CS/ECE 552 Introduction to Computer Architecture

Midterm Exam

Monday, October 20, 2003

7:15-9:15 p.m

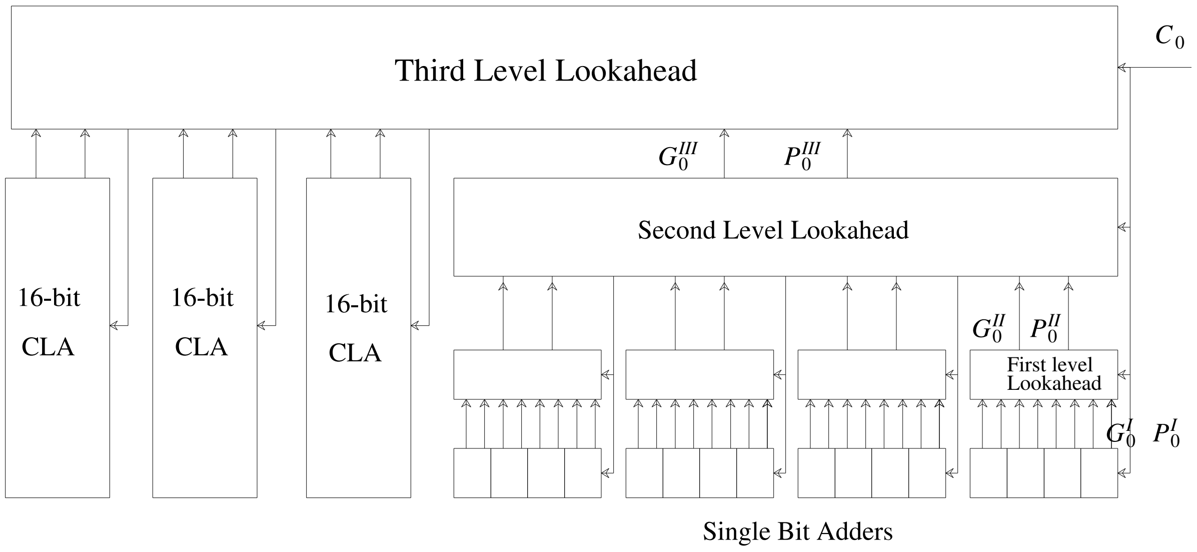
Name:____

Limit your answers to the space provided. Unnecessarily long answers will be penalized. If you use more space than is provided, you are probably doing something wrong. Use the back of each page for any scratch work.

Problem 1.	____________	(out of 20 points)
Problem 2.	____________	(out of 12 points)
Problem 3.	____________	(out of 14 points)
Problem 4.	____________	(out of 11 points)
Problem 5.	____________	(out of 19 points)
Problem 6.	____________	(out of 8 points)
Problem 7.	____________	(out of 16 points)
Total	____________	(out of 100 points)

(1) **Incrementer Design (20 points)**

The figure below shows the general structure of a 64-bit carry lookahead *adder* with three levels of lookahead. Some signals are shown in the figure to give you an idea of the structure of the adder; however not all relevant signals are shown (for example the carries from the first level lookahead to the single-bit adders). You are to use this structure (i.e., single-bit units with 3 levels of lookahead) to design a 64-bit *incrementer*. As discussed in class, and as you did in the homework, we can get an incrementer from an adder by making the Y input zero, and $C_0 = 1$.



(a) Let X_i, C_i and S_i, with $i = 0, 1, ..., 63$, represent the individual bits of the number to be incremented, the carries, and the sums, respectively. Let G_i^I and P_i^I, $i = 0,....,63$ represent the first level generates and propagates. Let G_j^{II} and P_j^{II}, $j = 0,...,15$ represent the second level generates and propagates, and let G_k^{III} and P_k^{III}, $k = 0,..,3$ represent the third level generates and propagates.

