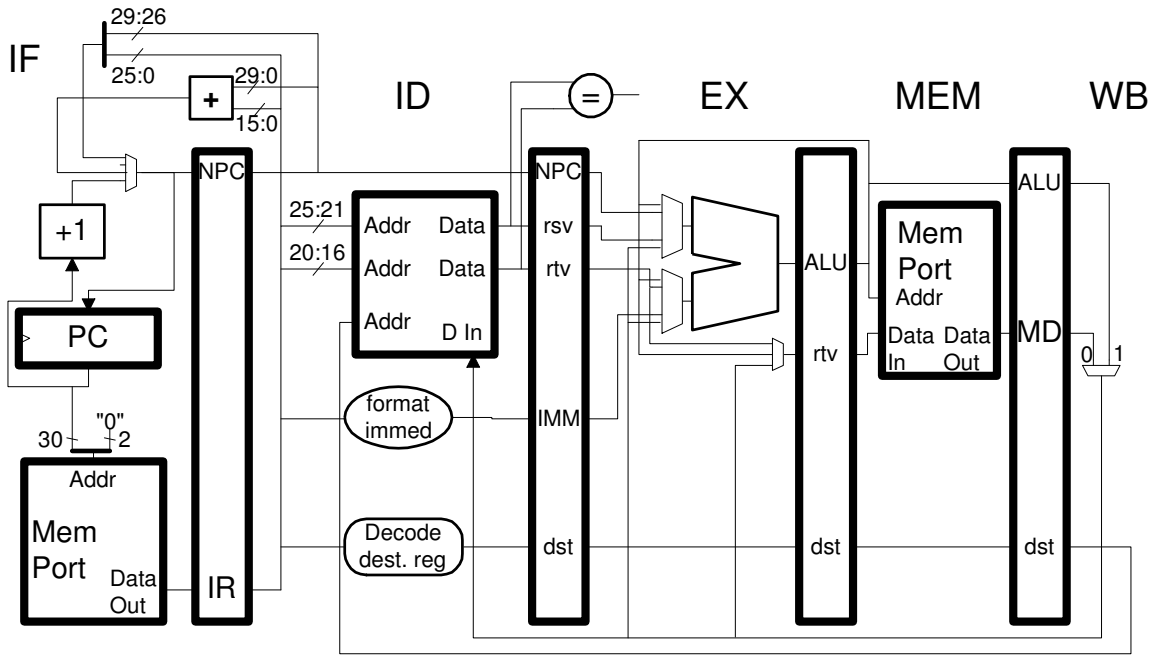
Problem 1	_____	(10 pts)
Problem 2	_____	(40 pts)
Problem 3	_____	(50 pts)
Exam Total	_____	(100 pts)

Alias _____

Good Luck!

Problem 1: The MIPS code below runs on the illustrated implementation. [10 pts]

☐ Show the execution of the code below on the illustrated pipeline.




```
# Cycle          0   1   2   3   4   5   6   7   8

lw r1, 0(r2)

add r1, r1, 7

sw r1, 0(r2)

# Cycle          0   1   2   3   4   5   6   7   8
```

