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] Does an average program's locality behavior stay the same while it is executing, or does it vary?

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

[2] How do two processes communicate when running on a shared memory machine?

[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] Does reducing the minimum feature size affect power density? If so, in what way?

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

[2] What is the relationship between speculation and power consumption?

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

[2] What two things are keeping parallel processors from being the dominant architecture? (In other words, what are the two biggest challenges in parallel processing?)

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

Short Answers (25 or fewer words)

[3] Clock rates have grown by a factor of 1000 while power consumed has only grown by a factor of 30. How was this accomplished, and why did it work?

[3] What is "leakage" current? If Vdd is lowered, what happens to the amount of leakage current?

[3] 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?

[3] What is a benchmark program?

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

[3] Why is it difficult to come up with good benchmarks for parallel processors?

[9] What do the following acronyms stand for?

SMT	**MIMD**	**NUMA**
WAW	**DSM**	**ILP**
VMM	**VLIW**	**TLB**

[3] In your first design of a 5-stage pipeline (F,D,E,M,W) F takes 25 time units, D takes 22, E takes 50, M takes 35 and W takes 15. What will the clock rate be for this pipeline? Is it a balanced? If not, explain what you could do to fix it.

[3] 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?

[10] For each technique in the following table, indicate how (on average) the 3 terms of the AMAT equation are affected (assume there is a single L1 cache). The first one is done as an example. (Decreases = "-", Increases = "+", Unaffected = "x"

Technique	Hit Time	Miss Rate	Miss Penalty
Increasing Line/Block Size	**x**	**-**	**+**
Decreasing Cache Size			
Increasing Associativity			
Compiler Optimizations			
Hardware Prefetching			
Compiler Controlled Prefetching			
Victim Cache			
Virtually Addressed Cache			
Non-blocking Cache			
Using Critical Word First			

[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.)

[8] 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 7-stage pipeline, with the following stages:

F D 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] An important program spends 65% of its time doing Integer operations, and 35% of its time doing floating point arithmetic. By redesigning the hardware you can make the Floating Point unit 30% faster (take 70% as long), or you can make the Integer Unit two orders of magnitude faster (take 1% as long). Which should you do and why?

[3] Assuming a 20-bit address and a 512-byte Direct Mapped cache with a linesize=8, show how an address is partitioned/interpreted by the cache.

[3] Assuming a 20-bit address and a 160-byte 5-way SA cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

[2] Assuming a 20-bit address and a 146-byte FA cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

[2] Given a 2 Megabyte physical memory, a 32 bit Virtual address, and a page size of 1K 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?

[10] Here is a code sequence.

```
load    R1, 0(R10)

load    R2, 4(R1)

add     R3, R2, R1

store   R3, 20(R9)

sub     R3,R7,R8

load    R11, 4(R6)

load    R5, 8(R10)

add     R6, R5, R4
```

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. How many NOPS are left?

[10] Here is a code sequence.

```
        sw      R1, 4(R10)

Label:  lw      R2, 0(R10)

        add     R4, R2, R3

        sw      R4, 20(R2)

        breq    R3,R4,Label         ; branch to Label if R3=R4

        lw      R3, 4(R0)

        add     R5, R1, R7

        subi    R5, R7, 24
```

Assuming a 6-stage pipeline (F,D,E,M1,M2,WB) that does not support hazard detection, does no forwarding, and uses 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. How many NOPS are left?

