Name ________________________________

Computer Architecture
EE 4720
Midterm Examination

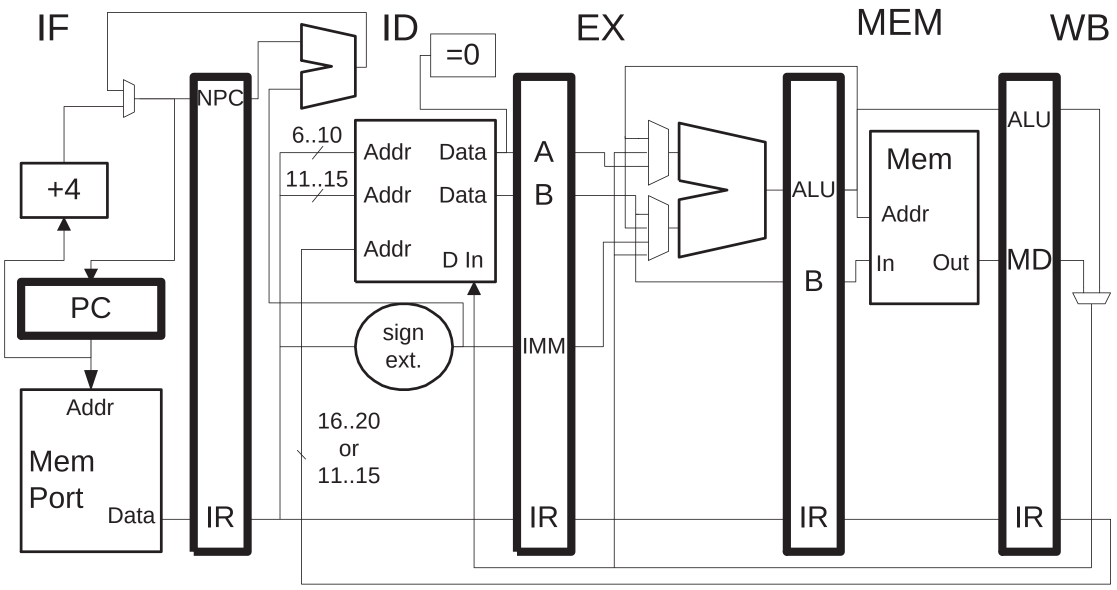
12 March 1999, 10:40–11:30 CST

Problem 1	________	(25 pts)
Problem 2	________	(30 pts)
Problem 3	________	(45 pts)
Exam Total	________	(100 pts)

Alias ________________________________

Good Luck!

Problem 1: Design control logic to generate the control signal for the multiplexor at the lower input to the ALU. The control logic should be located in the ID stage and should generate a two-bit integer for the multiplexor. The integer specifies which multiplexor input to use, they are numbered from zero starting at the top. (Input 0 connects to `ID/EX.B`, 1 connects to `EX/MEM.ALU`, etc.) The logic can use units that test for equality of their two inputs, $\boxed{=}$, and units that test for instruction formats, $\boxed{\texttt{= Type I}}$, $\boxed{\texttt{= Type R}}$, and $\boxed{\texttt{= Type J}}$, and can use the usual logic gates. Base the setting on instruction type, rather than the exact opcode. Show how the control signal is connected to the multiplexor. (25 pts)



Problem 2: Consider the code below.

```
LOOP:
 lw   r1, 0(r2)
 addi r3, r1, #12
 sw   4(r2), r3
 add  r5, r5, r1
 addi r2, r2, #8
 slt  r6, r2, r7
 bneq r6, LOOP
 xor  r8, r9, r10
```

(*a*) Show a pipeline execution diagram for execution up to the second time `lw` enters instruction fetch. Use the pipeline from problem 1. As with homework 3, a bypass path is unavailable if it's not shown. What is the CPI for a large number of iterations? (10 pts)

(*b*) Unroll the loop so that two iterations of the code above is performed by one iteration of the unrolled loop. (Assume the number of iterations in the original loop is a multiple of two.) Schedule the unrolled loop to minimize stalls. (10 pts)

(*c*) What is the CPI of the unrolled and scheduled loop found above? What conclusions about performance improvement can and cannot be made by comparing the CPI of the original and unrolled loop? What is the performance improvement? (Give a number for performance improvement, don't just say "it's good.") (10 pts)

Problem 3: Answer each question below.

(*a*) Show an example of DLX code that encounters a WAW hazard on the Chapter-3 implementation of DLX (in which the multiply floating-point functional unit has an initiation interval of 1 and a latency of 6 and the add floating-point functional unit has an initiation interval of 1 and a latency of 3) but which does not encounter a WAW hazard on a DLX implementation which is identical except the FP add latency is 1 and the FP multiply latency is 4. The code should not encounter a structural hazard on either implementation. (12 pts)

(*b*) Why are there separate `lh` (load half) and `lhu` (load half unsigned) instructions, a `sh` (store half) instruction but **no** `shu` (store half unsigned) instruction? (11 pts) (*c*) The code below is for a stack architecture. Rewrite the code below in DLX using as few instructions as possible. Assume that ADDRA is in r10, ADDRB is in r11, ADDRX is in r12, and ADDRY is in r13. The data at the addresses are double-precision floating-point values. (11 pts)

```
push ADDRX
push ADDRX
mult
push ADDRA
push ADDRB
add
push ADDRX
mult
push ADDRA
push ADDRB
mult
add
add
pop ADDRY
```

(*d*) Explain two ways in which precise exceptions can be made optional for floating-point instructions. That is, the programmer may choose to have precise exceptions where they are needed or may choose to not have precise exceptions were performance is most important. Explain what the programmer would have to do for each of the two ways. (11 pts)