1. (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. (2) What are the two main ways to define performance?

3. (2) What is a "balanced" pipeline?

4. (4) What do we mean when we say something is an N-operand machine?

5. (4) Artificial intelligence techniques can provide more accurate branch prediction than existing approaches - why are these techniques not used?

6. (4) Do benchmark programs remain valid indefinitely? Why or why not?

7. (3) Is it possible to have a WAW hazard in the standard MIPS 5-stage pipeline? Why or why not?

8. (2) What piece of hardware differentiates correlating predictors from other predictors?

9. (2) When dealing with control hazards, a prediction is not enough - what else is necessary in order to eliminate stalls?

10. (4) We talked about static scheduling in class - what is one thing that static schedulers have trouble identifying, which dynamic schedulers do not?

11. (6) You are in charge of designing a new embedded machine, and you must use a fixed 24 bit instruction size (for a variety of reasons). You would like to support 128 different operations, use a 3-operand instruction format, and have 64 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 the problem.

12. (7) The standard MIPS has a 5-stage pipeline, and uses a branch 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)

    a. Assuming this machine has a branch predictor and the branch condition is calculated by the end of the E stage, how big is the branch penalty (measured in cycles) when the prediction is incorrect?

    b. How many load delay slots would this machine need, assuming it has forwarding logic and the value has returned from memory by the end of M2?

13. (13) Here is a code sequence.

```
lw      R1, 0(R10)

sw      R2, 4(R10)

lw      R3, 8(R1)

add     R4, R3, R1

sw      R1, 20(R0)

lw      R4, 4(R2)

add     R5, R1, R4
```

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 NOPs are left (if any?)

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

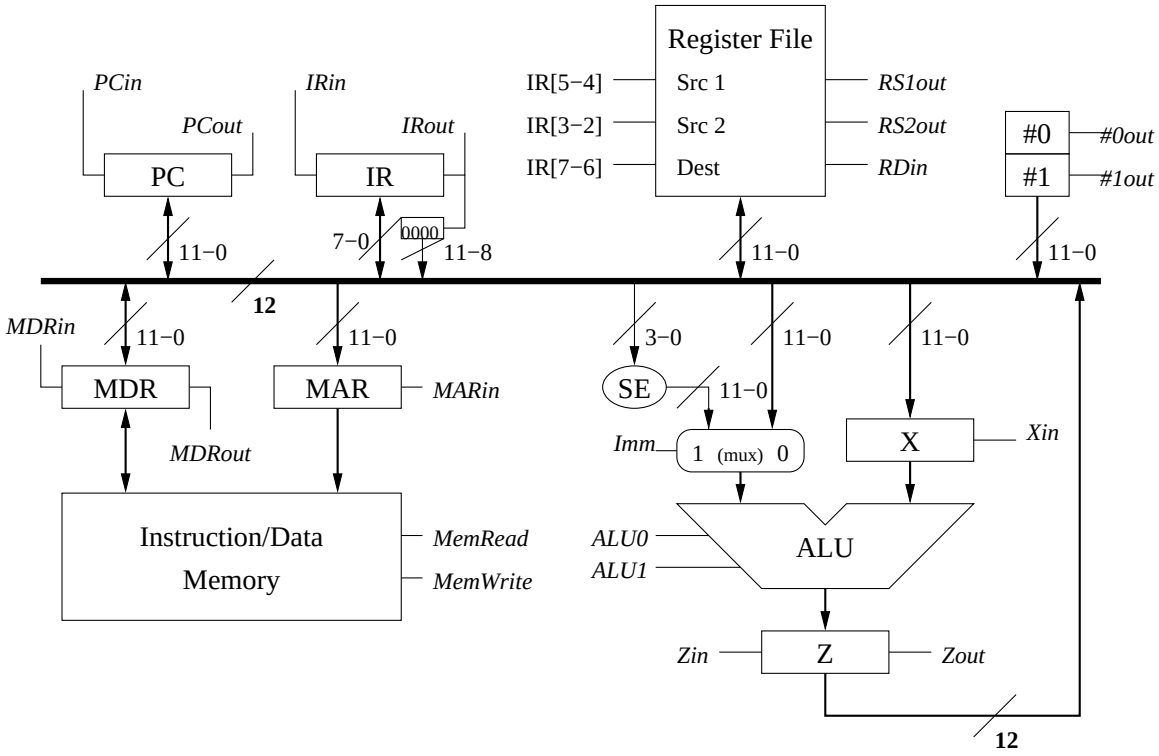
Here are some of the instructions that have been defined:

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.