Write the logic equations for the following signals, in the space provided. (The equations should be in terms of the inputs to the logic block for which they represent the outputs.) **(14 points)**

$G_0^I =$
$P_3^I =$
$P_3^{III} =$
$C_{16} =$
$C_{11} =$
$C_{54} =$
$S_{39} =$

(b) Assume that each gate delay is τ time units, and all gates are available with up to 4 inputs. Further assume that all X_i and C_0 are ready at time 0. In the table below, show at what time the chosen signal value is ready. Use the comment column for any comments (comments will be used to determine partial credit in case of an incorrect answer). (**6 points**)

Signal	Time Ready	Comments
S_{19}		
G_1^{III}		
C_{48}		

(2) **TRUE and FALSE** questions. **(12 points)**

For each of the following questions, answer **TRUE(T)** if the statement is true, and **FALSE** if the statement is false. You will be *credited* 2 points for each *correct* answer and *penalized* 1 point for each *incorrect* answer. **It is possible that you may perceive a question to be ambiguous. If that happens, please state any assumptions that you make in your answer.** The minumum score for this question will be zero.

(i) A barrel shifter to shift a 16-bit input number right by up to 15 bit positions can be built with 4 stages of 2-1 multiplexors.

(ii) In a load/store architecture, the only instructions that access memory are load and store.

(iii) The sign extension unit in the datapath you have studied is sequential logic.

(iv) More powerful instructions lead to higher performance since the total number of instructions executed is smaller for a given task with more powerful instructions.

(v) Pipelining improves performance by decreasing the latency of each instruction.

(vi) Since an add *operation* has 3 operands (2 input and 1 output), add *instructions* must be 3-address instructions.

(3) **Short Questions (14 points)**

(i) What is the primary implementation advantage of fixed-length instructions. **(3 points)**

(ii) What are the 5 components of a modern computer system? (*Hint:* Two of these components can be combined and called the processor.) **(2 points)**

(iii) What is the formula for the total execution time of a program? **(2 points)**

(iv) In a system executing jobs, when is:

(a) throughput = 1/latency **(2 points)**

(b) throughput > 1/latency **(2 points)**

(v) What is a stored program computer? **(3 points)**

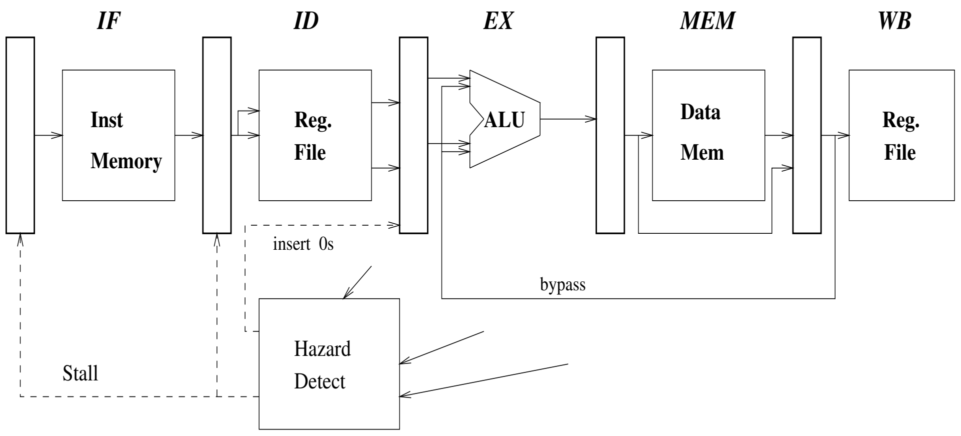
(4) **More Short Questions (11 points)**

(i) What are structural hazards in a pipeline? **(3 points)**

(ii) What are branch hazards in a pipeline? **(3 points)**

(iii) In the MIPS instruction set, the top 6 bits (31..26) are used for an opcode. With 6 bits, we can have 64 different combinations. Does this imply that the total number of instructions in the MIPS architecture is less than 64? Explain. No credit without an explanation. **(5 points)**

(5) **Pipelining (19 points)**
Following is a simple pipeline; with a bypass from the output of the MEM stage (i.e., the MEM/WB latch) to the EX stage. (*Note: There is no bypass from the EX stage, i.e., the EX/MEM latch, back to the EX stage.*) Assume that the register file is written in the first half and read in the second half of the clock cycle. Hence, a register value can be written and read in the same cycle.



