Name ______________________________

Computer Architecture
EE 4720
Midterm Examination

14 March 1997, 12:40–13:30 CST

Problem 1	________	(30 pts)
Problem 2	________	(30 pts)
Problem 3	________	(40 pts)
Exam Total	________	(100 pts)

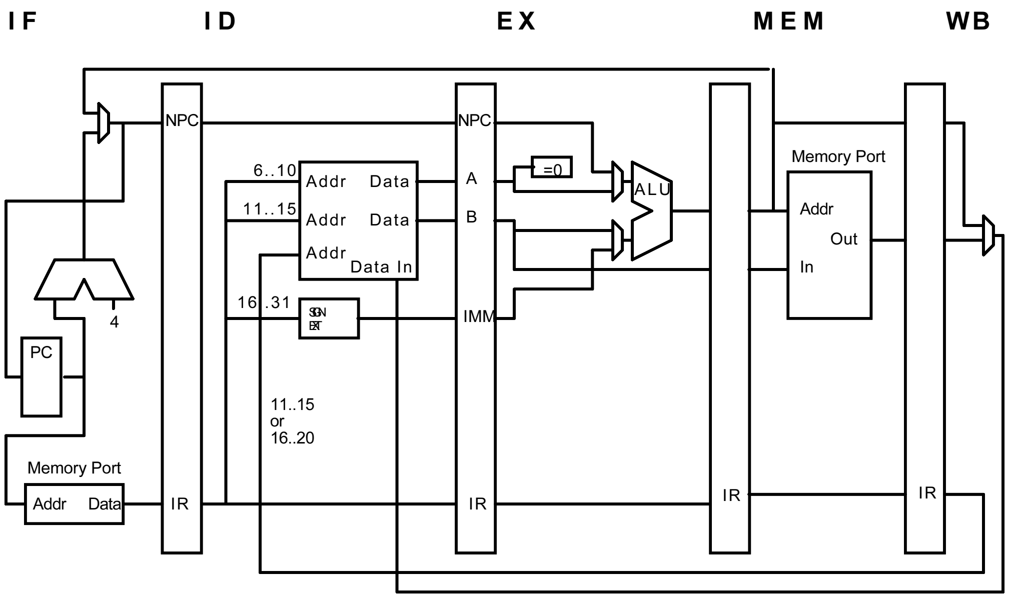
Alias ______________________________

Good Luck!

Problem 1: Consider a new instruction, `mems r1,(r2)`, which loads and examines consecutive memory words, starting at the address stored in r2, until a word equal to the contents of r1 is loaded. When the instruction finishes, r2 will have the address of the matching word. (The loaded words are not written to registers, although they might be temporarily stored somewhere.) For example, if the contents of r1 was 123 and the contents of r2 was 1000, and a 123 were stored at memory location 1016 (and not at memory locations before 1016), then after `mems r1,(r2)` executed there would be a 1016 in r2.

(*a*) Modify the pipeline to implement this instruction and show its execution on the modified pipeline (ID, EX, etc.). Show only the data connections on the modified pipeline, not control or changes in the IR path. When showing the execution of `mems` briefly explain what is happening in each stage. The implementation can be either low cost or high speed. (10 pts)

(Hint: first do part b. Then returning to this part, try to figure out how the instruction might execute. Use this to modify the pipeline and to solve part c. Even if the execution is based on a wild guess, it can still be used for part c.)



Problem 1, continued.

(*b*) Write a DLX program that performs the same function as `mems`. (10 pts)

(*c*) Compare the time needed to execute mems to the time needed to run the program when the word is found after 1000 loads. (10 pts)

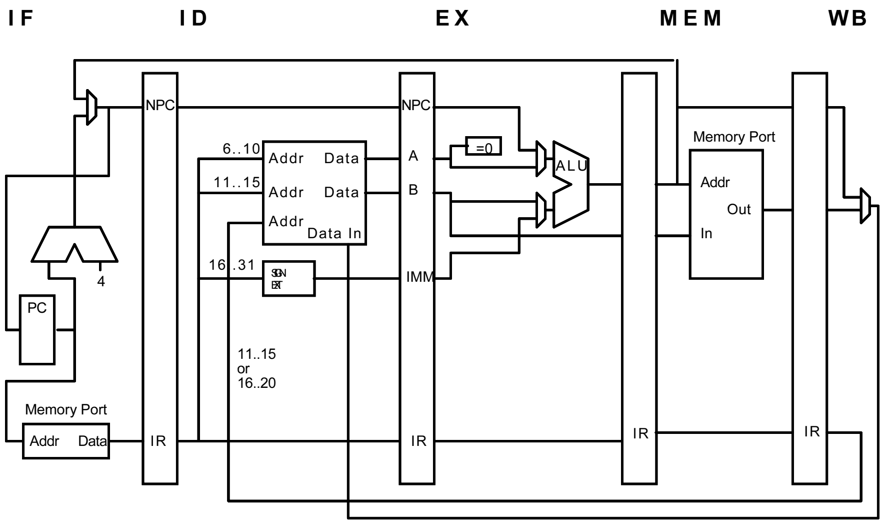
Reference Information:

Instruction type/opcode	**Instruction Meaning**
LW, SW	Load word, store word (to/from integer registers)
ADD, ADDI, ADDU, ADDUI	Add, add immediate (all immediates are 16 bits); signed and unsigned
AND, ANDI	And, and immediates
LHI	Load high immediate—loads upper half of register with immediate
S__	Set conditional: "__" may be `LT, GT, LE, GE, EQ, NE`
BEQZ, BNEZ	Branch GPR equal/not equal to zero; 16-bit offset from PC+4
J, JR	Jumps: 26-bit offset from PC+4 `(J)` or target in register `(JR)`
JAL, JALR	Jump and link: save PC+4 in R31, target is PC-relative `(JAL)` or a register `(JALR)`

Problem 2: Suppose arithmetic instructions skipped the MEM stage, going straight from EX to WB, saving a cycle.

(*a*) Because the register memory has only one write port, this scheme introduces a structural hazard in the WB stage. Find a sequence of instructions that encounter the hazard, show how they would execute (IF ID, etc.) if nothing were done about the hazard, and show where the hazard occurs. (7 pts)

(*b*) Ignoring the hazard, show how the pipeline would have to be modified to accommodate such instructions. Show both data and control (IR path) modifications. (8 pts)



(*c*) Describe two ways the structural hazard could be dealt with without delaying execution. (8 pts)

(*d*) Is this skipping of the MEM stage for arithmetic instructions worthwhile? *(Hint: It depends on whether register forwarding is used.)* Explain, stating any reasonable assumptions about pipeline features. (7 pts)

Problem 3: Answer each question below and on the next page.

(*a*) An ISA defines 200 out of 256 possible opcodes. An implementation can raise an illegal instruction exception when any of the unused instructions are encountered, or it might allow the instruction to execute (with the result being undefined). The first option is more expensive than the second. Why is the first option still the better choice? (8 pts)

(*b*) Describe two differences between a trap instruction and a subroutine call (jump and link). (8 pts)

(*c*) Why is the pipelined DLX implementation able to start fetching registers before decoding the instruction? (8 pts)

It's not over yet, more problems on next page.

(*d*) What is the advantage of the geometric mean over the harmonic and arithmetic means when combining normalized execution rates? (8 pts)

(*e*) Consider two implementations of an ISA, one old and one new. Describe why each factor, ϕ, IC, and CPI, might change from the old to the new implementation. (8 pts)