(1) Give a one-word definition of coherence.

(1) Give a one-word definition of consistency.

(1) Using a different mapping scheme will reduce which type of cache miss?

(1) Which type of cache miss can be reduced by using longer lines?

(1) Which type of cache miss can be reduced by using shorter lines?

(1) When performing branch prediction, what two things are necessary? (A prediction, and what?)

(1) What is a "balanced" pipeline?

(1) What type of multiple issue machine requires static scheduling exclusively?

(1) What type of multiple issue machine provides extensive hardware support for dynamic scheduling?

(1) As minimum feature sizes decrease, what happens to wire resistance?

(2) What is the relationship between speculation and power consumption?

(2) What are the two biggest challenges to obtaining a substantial decrease in response time when using a MIMD parallel processor?

(2) What pair of instructions are used to implement a lock in RISC systems (as described in the text)?

(2) There are two ways to define performance - what are they?

(2) Lowering the associativity is one technique for reducing the Hit Time. List 2 others.

(2) Prefetching is one technique for reducing the Miss Rate. List 2 others. (You cannot reuse any from the previous question.)

(2) Giving reads priority over writes is one technique for reducing the cache miss penalty. List 2 others. (You cannot reuse any from the previous two questions.)

(2) Clock rates have grown by a factor of 1000 while power consumed has only grown by a factor of 30. How was this accomplished?

(2) Pipelining improves instruction _____ rather than instruction _____ (Fill in the blanks)

(2) What does CAM stand for? Where is it used in a processor?

(2) Why do most pipelined machines avoid the use of condition codes?

(2) What does UMA stand for? Which is more sensitive to where data is placed, an UMA machine or a NUMA machine?

(2) Writes to a cache are inherently slower than reads from a cache - why?

(2) Which is more expensive to build - a shared memory machine, or a message passing machine? Why?

(2) Which is easier to write a program for, a shared memory machine or a message passing machine? Why?

(2) What is the definition of a basic block? Why is there a desire to create a bigger one?

(3) The memory system presents challenges to ILP designers as well. What is it about the memory system that makes it hard for compilers to optimize (schedule) code, and also for execution units to achieve maximal performance? (This is not a question about technology, it's a higher-level question).

(3) What is Register Renaming, and why is it important/useful? (What problem is it trying to solve?)

(2) What is a benchmark program? What is the perfect/ideal benchmark program?

(2) Do benchmark programs remain valid indefinitely? Why or why not?

(2) Why is it difficult to come up with good benchmarks for parallel processors? (Think about what a benchmark is, and what it is supposed to measure)

(3) Supporting precise interrupts in machines that allow out of order completion is a challenge. Explain why.

(3) What is the difference between static and dynamic scheduling? And why do we want to schedule code, anyway?

(2) What is the primary difference between superscalar and VLIW processors?

(3) Give two advantages to using a superscalar processor, and 1 advantage to using a VLIW.

(2) Some instruction sets are easier to pipeline than others. Give one thing an instruction set should have in order to make it easier to pipeline.

(2) Some instruction sets are harder to write virtual machine monitors for than others. Why is that? What makes one instruction set harder to write a virtual machine for than another one?

(3) The designer has the choice of using a physically addressed cache or a virtually addressed cache. Explain the difference, and give 1 advantage for each.

(4) The standard MIPS has a 5-stage pipeline, and uses a branch delay slot and a load delay slot. If the machine is redesigned to be an 8-stage pipeline, with the following stages:

F D RR E1 E2 MEM WB (where RR stands for Register Read)

a. How many branch delay slots will this new design require, assuming the branch condition is calculated and available at the end of E1? E2?

b. How many load delay slots would this machine need (assuming it has forwarding logic) assuming the memory returns the value during M1? M2?

(2) Is it possible to have a WAW hazard in the above 8-stage pipeline? Why or why not?

(3) You are responsible for designing a new processor, and for a variety of reasons you must use a fixed 12 bit instruction size. You would like to support 16 different operations, use a 3-operand instruction format, and have 8 registers. If it is possible to do this, draw what an instruction would look like. If it is not possible, explain why, and show what you would do to fix

(4) Understanding the hardware can influence how you write programs. Give 2 examples of how you might write software differently for a heavily pipelined machine verses a non-pipelined one.

(3) Assuming a 21-bit address and a 256-byte Direct Mapped cache with a linesize=8, show how an address is partitioned/interpreted by the cache.

