Very short answer questions. You must use 10 or fewer words. "True" and "False" are considered very short answers.

[1] Which is on average more effective, dynamic or static branch prediction?

[1] Predicting the direction of a branch is not enough. What else is necessary?

[2] When we talk about the number of operands in an instruction (a 1-operand or a 2-operand instruction, for example), what do we mean?

[2] If we make transistors and wires smaller and smaller but make no other changes, what happens to the power density?

[2] There are two main ways to define performance - what are they?

[2] What are the two instructions used in RISC machines to support atomicity?

[2] What is the main difference between a commodity cluster and a custom cluster?

[2] Area on the die used to be the most critical design constraint. That is no longer true - what is the most critical factor now? Why?

[2] As speculation _____ power consumption _____

[2] Coherence refers to _____ is returned while consistency has to do with _____.

Short Answers (25 or fewer words)

[2] Writing to a cache is inherently slower than reading from a cache. Why?

[3] The power consumed by a chip has increased over time, but the clock rate has increased at a far greater relative rate. How was this possible? Explain how designers keep the chip from melting.

[3] Why is it so difficult for the processing elements on a CMOS-based chip to communicate with things that located off the chip?

[2] What is a benchmark program?

[2] Do benchmark programs remain valid indefinitely? Why or why not?

[2] Why is it difficult to come up with good benchmarks that will work across all parallel processors?

[5] In your first design of a 5-stage pipeline (F,D,E,M,W) F takes 22 time units, D takes 26, E takes 50, M takes 30 and W takes 19.

a) What will the clock cycle time be for this pipeline?

b) Is it a balanced pipeline? If not, explain what you could do to fix it. What would the cycle time be now?

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

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

[3] Supporting precise interrupts in a machine that allows out of order completion is a challenge. Why do we care? What is a precise interrupt, and why is it important to support precise interrupts?

[4] What is the primary difference between superscalar and VLIW processors? Give two advantages to using a superscalar processor, and 1 advantage to using a VLIW.

[15] For each of the following techniques, explain briefly how it works and indicate how (on average) the 3 terms of the AMAT equation are affected. The first one is done for you as an example.

Technique	Hit Time	Miss Rate	Miss Penalty
Using Critical Word First	**Unaffected**	**Unaffected**	**Decreased**

In this technique the word that is requested in the line is brought back first, allowing the processor to consume the data and proceed while in parallel the rest of the line is brought in and put into the cache. Thus, the Miss Penalty is lowered.

Technique	Hit Time	Miss Rate	Miss Penalty
Increasing Associativity			
Decreasing Cache Size			
Non-blocking Cache			
Hardware/Software Prefetching			
Victim Cache			

[7] The MIPS implementation we used in class has a 5-stage pipeline, writes to the register file during the first half of the cycle and reads during the second half, and uses both a branch delay slot and a load delay slot. If the machine is redesigned to be a 8-stage pipeline, with the following stages:

F D RR E1 E2 M1 M2 WB

a) Assuming this machine has a branch predictor and the branch condition is calculated by the end of the E1 stage, how big is the branch penalty (measured in cycles) when the prediction is incorrect? What if the branch condition is not calculated until the end of E2?

b) How many load delay slots would this machine need (assuming it has forwarding logic and you are forwarding to E1) assuming the memory returns the value by the end of M1? M2?

c) What type of data hazard does the above pipeline need to worry about?

d) If the above pipeline were modified to support out of order completion, what new data hazard would be introduced?

e) If in addition to completing out of order, instructions were allowed to issue out of order, what new data hazard would be introduced?

[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.

[3] Assuming a 22-bit address and a 1k-byte Direct Mapped cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

[3] Assuming a 22-bit address and a 160-byte 5-way SA cache with a linesize=8, show how an address is partitioned/interpreted by the cache.

[2] Assuming a 22-bit address and a 196-byte FA cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

[4] Given a 1 Gigabyte physical memory, a 48 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?

[2] The CPI of the WhizBang 8000 is 2, assuming a perfect memory system (all references to memory take a single cycle.) Assume we are able to keep a perfect instruction memory system but must use a more realistic data memory system - a data cache with a miss rate of 10% and a miss penalty of 40 cycles. If 35% of all instructions are loads or stores, what does the CPI become in this case?

