Very short answer questions. "True" and "False" are considered very short answers.

(1) Some instruction sets are easier to pipeline than others.

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

(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) Give an example of an instruction set that is easily virtualizable.

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

(2) What is the main difference between a commodity cluster and a custom cluster?

(2) What is the equation to calculate the average memory access time?

Short answer questions:

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

(2) Caches and Virtual Memory both work because of what two principles?

(2) What is Weak Scaling?

(2) What is a "balanced" pipeline?

(2) What is CUDA, and what is it used for?

(2) How is the Network Bandwidth calculated?

(2) Why are there multiple dies per silicon wafer? Why not just fabricate one huge die per wafer?

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

(2) Predicting if a branch is taken or not taken is only half of the problem. What else is necessary in order to overcome control flow hazards?

(2) What is the goal of the memory heirarchy?

(2) There is one particular type of branch that is particularly difficult for predictors to deal with. What is it?

(2) What is the piece of hardware which makes correlating predictors different from other predictors?

(2) Give two techniques which will help reduce the Hit Time.

(2) Give two techniques will help reduce the Miss Rate. You cannot reuse any from the previous question.

(2) Give two techniques will help reduce the Miss Penalty. 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) What is the difference between a virtually addressed cache and a physically addressed cache?

(2) Give 1 disadvantage to using a virtually addressed cache, and 1 disadvantage to using a physically addressed cache.

(3) Explain what a precise interrupt is.

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

(3) 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) Briefly describe how a "Vector" processor works.

(3) What is Cache Coherence, and why is it necessary?

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

(2) What is the difference between static and dynamic scheduling?

(2) Why do we want to schedule code, anyway?

(3) 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.

(3) Assuming an 18-bit address and a 256-byte Direct Mapped cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

(3) Assuming an 18-bit address and a 96-byte 3-way SA cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

(3) Given a 1 Megabyte physical memory, a 25 bit Virtual address, and a page size of 2K bytes, write down the number of entries in the Page Table, and the width of each entry.

(4) Given a 1 Megabyte physical memory, a 32 bit Virtual address, and a page size of 2K 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?

(4) An important program spends 60% of its time doing Floating Point operations, and 40% of its time doing integer arithmetic. By redesigning the hardware you can either make the Floating Point unit 50% faster (take 50% as long), or the integer unit 70% faster (take 30% as long). Which should you do, and why?

(4) The standard 5-stage MIPS design uses a load delay slot. If the machine is redesigned to be a 7-stage pipeline, with the following stages:

F D RR E M1 M2 WB (where RR stands for Register Read)

How many load delay slots will this new design require, assuming the memory returns the value during M2? During M1?

(1) Is it possible to have a WAR hazard in the above 7-stage pipeline?

(4) You have an instruction that is a fixed size of 16 bits, and you want to support 32 different operations. You also want to have 16 registers, and use a 3-operand instruction format. If it is possible to this, draw what an instruction would look like. If it is not possible, explain why, and show what you would do to fix the problem.

(4) Snooping is one approach to providing cache coherence - state what the other main approach is, and briefly outline how each of them work.

(4) What are the three types of hazards? Which one can be eliminated by "throwing more money at the problem"?

(4) What does SMT stand for? Briefly describe how it works.

(6) Describe the difference between shared memory and message passing machines. Include the impact on design, cost, and programming model.

(6) 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.

(12 pts) Here is a code sequence.

lw R2, 8(R10)

sub R4, R3, R2

add R1, R4, R2

sw R2, 20(R0)

and R3, R1, R6

or R7, R6, R5

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

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.

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 [00], AND [01], OR [10], NOT [11]

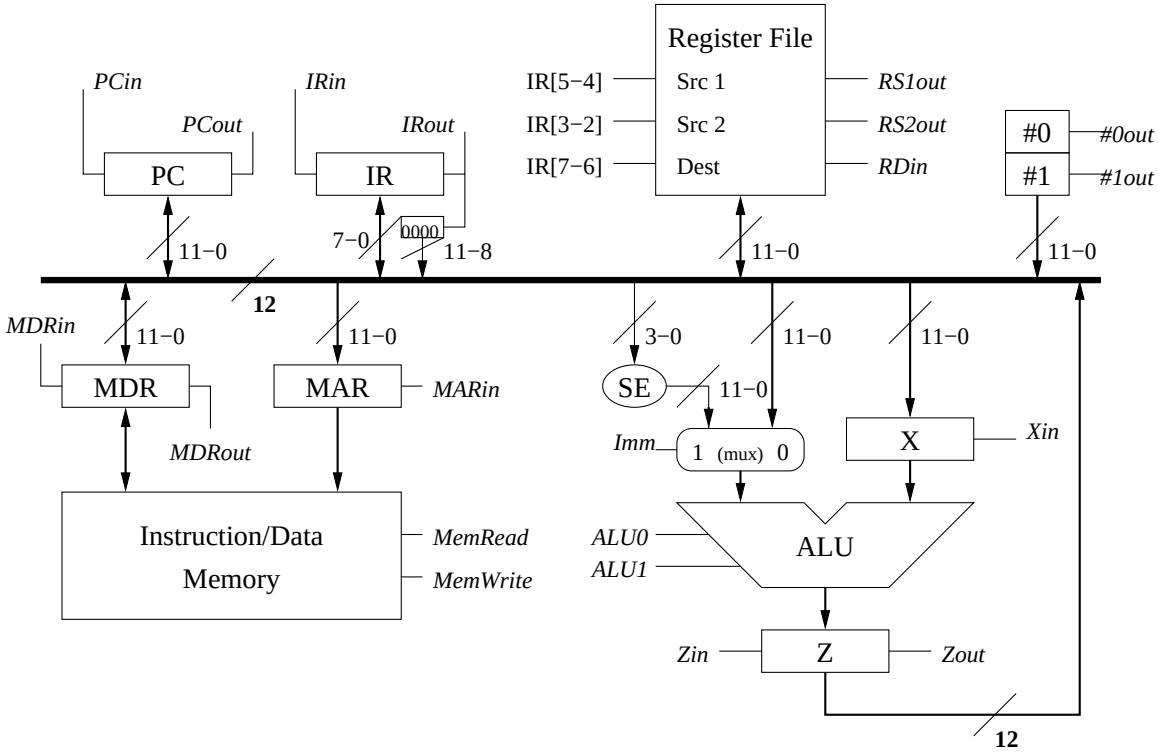
Here are some of the instructions that have been defined:

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.



(1) (6) Fill in the microcode steps necessary to perform an instruction fetch.

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

(4) 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 6 stage pipeline):

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

What happens in this case?

	1	2	3	4	5	6	7	8	9	10	
i	IF	ID	RR	EX	MEM	WB	<- Execution stage has overlflow				
i+1		IF	ID	RR	EX	MEM	WB	<- Inst Read causes Page Fault			
i+2			IF	ID	RR	EX	MEM	WB			
i+3				IF	ID	RR	EX	MEM	WB		
i+4					IF	ID	RR	EX	MEM	WB	
i+5						IF	ID	RR	EX	MEM	WB

---

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