(a) What purpose does the bypass serve? **(3 points)**

(b) For the following instruction sequence, show the pipeline timing in the table. Part of the timing for the first instruction is shown. **(12 points)**

R1 <-- R2 + R3	ADD1
R4 <-- R1 + R5	ADD2
R5 <-- Mem[R4]	LOAD1
R6 <-- Mem[R5]	LOAD2
R7 <-- R5 + R4	ADD3

Cycles	IF	ID	EX	MEM	WB
1	ADD1				
2	ADD2	ADD1			
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					

THIS QUESTION CONTINUES ON THE NEXT PAGE

(iii) Suggest technique(s) to eliminate the bubbles in part (ii). Which bubbles would your proposed technique(s) eliminate? **(4 points)**

(6) **Machine Performance (8 points)**

Consider two machines, machine A and machine B. Machine B runs floating-point instructions 6 times faster than machine A.

(i) Consider a program that takes 100 seconds to run on machine A, and spends half its time in floating-point instructions. How much faster is machine B (over machine A) at executing this program? **(3 points)**

(ii) Suppose we want a benchmark program to run 3 times faster on machine B than it runs on machine A. What fraction of its time does it spend executing floating point instructions on machine A? What fraction of its time does the same program spend executing floating-point instructions on machine B? **(5 points)**

(7) **Datapath (16 points)**

In this problem you will extend the multi-cycle datapath as presented in the course text in Figures 5.33 and 5.34. These figures have been provided on the last page of this exam.

It is possible to decrease the execution time of a program by combining two or more instructions (that would normally take two or more instruction times to execute) into a single instruction that takes only one instruction time (or a fraction more than one) to execute. These hybrid instructions are chosen because the instructions they replace appear often as a group in programs, and because they can be implemented without extensive modification to the datapath.