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)*
	(Branch:			*if (rd OP rs1)=COND, PC=PC+Immediate)*
Opcode	rd	rs1	Immediate	
11-8	7-6	5-4	3-0	

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

The ALU can perform 4 functions, written this way: OP [ALU1 ALU0]
Add [11], SUB [10], AND [01], OR [00]

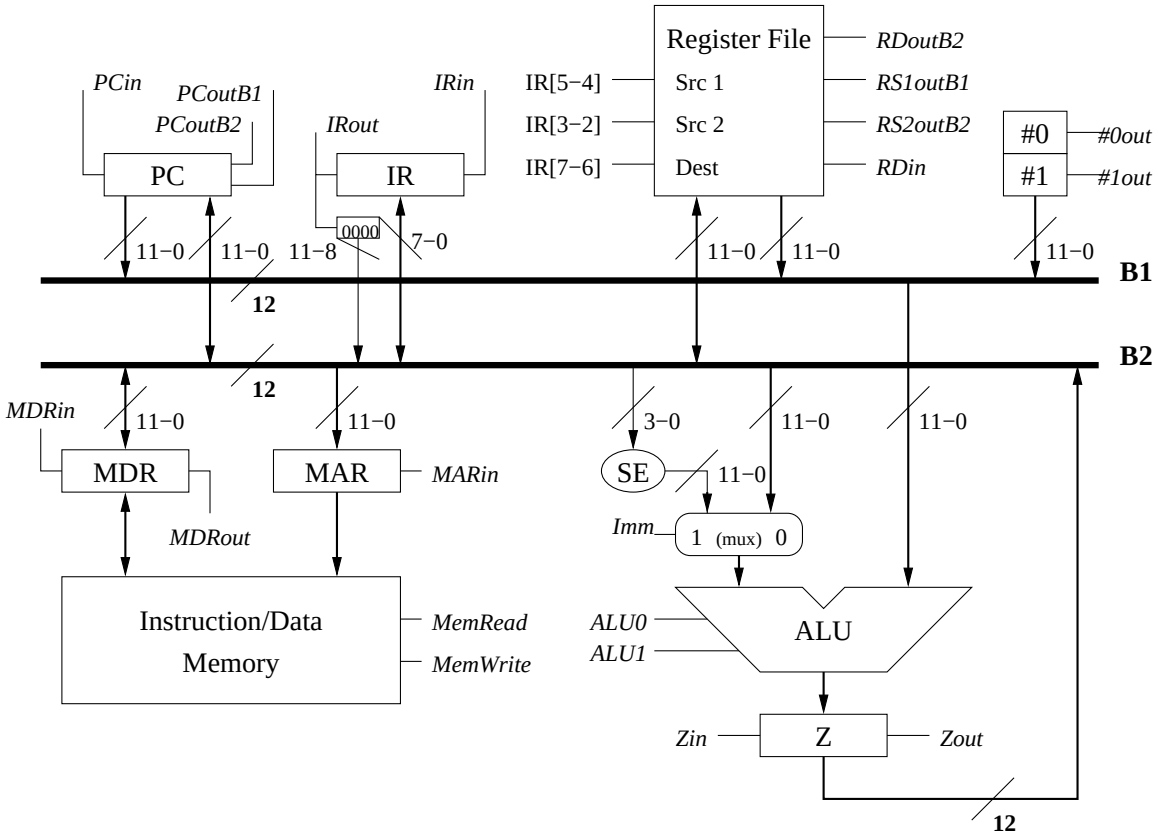
Here are some of the instructions that have been defined:

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 control signals.

PCin	IRin	MDRin	MARin	Zin	RDin	Imm
PCoutB1	PCoutB2	IRout	MDRout	Zout	RDoutB2	RS1outB1
RS2outB2	MemRead	MemWrite	#0out	#1out	ALU0	ALU1

Here is a diagram of the machine.



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

Step	PCin	IRin	MARin	MDRin	RDin	Zin	IRout	PCoutB1	PCoutB2	MDRout	Zout	RDoutB2	RS1outB1	RS2outB2	#0out	#1out	ALU1	ALU0	Mread	MemWrite	Imm
0																					
1																					
2																					
3																					
4																					
5																					

