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

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

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

[1] Is peak performance usually the same as sustained performance?

[1] Which type of cache miss can be reduced by using longer lines?

[1] Using a different mapping scheme will reduce which type of cache miss?

[1] What pipeline hazard can be avoided using a technique known as value prediction?

[1] As transistors and wires shrink, what happens to the power density?

[1] What is the definition of a precise interrupt?

[1] Why is it important to support precise interrupts in modern pipelined processors?

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

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

[2] There are two major challenges to obtaining a substantial decrease in response time when using the MIMD approach. What are they?

[2] The Trap Handler needs to know what two things in order to deal with an interrupt?

[2] Give a one-word definition of coherence, and a one-word definition of consistency.

Short Answers (20 or fewer words)

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

[2] What is a benchmark program?

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

[2] What does ROB stand for, and why is it used in modern advanced pipelines? (What necessary function does it help support?)

[2] If I add processors but keep the job size the same, am I measuring strong or weak scaling? Does this correspond most closely to response time or throughput?

[3] Over the years, clock rates have grown by a factor of 1000 while power consumed has only grown by a factor of 30. How was this accomplished without melting the chip?

[3] Why is it difficult/impossible to create a benchmark that will work across all classes of parallel processors?

[3] Speculation is a very useful technique for improving performance. However, it is not being used as extensively as it once was - why not?

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

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

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

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

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

[4] Vector machines are an example of a SIMD style of parallel processing. They feature instructions that look like **VR0 = VR1 + VR2**. Explain briefly why these machines are able to fetch and decode many fewer instructions than a tradition processor does. Use pictures if that will help get your point across.

[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.

[3] The designer has the choice of using a physically addressed cache or a virtually addressed cache. Explain the difference (drawing a picture is fine!), and give 1 advantage for each.

[3] Processor A requires 250 instructions to execute a given program, uses 4 cycles per instruction, and has a cycle time of 6 ns. Processor B requires 2 cycles per instruction, and 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?

[3] You are responsible for designing a new embedded processor, and for a variety of reasons you **must** use a fixed 25 bit instruction size and you **must** support at least 64 different opcodes. You would like to use a 3-operand instruction format, and have 128 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.

[3] Find the Average Memory Access Time (AMAT) for a processor with a 1 ns clock cycle time, a miss penalty of 30 clock cycles, a miss rate of 0.10 misses per instruction, and a cache access time (including hit detection) of 1 clock cycle. Assume that the read and write miss penalties are the same and ignore other write stalls.

[15] For each of the following techniques, circle the arrow(s) associated with each of the terms in the Average Memory Access Time which indicates how (on average) that term is affected. If there is more than one answer, then circle more than one term. For example, if the HT goes up, you would circle the up arrow - if the HT goes down, circle the down arrow. If the HT is unaffected, do not circle anything. Assume there is a single L1 cache.

Increasing cache size (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)

Decreasing Associativity (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)

Increasing Line/Block size (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)

Hardware Prefetching (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)
(Assume prefetched data arrives before it is needed)

Hardware Prefetching (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)
(Assume prefetched data does not arrive before it is needed)

Compiler optimizations (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)
(Excluding prefetching)

Nonblocking cache (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)

Adding an L3 cache (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)
(Affect on L2 parameters)

Using Physically Addressed Cache (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)

Adding a Victim Cache (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)
(Assume it is accessed in parallel)

Adding a Victim Cache (HT: ↑ ↓ MR: ↑ ↓ MP: ↑ ↓)
(Assume it is not accessed in parallel)

[9] Understanding the hardware can influence how you write high-level programs. You have been writing C programs for a high performance, *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 with an advanced tournament-style branch predictor (but no RAS). Your programs must execute as fast as possible (the emphasis is on response time, not throughput), and your code will be compiled using a highly optimizing compiler on the -O3 setting.

a) Give an example of how you will change the way you write your C program. Explain in detail why you decided to make the change (what is the problem you are overcoming?)

b) Give another example of how you will change the way you write your C program. Explain in detail why you decided to make the change (what is the problem you are overcoming?)

c) Which of your two changes is likely to make the biggest difference in performance, and why?