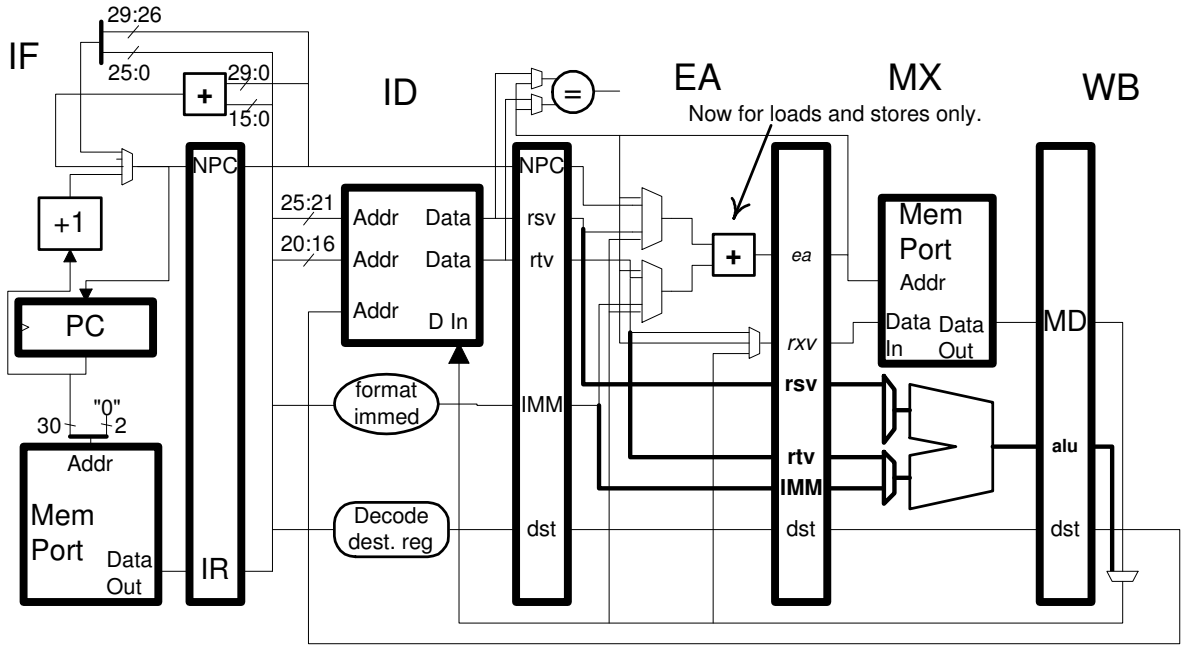
Note: To keep the test from getting too long the following parts were not included on the original exam. There should be at least one stall.

Think about the questions below for a few moments, then look at the problem on the next page.

☐ Is it possible to add bypass connections to eliminate the stall and gain performance?

☐ If bypasses won't work does that mean it's impossible to eliminate the stall?

Problem 2: The five-stage MIPS implementation below has an ALU in the MX (new name for MEM) stage (connections for that ALU are shown bold) and what was the ALU in the EA (new name for EX) stage is now just an adder and is to be used only for loads and stores.



(*a*) Add bypass connections needed so that the code below (same as last problem) executes as shown. Label those bypass connections: LAS-*c*, where *c* is the cycle number in which the bypass connection is used. Do not add unneeded bypass connections!

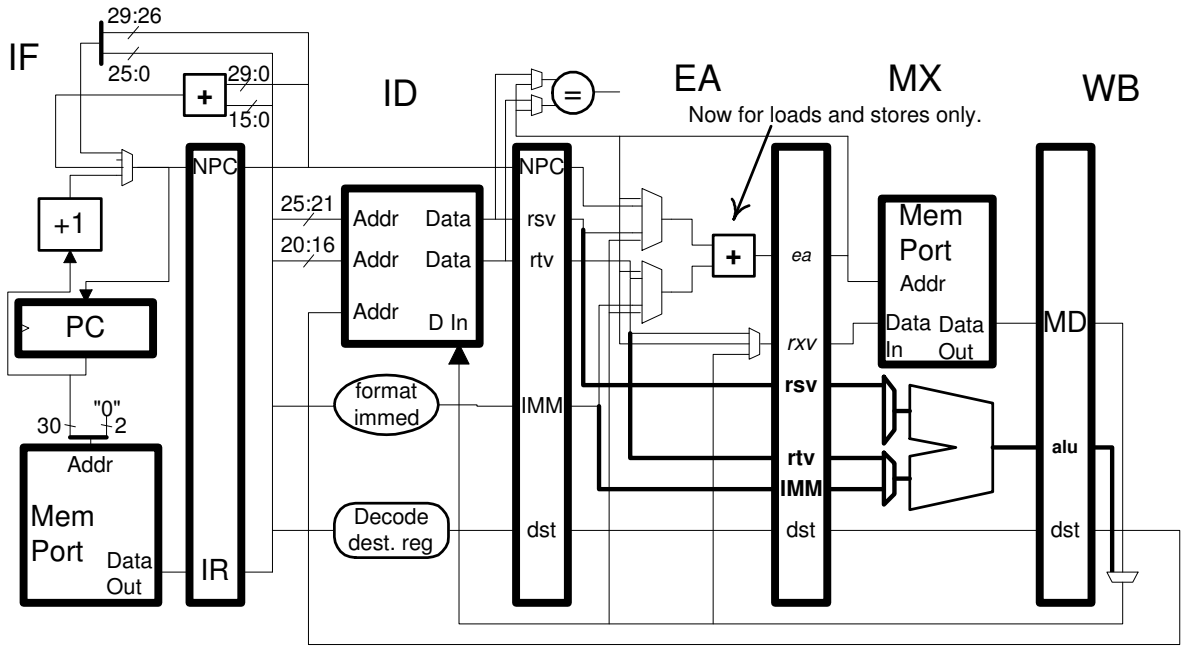
☐ [10 pts] Add bypass connections, label with LAS-*c*.

```
# Cycle          0  1  2  3  4  5  6
lw r1,0(r2)      IF ID EA MX WB
addi r1, r1, 7      IF ID EA MX WB
sw r1, 0(r2)           IF ID EA MX WB
# Cycle          0  1  2  3  4  5  6
```