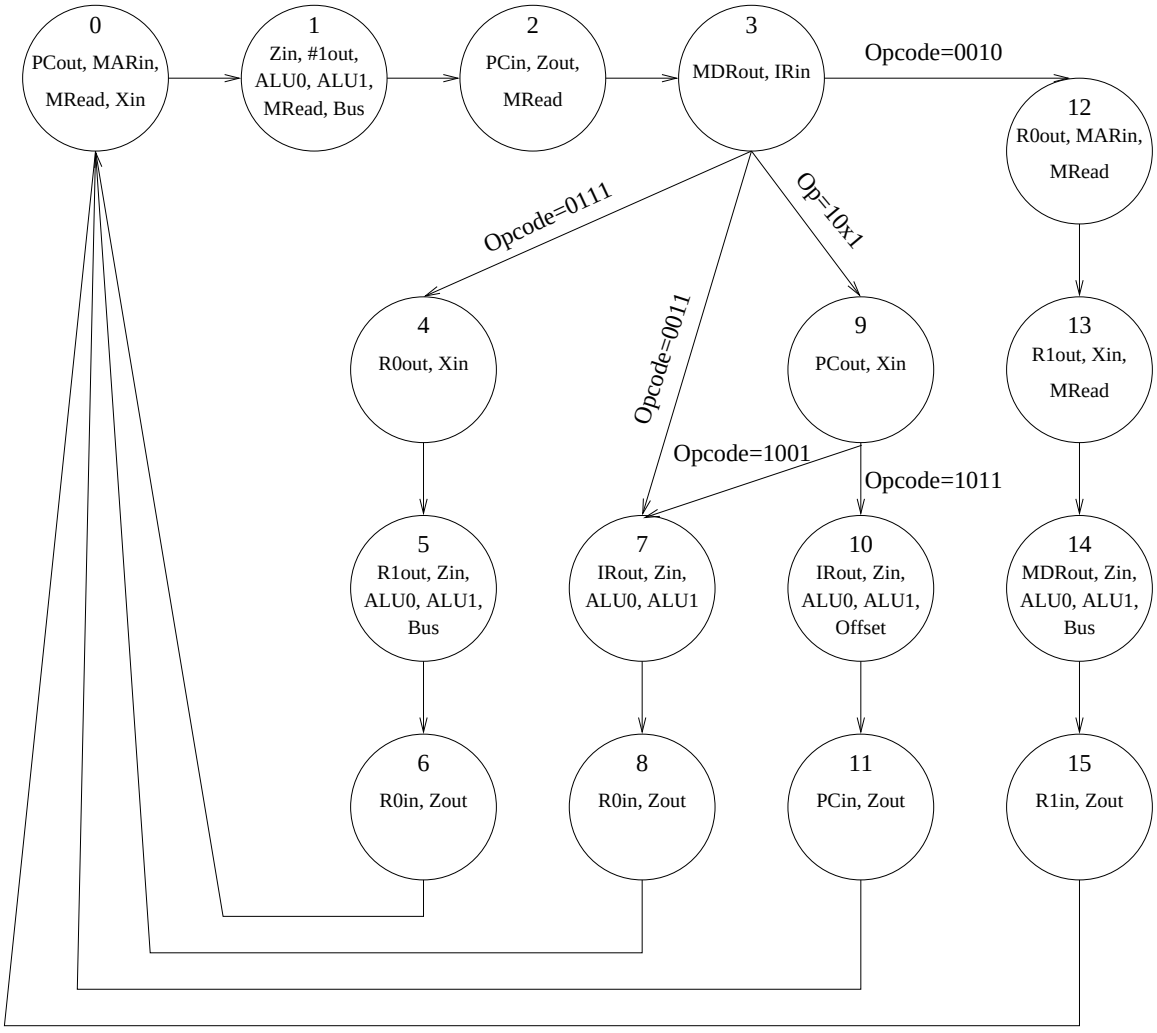
[2] Write down the binary bit pattern (the bit pattern that will be in the IR after you have done the instruction fetch) for the instruction: **LW R3, -2[R1]**

[6] Now that you have done the instruction fetch, fill in the microcode steps necessary to perform the following instruction: **LW R3, -2[R1]**

Step	PCin	IRin	MARin	MDRin	RDin	Zin	IRout	PCoutB1	PCoutB2	MDRout	Zout	RDoutB2	RS1outB1	RS2outB2	#0out	#1out	ALU1	ALU0	Mread	MemWrite	Imm
0																					
1																					
2																					
3																					
4																					
5																					

[4] You have been writing C programs for a simple, non-pipelined machine. You have recently received a promotion, and now your job is to write C programs for a heavily pipelined, high performance processor. These new programs must execute as fast as possible (the emphasis is on response time, not throughput). Give at least 2 examples of things you should do differently now, and be sure to explain in detail why (what is the problem you are overcoming?) (Note - make sure you are describing things that you can do in a high level language, not things that are done at the assembly language level.)