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 7 functions, written this way: OP [ALU2 ALU1 ALU0]
Increment X [001], Decrement X [010], Add [011], SUB [100], AND [101], OR [110], NOT [111]

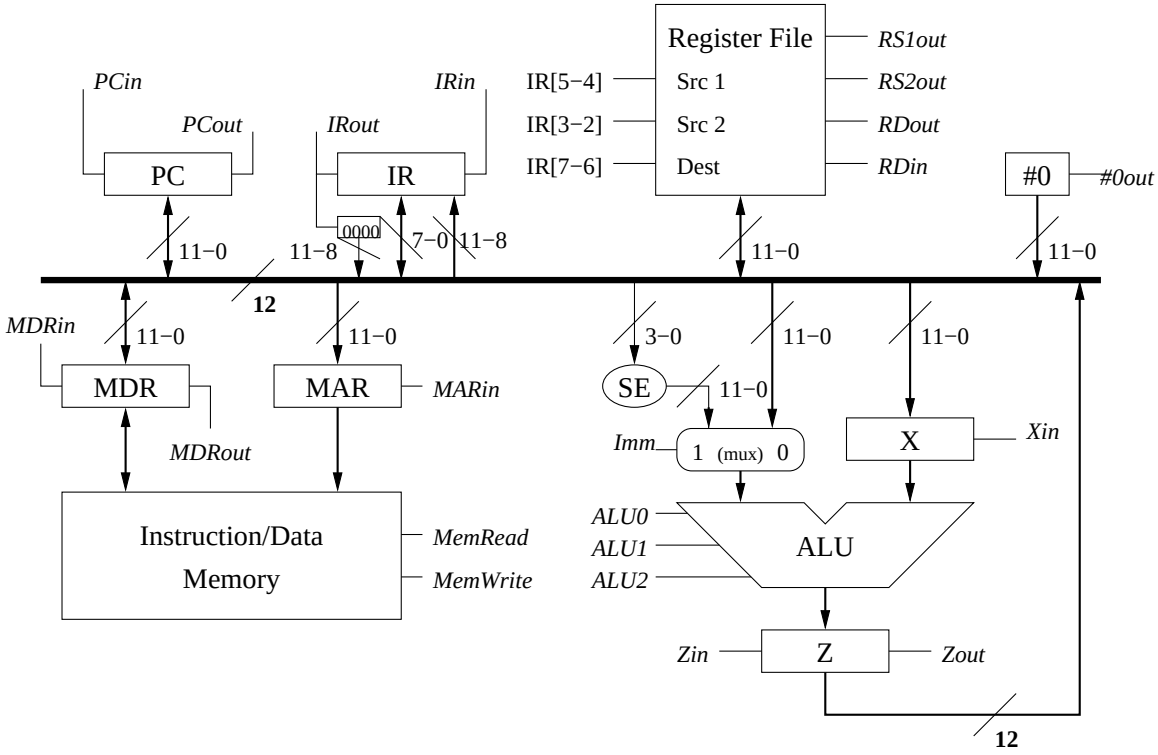
Here are some of the instructions that have been defined:

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	ALU0	ALU1	ALU2	Xin

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	Xin	Zin	IRout	PCout	MDRout	Zout	RDout	RS1out	RS2out	#0out	ALU2	ALU1	ALU0	Mread	Mwrite	Imm
0																					
1																					
2																					
3																					
4																					
5																					