[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 7-stage pipeline, with the following stages:

F1 F2 D E M1 M2 WB

a) Assuming this machine has a branch predictor and the branch condition is calculated by the end of the F2 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 E?

b) How many load delay slots would this machine need (assuming it has forwarding logic and you are forwarding to E) 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] Assuming a 15-bit address and a 1024-byte Direct Mapped cache with a linesize=8, show how an address is partitioned/interpreted by the cache.

[3] Assuming a 15-bit address and a 96-byte 3-way Set Associative cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

[2] Assuming a 15-bit address and a 426-byte Fully Associative cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

[2] Given a 4 Megabyte physical memory, a 25 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.

[5] Given a 1 Gigabyte physical memory, a 44 bit Virtual address, and a page size of 16K 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?

In this question, we are going to wire up a 14-bit processor. The machine is word-addressable, where a word is 14 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	Unused	
13-8	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	
13-8	7-6	5-4	3-0	

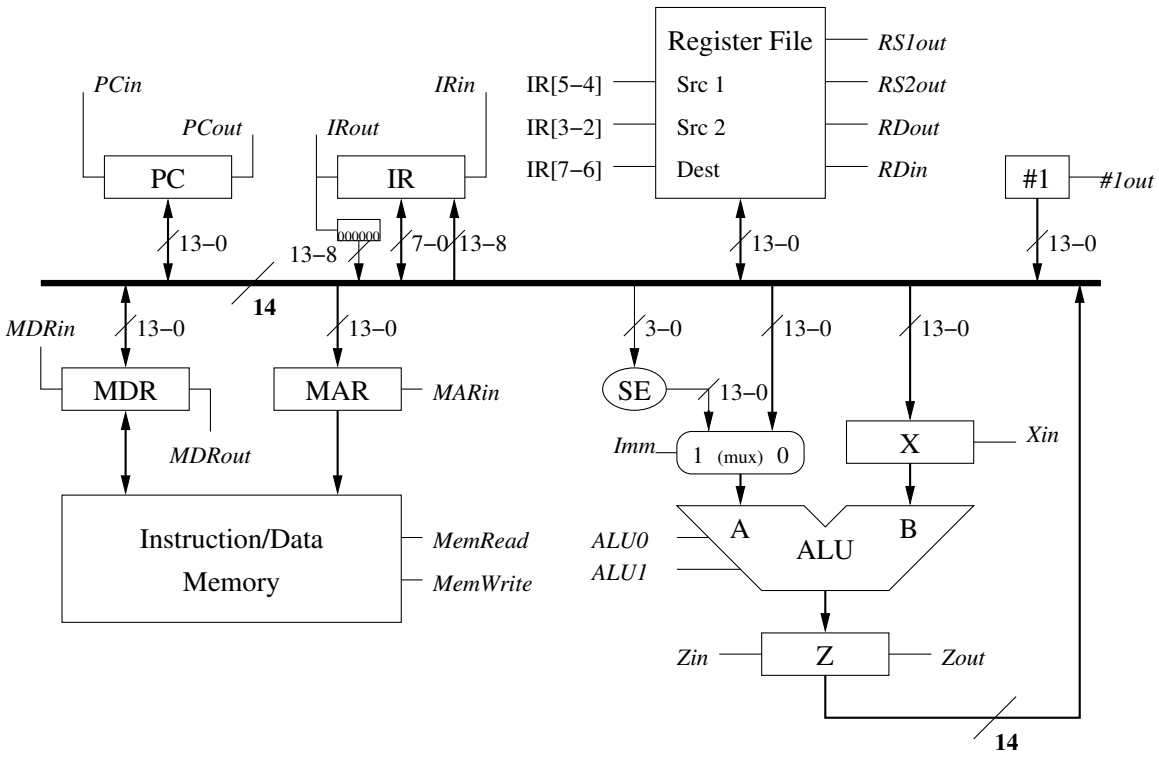
J-type:	*(Jump:*	*PC = Offset)*
Opcode	Offset	
13-8	7-0	

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

Here are some of the instructions that have been defined:

Name	**Opcode**	**Name**	**Opcode**	**Name**	**Opcode**
NOP	000000	lw	000100	sw	001100
SUB	100000	BEQZ	010000	j	010100
AND	100001	AND Imm	100100	ADD	110001
OR	100010	OR Imm	101000	ADD Imm	110100
XOR	100011	XOR Imm	101100	SUB Imm	110001