(3) Assuming a 21-bit address and a 96-byte 3-way SA cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

(2) Assuming a 21-bit address and a 110-byte FA cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

(2) Given a 1 Megabyte physical memory, a 26 bit Virtual address, and a page size of 4K bytes, write down the number of entries in the Page Table, and the width of each entry.

(3) Given a 1 Megabyte physical memory, a 32 bit Virtual address, and a page size of 4K bytes, write down the number of entries in the Page Table, and the width of each entry. Is there a problem with this configuration? If so, how can you fix it?

(10 pts) Here is a code sequence. (have them identify the hazards, both true and name.)

lw R1, 0(R10)

lw R2, 4(R10)

add R3, R2, R1

sw R3, 20(R0)

lw R2, 4(R10)

add R4, R1, R4

sw R4, 24(R0)

a) Assuming a standard 5-stage pipeline that does not support hazard detection and does no forwarding, insert as many NOPS as required in order to ensure this code runs correctly. (Remember, writes to the register file occur on the first half of the cycle, and reads occur during the second half).

b) Circle the NOPs that can be removed if forwarding and hazard detection logic is implemented.

c) Reorder the code to remove as many stalls as possible (assuming there is forwarding logic). How many cycles does it take now?

(2) Processor A requires 300 instructions to execute a given program, uses 2 cycles per instruction, and has a cycle time of 6 ns. Processor B requires 3 cycle per instruction, and also requires 300 instructions to do the same program. What must the cycle time of Processor B be in order to give the same CPU time as Processor A?

In this question, we are going to wire up a 12-bit processor. The machine is word-addressable, where a word is 12 bits. Immediates are sign extended, Offset is not. The machine has 3 different instruction formats: R, I, and J. Memory takes a single cycle to return a value.

R-type:	*(Arithmetic and logical:*				*rd = rs1 OP rs2)*
Opcode	rd	rs1	rs2	funct	
11-8	7-6	5-4	3-2	1-0	

I-type:	*(Arithmetic and logical:*			*rd = rs1 OP Immediate)*
	(Load:			*rd = mem[rs1+Immediate])*
	(Store:			*mem[rs1+Immediate]=rd)*
Opcode	rd	rs1	Immediate	
11-8	7-6	5-4	3-0	

J-type:	*(*	*PC = Offset)*
Opcode	Offset	
11-8	7-0	

The ALU can perform 4 functions, written this way: OP [ALU0 ALU1]
Add [11], AND [01], OR [10], NOT [00]

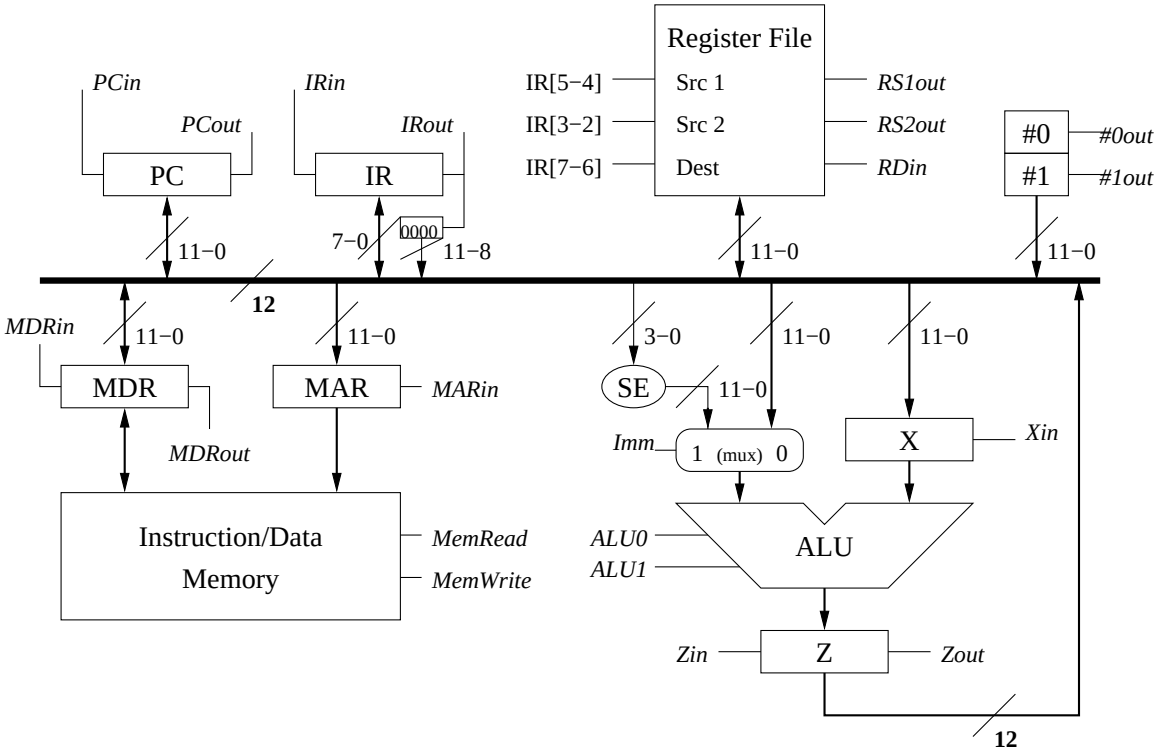
Here are some of the instructions that have been defined:

Name	**Opcode(Funct)**	**Name**	**Opcode(Funct)**	**Name**	**Opcode(Funct)**
NOP	0000(00)	lw	0001(xx)	sw	0011(xx)
NOT	1000(00)	beqz	0100(xx)	j	0101(xx)
AND	1000(01)	AND Imm	1001(xx)	ADD	1100(01)
OR	1000(10)	OR Imm	1010(xx)	ADD Imm	1101(xx)
XOR	1000(11)	XOR Imm	1011(xx)	SUB	1100(01)

Here are the 21 control signals.

PCin	PCout	IRin	IRout	MDRin	MDRout	MARin
Zin	Zout	RDin	RDout	MemRead	MmWrite	Imm
RS1out	RS2out	#0out	#1out	ALU0	ALU1	Xin

Here is a diagram of the machine.



(8) Fill in the microcode steps necessary to perform an instruction fetch (incrementing the PC is considered part of fetch).

Step	PCin	IRin	MDRin	MARin	Zin	RDin	Xin	PCout	IRout	MDRout	Zout	RDout	RS1out	RS2out	#0out	#1out	ALU0	ALU1	Mread	Mwrit	Imm
0																					
1																					
2																					
3																					
4																					
5																					

(8) Now that you have done the instruction fetch, fill in the microcode steps necessary to perform the following instruction: LW R3, 9(R1)

Step	PCin	IRin	MDRin	MARin	Zin	RDin	Xin	PCout	IRout	MDRout	Zout	RDout	RS1out	RS2out	#0out	#1out	ALU0	ALU1	Mread	Mwrit	Imm
0																					
1																					
2																					
3																					
4																					
5																					

(4) Suppose we want to use microcode to provide the control signals for this machine. Assuming the longest instruction takes a total of 10 cycles and the simplest microcode configuration is used,