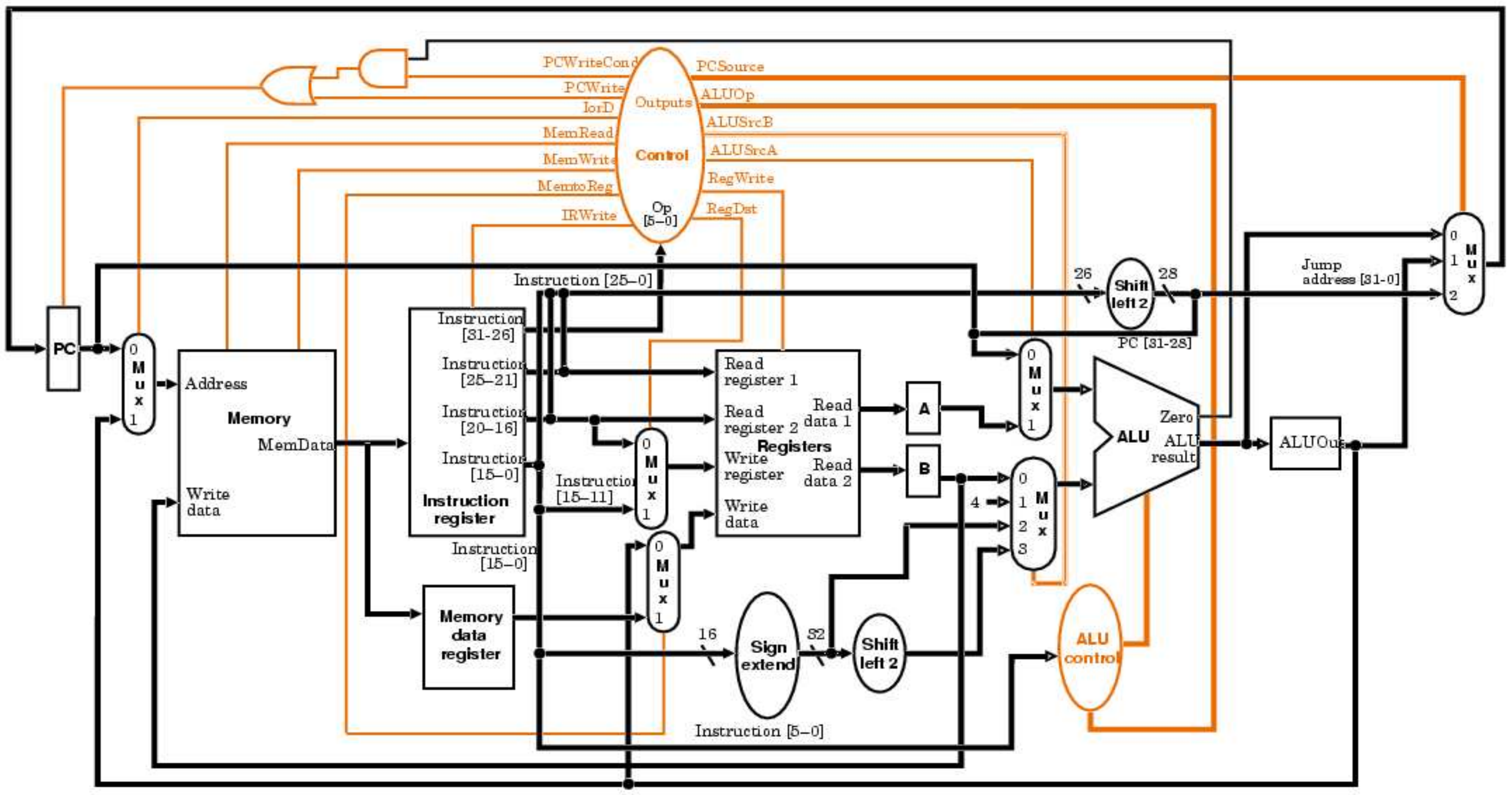
Modify the datapath as necessary to allow the implementation of each instruction. Be sure to include complete information about any new signals that appear in the modified datapath.

After the datapath is modified, fill in the list of control signals that must appears at each clock cycle in order to implement the instruction. To reduce the number of signals that you have to write, assume that the initial value of all signals is zero. Remember that if you assert a signal, you must de-assert it when necessary. The number of clock cycles needed for each instruction may vary. Use as many as you need, but make the implementation as efficient as possible. The signals for the first clock cycle already appear in the table. You should not have to modify them.

ALL OF YOUR WORK FOR EACH INSTRUCTION SHOULD APPEAR ONLY ON THE PAGE FOR THAT INSTRUCTION.

Note: In figure 5.34, ALU stands for arithmetic logic unit, MDR for memory data register, PC for program counter, and IR for instruction register.

(i) On the next page, modify the datapath(and control signals, as necessary), and fill in the list of control signal to implement a `decrement and branch if non-zero (DCRBNEQ)` instruction. This instruction has a 6 bit opcode (bits 26-31), a 5-bit Rs register field (bits 21-25), a 5-bit Rt register field (bits 16-20), and a 16-bit displacement field (bits 0-15). The semantics of the instruction are as follows: the Rs register is decremented by 1 and the new value is stored in the Rs register. If the new value is non-zero, the branch is taken, else it is not. The branch target address is calculated by adding the 16-bit displacement, sign extended and shifted left by 2 bits, to PC+4 (i.e, the PC of the next instruction). The Rt register field of the instruction is not used. (**Note: you may not need all the clock cycles provided in the table on the next page.**). **Points will be taken off for solutions that require more hardware, or take more time steps, than necessary. (16 points)**



Clock Cycle	Functional Description	Signals
1	Instruction -> IR and PC=PC+4	MemRd=1, ALUSrcA=0, IorD=0, IRWrite=1, ALUSrcB=01 ALUOp=00, PCWrite=1, PCSource=00
2		
3		
4		
5		
6		
7		
8		