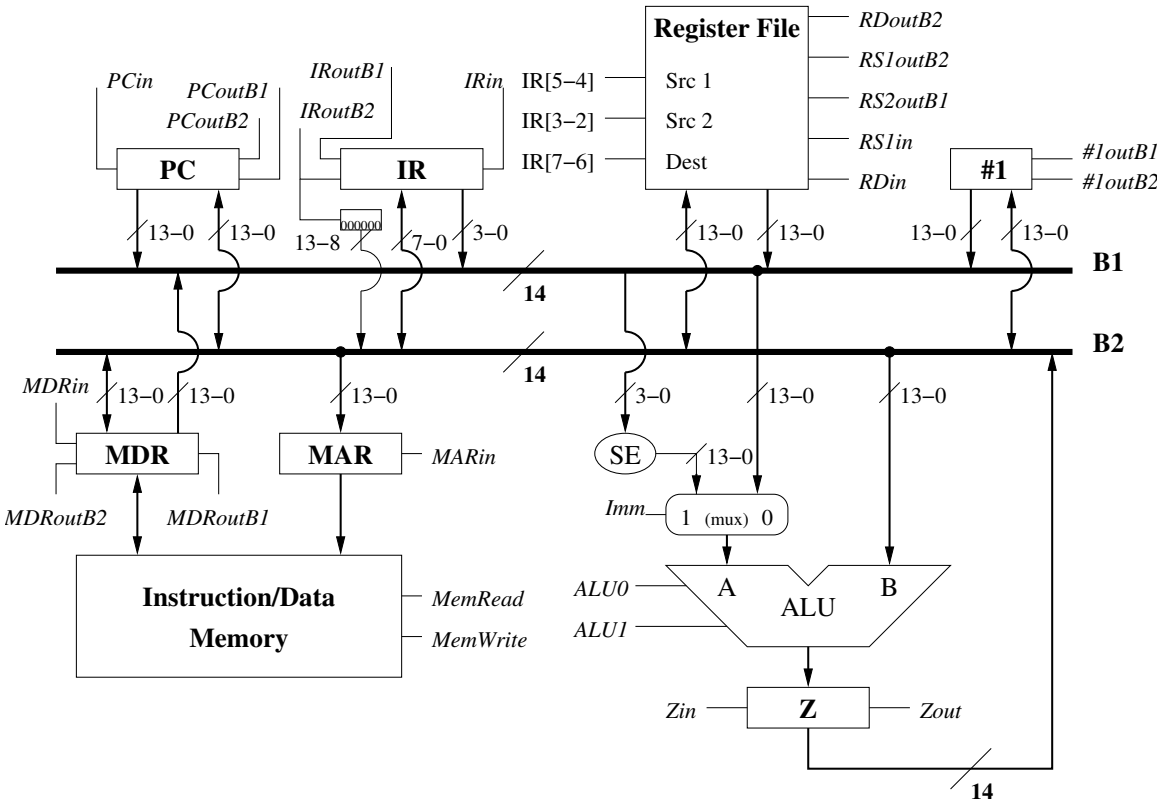
Here is a diagram of the machine using a single bus.



[8] Fill in the microcode steps necessary to perform an instruction fetch (incrementing the PC is considered part of fetch). Assume the memory can respond in a single cycle.

Step	PCout	IRout	MDRout	Zout	RDout	RS1out	RS2out	#1out	PCin	IRin	MARin	MDRin	Xin	Zin	RS2ou	ALU1	ALU0	MemRead	MemWrite	Imm
0																				
1																				
2																				
3																				
4																				
5																				

Here is a diagram of the machine using 2 busses.



[8] Fill in the microcode steps (including instruction fetch) necessary to perform the following instruction: **XORImm R2, R0, #3**

Step	PCoutB1	PCoutB2	IRoutB1	IRoutB2	RDoutB2	RS1outB2	RS2outB1	MDRoutB1	MDRoutB2	#1outB1	#1outB2	Zout	PCin	IRin	RDin	RS1in	Zin	MARin	MDRin	ALU1	ALU0	MemRead	MemWrite	Imm
0																								
1																								
2																								
3																								
4																								
5																								
6																								

[10] Your company is considering adding an auto incrementing addressing mode to this machine. This addressing mode is indicated using a "+" sign, which indicates that the register associated with the "+" sign should be incremented. For example, ADD R0,R1+,R2 would add R1 to R2, place the result in R0, and then increment R1.