14. (9) Fill in the microcode steps necessary to perform an instruction 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																					

15. (4) 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: AND Immd R2, R1, 3

16. (6) Now that you have done the instruction fetch, fill in the microcode steps necessary to perform the above instruction (AND Immd R2, R1, 3)

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																					

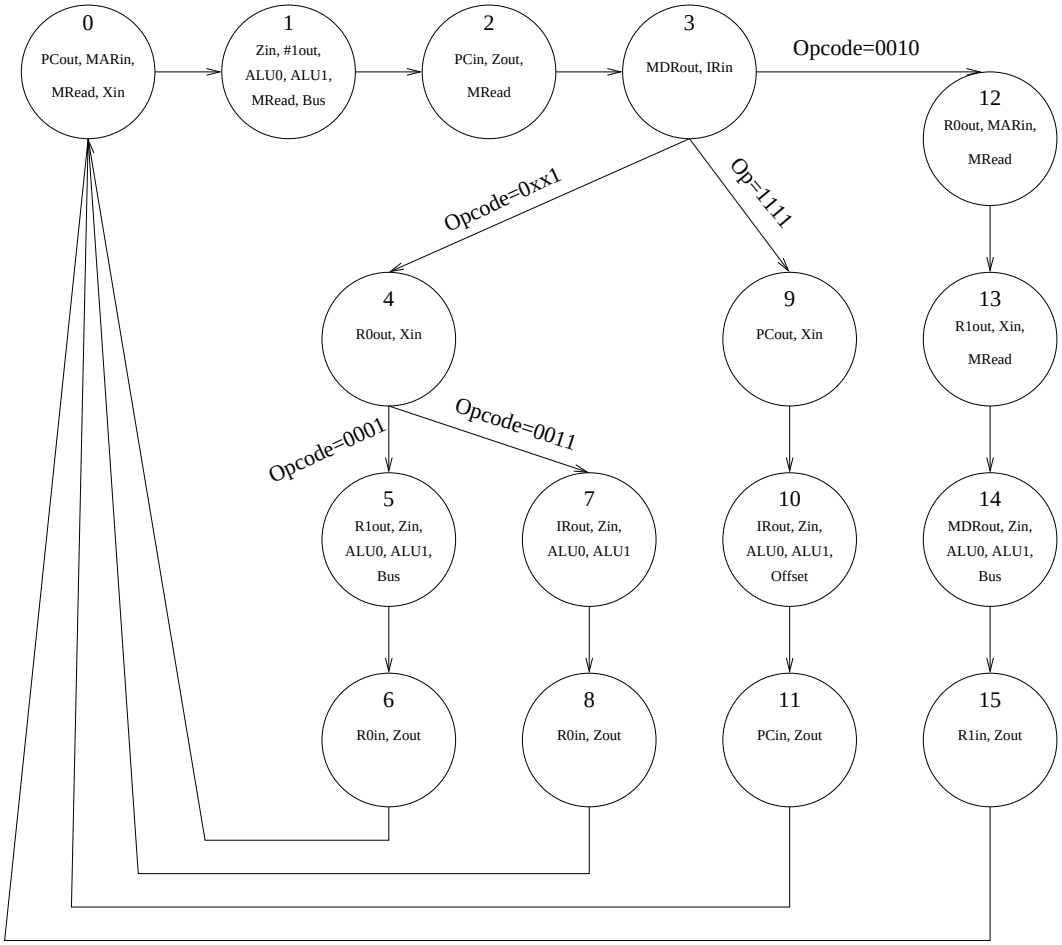
17. (4) We want to use microcode to provide the control signals for the above machine. Assuming the longest instruction takes 10 cycles and you can use the simplest microcode configuration,

    a. How many entries would the microcode memory have?

    b. How wide would each entry be?

    (Drawing a picture might be useful).

Here is the state diagram for a random machine:



18. (4) Assuming there are 4 state variables (Y3-Y0), that State0=!Y3*!Y2*!Y1*!Y0 (0000) and State15=Y3*Y2*Y1*Y0 (1111), write down the exact boolean equation for the R1out signal.

19. (4) Assuming the same situation as in the previous question, write down the exact boolean equation for NextState5.

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

21. (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	
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 experiences a fault in the MEM1 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

What happens in this case?

	1	2	3	4	5	6	7	8	9	10		
i	IF	ID	RR	EX	M1	M2	WB	<- M2 stage has Page Fault				
i+1		IF	ID	RR	EX	M1	M2	WB	<- EX unit has overflow error			
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

---

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