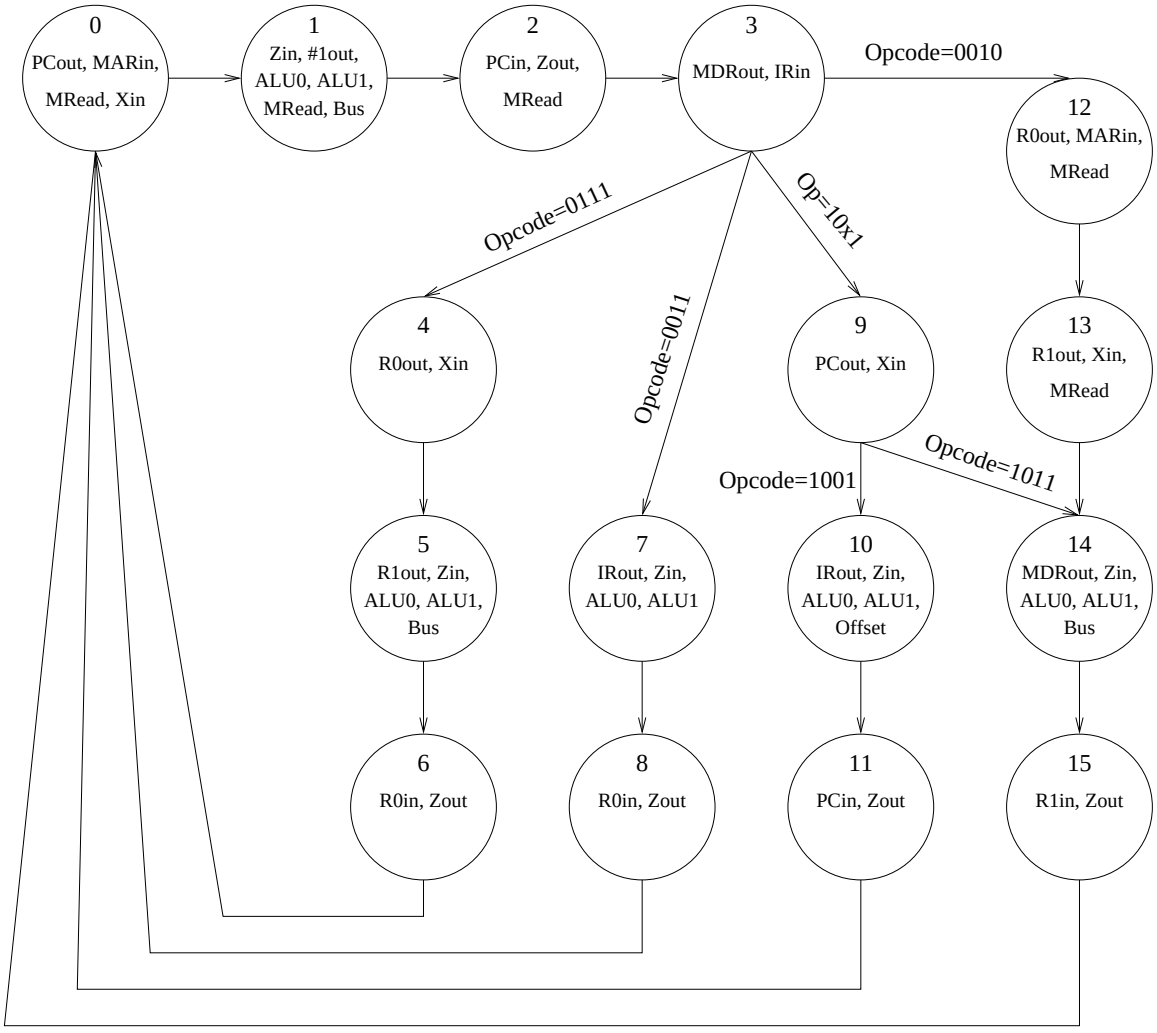
[2] Write down the binary bit pattern (the bit pattern that will be in the IR after you have done the instruction fetch) for the instruction: **LW R1, -3[R2]**

[6] Now that you have done the instruction fetch, fill in the microcode steps necessary to perform the following instruction: **LW R1, -3[R2]**

Step	PCin	IRin	MARin	MDRin	RDin	Xin	Zin	IRout	PCout	MDRout	Zout	RDout	RS1out	RS2out	#0out	ALU2	ALU1	ALU0	Mread	Mwrite	Imm
0																					
1																					
2																					
3																					
4																					

[3] You are responsible for designing a new embedded processor, and for a variety of reasons you must use a fixed 15 bit instruction size. You would like to support 32 different opcodes, use a 3-operand instruction format, and have 16 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.

Here is the state diagram for a random machine:



[4] 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 **Bus** signal.

[3] Assuming the same situation as in the previous question, write down the exact boolean equation for **NextState14**.

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

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				

[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+1 experiences a fault in the M1 stage (Page Fault, 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	<- EX 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 one time in the above machine? Why?