Given this information, write the microcode for this new instruction: **SW R2, {4(R0+)}** Assume the memory can respond in a single cycle. (**Note** - *for this problem you can assume the instruction fetch steps have already occurred.*)

Step	PCoutB1	PCoutB2	IRoutB1	IRoutB2	RDoutB2	RS1outB2	RS2outB1	MDRoutB1	MDRoutB2	#1outB1	#1outB2	Zout	PCin	IRin	RDin	RS1in	Zin	MARin	MDRin	ALU1	ALU0	Mread	MemWrite	Imm
0																								
1																								
2																								
3																								
4																								
5																								
6																								
7																								

---

[5] In your first design of a 5-stage pipeline (F/D,E,M1,M2,W) F/D takes 48 time units, E takes 25, M1 takes 12, M2 takes 14, and W takes 25.

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 make it more balanced. What would the cycle time be now?

[6] Here is a code sequence. This is (obviously) generic code.

```
        INSTRUCTION 1

        INSTRUCTION 2

        NOP

        INSTRUCTION 3

        INSTRUCTION 4

Label:  INSTRUCTION 5

        NOP

        INSTRUCTION 6

        NOP

        NOP

        INSTRUCTION 7

        INSTRUCTION 8

        INSTRUCTION 9

        breq    CONDITION,Label         ; branch to Label if CONDITION
```

Assume there are the following dependencies in this code:

RAW between "INSTRUCTION 2" and "INSTRUCTION 3"
RAW between "INSTRUCTION 5" and "INSTRUCTION 6"
RAW between "INSTRUCTION 6" and "INSTRUCTION 7"

WAW between "INSTRUCTION 3" and "INSTRUCTION 4"

WAR between "INSTRUCTION 8" and "INSTRUCTION 9"

Draw in the arrows, and then indicate how you would schedule the code to remove the maximum number of stalls. How many are left when you are done?

[12] Here is a code sequence.

```
store   R6, 100(R7)

and     R5, R4, R6

xor     R5, R7, R8

load    R1, 50(R5)

add     R3, R9, R1

and     R9, R4, 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) There are things the static scheduler does not know which makes guaranteeing correctness difficult. What if, at execution time, R5=200 and R7=150? Will your dependency graph and ability to schedule this code change? If so, explain/show how.

[10] Here is a code sequence.

```
add     R5, R4, R7

load    R2, 20(R5)

sub     R3, R9, R2

xor     R3, R7, R1

and     R1, R4, R7
```

Assuming a 6-stage pipeline (F,D,E1,E2,M,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 from a load is needed at the beginning of E1. Also assume that on an arithmetic instruction, data is ready at the end of E2, but not at the end of E1.

[8] Assume our machine has 8 logical and 16 physical registers. On the left below is the code you are dealing with. On the right below is the register mapping upon entering the code sequence.

ADD R1, R3, R5

OR R1, R1, R2

XOR R2, R3, R6

BEFORE	
Logical	Physical
0	7
1	8
2	9
3	10
4	11
5	12
6	13
7	14

Free Pool: 15,0,1,2,3,4,5,6

(a) Indicate all the dependencies on the code segment above.

(b) Rewrite the code sequence below using the actual physical register names instead of the logical ones.

ADD P___, P___, P12

OR P___, P___, P9

XOR P___, P___, P13

(c) Now indicate all the dependencies in the above renamed code.

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

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

Fill in the following table assuming fine-grained scheduling is being used.

Time	Slot1	Slot2	Slot3
0			
1			
2			
3			

Fill in the following table assuming the use of coarse-grained scheduling.

Time	Slot 1	Slot2	Slot3
0			
1			
2			
3			

Now, repeat the process assuming the use of simultaneous multithreading on a 5-issue multithreaded machine with 4 threads - A, B, C and D. Assume:

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

Time	Slot1	Slot2	Slot3	Slot4	Slot 5
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	11	12	13	14
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	
i+7								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	11	12	13	14
i	IF	ID	RR	EX	M1	M2	WB			<- M1 stage has page fault				
i+1		IF	ID	RR	EX	M1	M2	WB		<- ID stage has page fault				
i+2			IF	ID	RR	EX	M1	M2	WB	<- EX stage has arithmetic fault				
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 (assuming no hardware errors)? Explain how you got your answer.