(*b*) Several connections to the EA-stage adder are now unneeded (but were needed when it was an ALU). (Don't add new connections here.)

☐ [10 pts] Label unneeded EA-adder connections with an X at mux inputs.

Problem 2, continued: The implementation below is identical to the one on the previous page.



(*c*) For each code fragment below: Add reasonable bypass connections to eliminate stalls (if any). Show the execution and label the bypass connections that are used (new or existing) with a cycle number. If there is a stall circle Yes if an EX-stage ALU (in addition to the ME stage ALU) would remove the stall, if an EX-stage ALU wouldn't help circle No; if there is no stall don't circle anything. [20 pts]

☐ Add bypasses (if needed). Label bypasses used AS-*c*. Show execution. EX ALU would help: Yes No.

```
# Cycle          0   1   2   3   4   5   6   7   8
add r1, r2, r3   IF  ID  EA  MX  WB

sub r4, r1, r5
```

☐ Add bypasses (if needed). Label bypasses used AL-*c*. Show execution. EX ALU would help: Yes No.

```
# Cycle          0   1   2   3   4   5   6   7   8
add r1, r2, r3   IF  ID  EA  MX  WB

lw r4, 4(r1)
```

☐ Add bypasses (if needed). Label bypasses used LL-*c*. Show execution. EX ALU would help: Yes No.

```
# Cycle          0   1   2   3   4   5   6   7   8
lw r1, 12(r2)    IF  ID  EA  MX  WB

lw r3, 16(r1)
```

☐ Add bypasses (if needed). Label bypasses used AB-*c*. Show execution. EX ALU would help: Yes No.

```
# Cycle          0   1   2   3   4   5   6   7   8
addi r1, r1, 1   IF  ID  EA  MX  WB

beq r1,r2 TARG

nop
```

Problem 3: Answer each question below.

(*a*) A computer's scores on SPECint2000 are baseline, 2011; and result (peak) 2014. These baseline and result scores are close, who might be responsible and should they be proud or ashamed: [10 pts]

☐ How might the people who conducted the test be responsible for the close scores? Should they be proud or ashamed?

☐ How might the people who wrote the compiler be responsible for the close scores? Should they be proud or ashamed?

(*b*) On a SPARC system the handler for trap number 4 is located at address `0xa0001000` and is 100 instructions long. [10 pts]

☐ Show the Trap Base Register (TBR) value and Trap Table contents needed so that execution of a trap instruction will reach this handler. (Don't worry about returning.)

☐ Why isn't a user program allowed to call the handler directly, for example, using the ordinary call instruction `call 0xa0001000`. (The SPARC call has a 30-bit immediate field so the target address is not too far away.)

☐ What would happen if the user tried to execute the call instruction above?

Problem 3, continued:

(*c*) Packed integer and BCD data types are very superficially similar. [10 pts]

☐ Describe a situation in which a packed integer data type would be useful.

☐ Describe a situation in which BCD would be useful.

(*d*) Describe the contents of a typical bundle in a VLIW ISA. [10 pts]

☐ Bundle contents.

(*e*) Which is it more important to do a good job on, the ISA or the implementation? Explain. [10 pts]

☐ Good job on ISA or implementation? Explain.