Here is the state diagram for a random machine:



[3] Assuming there are 4 state variables (Y3-Y0), that State0 = $\overline{Y3}$*$\overline{Y2}$*$\overline{Y1}$*$\overline{Y0}$ (0000) and State15 = Y3*Y2*Y1*Y0 (1111), write down the exact boolean equation for the **R1out** signal.

[4] Assuming the same situation as in the previous question, write down the exact boolean equation for **NextState7**.

[2] If you were using a minimized microcode configuration for this machine, circle on the diagram where the dispatch roms points are.

[5] Here is a code sequence.

```
load    R1, 0(R10)

add     R9, R2, R1

store   R3, 20(R9)

load    R6, 10(R8)
```

Assuming a standard 5-stage pipeline that supports hazard detection and does forwarding,

a) Indicate all dependencies (draw lines/arrows between them, and write beside each line/arrow which hazard is involved).

b) Insert as many No Operations (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).

c) Schedule the code to remove as many stalls as possible. How many NOPS are left?

d) There are things the static scheduler does not know which makes guaranteeing correctness difficult. What if, at execution time, R8=40 and R9=30? Will your dependency graph and ability to schedule this code change?

[2] An important program spends 70% of its time doing Integer operations, and 30% of its time doing floating point arithmetic. By redesigning the hardware you can make the Integer unit 40% faster (take 60% as long), or you can make the Floating Point unit 90% faster (take 10% as long). Which should you do and why? (You must show your work to get full credit.)

[8] Here is a code sequence.

```
load    R1, 0(R10)

add     R3, R2, R1

store   R3, 20(R9)

load    R3, 4(R12)

sub     R9, R7, R8

load    R9, 4(R12)

add     R6, R2, R7
```

Assuming a standard 5-stage pipeline that does not support hazard detection and does no forwarding,

a) Indicate all dependencies (draw lines/arrows between them, and write beside each line/arrow which hazard is involved).

b) Insert as many No Operations (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).

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

d) Assuming the pipeline has been modified so that it does support hazard detection and forwarding, schedule the code to remove as many stalls as possible. Assume there are no hazards through the memory system. How many NOPS are left?

[6] Here is a code sequence.

```
        sub     R7, R8, R10

        add     R7, R1, R7

Label:  lw      R3, 0(R10)

        add     R4, R2, R3

        sw      R4, 20(R2)

        breq    R3,R6,Label         ; branch to Label if R3=R6
```

Assuming a 6-stage pipeline (F,D,E,M1,M2,WB) that does not support hazard detection, does no forwarding, and does not use a branch delay slot,

a) Indicate all dependencies (draw lines/arrows between them, and write beside each line/arrow which hazard is involved).

b) Insert as many No Operations (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).

c) Circle the NOPs that can be removed if forwarding and hazard detection logic is implemented. Assume data on a read returns at the end of M2.

d) Assuming the pipeline has been modified so that it does support hazard detection and forwarding, schedule the code to remove as many stalls as possible. Assume there are no hazards through memory. How many NOPS are left?

[6] Suppose I have a 4-issue multithreaded machine, and there are 3 threads - A, B, and C. Assuming:

The number of independent instructions Thread A can find (in order): 2, then 3, then 0, then 1
The number of independent instructions Thread B can find (in order): 1, then 1, then 2, then 2
The number of independent instructions Thread C can find (in order): 1, then 0, then 2, then 1

Fill in the following table if coarse grained scheduling is being used.

Time	Slot1	Slot2	Slot3	Slot4
0				
1				
2				
3				

Now fill in the following table assuming the use of fine-grained scheduling.

Time	Slot1	Slot2	Slot3	Slot4
0				
1				
2				
3				

Now, repeat the process assuming simultaneous multithreading is being used.

Time	Slot1	Slot2	Slot3	Slot4
0				
1				
2				
3				

[3] You are responsible for designing a new embedded processor, and for a variety of reasons you must use a fixed 18 bit instruction size. You would like to support 64 different opcodes, use a 3-operand instruction format, and have 32 registers. If it is possible to do this, draw what an instruction would look like. If it is not possible, explain why, and give at least 2 different ways to solve the problem.

[6] 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 and supports imprecise interrupts. RR stands for Register Read):

	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 for the same machine, instruction i 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	
i+6							IF	ID	RR	EX	M1	M2	WB

Assuming precise interrupts are being supported, 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	
i+6							IF	ID	RR	EX	M1	M2	WB
i+7								IF	ID	RR	EX	M1	M2	WB

What is the maximum number of exceptions that could happen at a single time in the above machine? Explain how you got your answer.