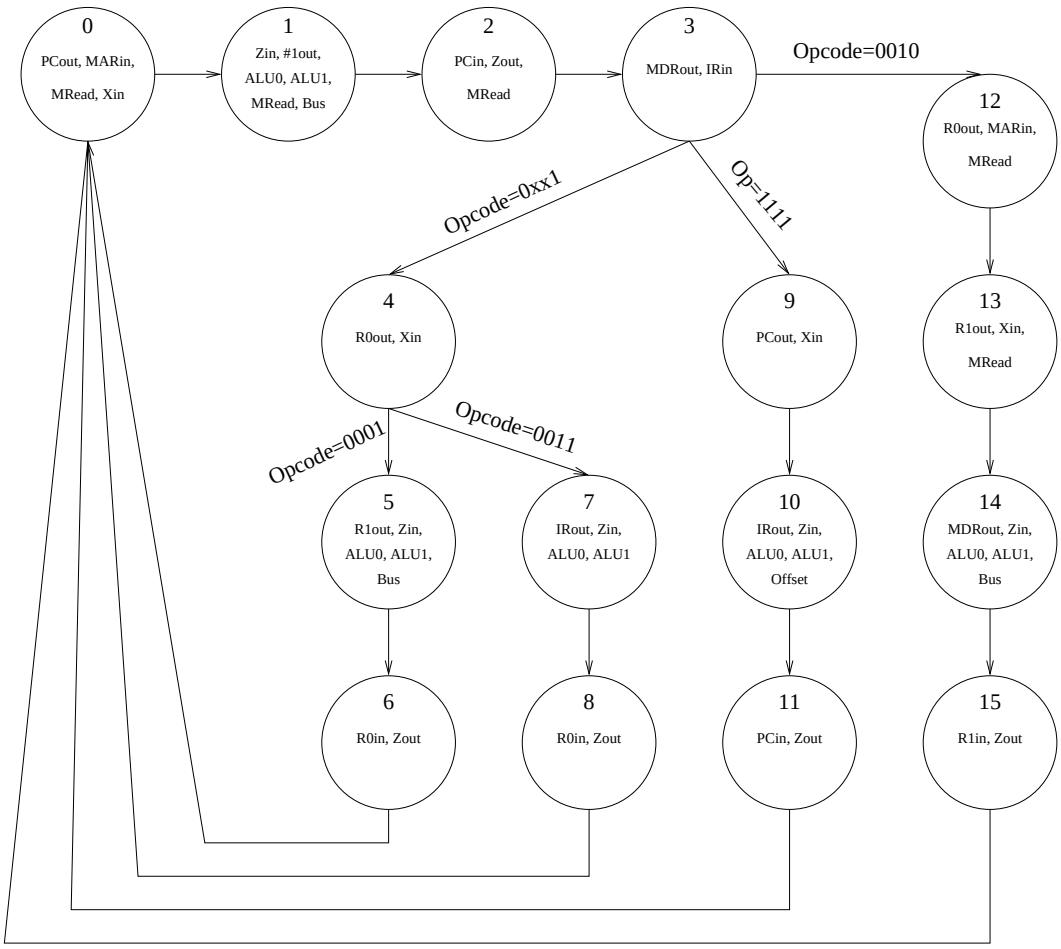
a. (1) How many entries would the microcode memory have?

b. (1) How wide would each entry be?

c. (2) What would be contained in each memory location?

(3) An important program spends 75% of its time doing Floating Point operations, and 25% of its time doing integer arithmetic. By redesigning the hardware you can either make the Floating Point unit 1/3 faster (take 2/3 as long), or the integer unit 99% faster (take 1% as long). Which should you do, and why?

Here is the state diagram for a random machine:



(4) Assuming there are 4 state variables (Y3-Y0), that State0=!Y3*!Y2*!Y1*!Y0 (000) and State15=Y3*Y2*Y1*Y0 (1111), write down the exact boolean equation for the MDRout signal.

(2) Assuming the same situation as in the previous question, write down the exact boolean equation for NextState12.

(4) If you were using a minimized microcode configuration for this machine, circle on the diagram where the dispatch roms points are.

(7) In class, we talked about the cycle by cycle steps that occur on different interrupts. For example, here is what happens if there is an illegal operand interrupt generated by instruction i (note this machine has a 7 stage pipeline):

	1	2	3	4	5	6	7	8	9	10	11	12
i	IF	ID	RR	EX	M1	M2	WB	<- Interrupt detected				
i+1		IF	ID	RR	EX	M1	M2	WB	<- Instruction Squashed			
i+2			IF	ID	RR	EX	M1	M2	WB	<- Trap Handler fetched		
i+3				IF	ID	RR	EX	M1	M2	WB		
i+4					IF	ID	RR	EX	M1	M2	WB	

Fill out the following table if instruction i+1 experiences a fault in the EX stage (Overflow, for example):

	1	2	3	4	5	6	7	8	9	10		
i	IF	ID	RR	EX	M1	M2	WB					
i+1		IF	ID	RR	EX	M1	M2	WB				
i+2			IF	ID	RR	EX	M1	M2	WB			
i+3				IF	ID	RR	EX	M1	M2	WB		
i+4					IF	ID	RR	EX	M1	M2	WB	
i+5						IF	ID	RR	EX	M1	M2	WB

What happens in this case?

	1	2	3	4	5	6	7	8	9	10		
i	IF	ID	RR	EX	M1	M2	WB	<- M1 stage has page fault				
i+1		IF	ID	RR	EX	M1	M2	WB	<- Inst Decode has Illegal Instruction			
i+2			IF	ID	RR	EX	M1	M2	WB			
i+3				IF	ID	RR	EX	M1	M2	WB		
i+4					IF	ID	RR	EX	M1	M2	WB	
i+5						IF	ID	RR	EX	M1	M2	WB

What is the maximum number of exceptions that could happen at one time in the above machine? Why?