[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] What are the two main ways to define performance?

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

[3] Over a period of many years, clock rates grew by a factor of 1000 while power consumed only increased by a factor of 30. How was this accomplished? Why is this technique no longer effective?

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

[4] Briefly describe a Tournament Branch Predictor. Be sure to touch on how it works and what characteristic of branch predictors it is exploiting. Pictures are welcome.

[3] Given the binary bit pattern

x x x x x x 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 1 0 1 1

If this is a MIPS jump instruction, write down exactly what happens (in other words, what is the binary value that is put into the PC?)

[4] It is difficult for the internal processing elements on a CMOS chip to cross the chip boundary and communicate with things that are on other chips. Explain why that is.

[3] You are in charge of designing a new embedded machine, and for a variety of reasons you **must** use a fixed 20 bit instruction size. You would like to support 128 different operations, 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 two different ways (excluding reducing the number of instructions) to fix the problem.

[3] 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 either make the Floating Point unit 90% faster (take 10% as long), or the integer unit 40% faster (take 60% as long). Which should you do, and why?

[11] You are given a machine with a 9 stage pipeline, which writes to the register file during the D1 cycle and reads during D2. This machine uses neither a branch delay slot nor a load delay slot. These are the stages:

F D1 D2 E1 E2 M1 M2 M3 W

a. Assuming this machine has a branch predictor and the branch condition is calculated by the end of the D2 stage, how big is the branch penalty (measured in cycles) when the prediction is incorrect? What if the branch condition is calculated by the end of E1?

b. Assume on a load the value is available at the end of M3. How many stall cycles stalls would this machine need on a load if it has forwarding logic and you are forwarding to E1? What if you were forwarding to E2?

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?

[15] Here is a code sequence.

```
load    R5, 0(R2)

load    R2, 4(R6)

add     R3, R2, R1

sub     R8,R7,R6

store   R3, 8(R9)

and     R8,R9,R10
```

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?

e) If R9 contains the value 0x12, what address in memory is written to when executing the store instruction?

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]
Decrement X [111], Increment X [110], Add [011], SUB [100], AND [101], OR [001], NOT [010]

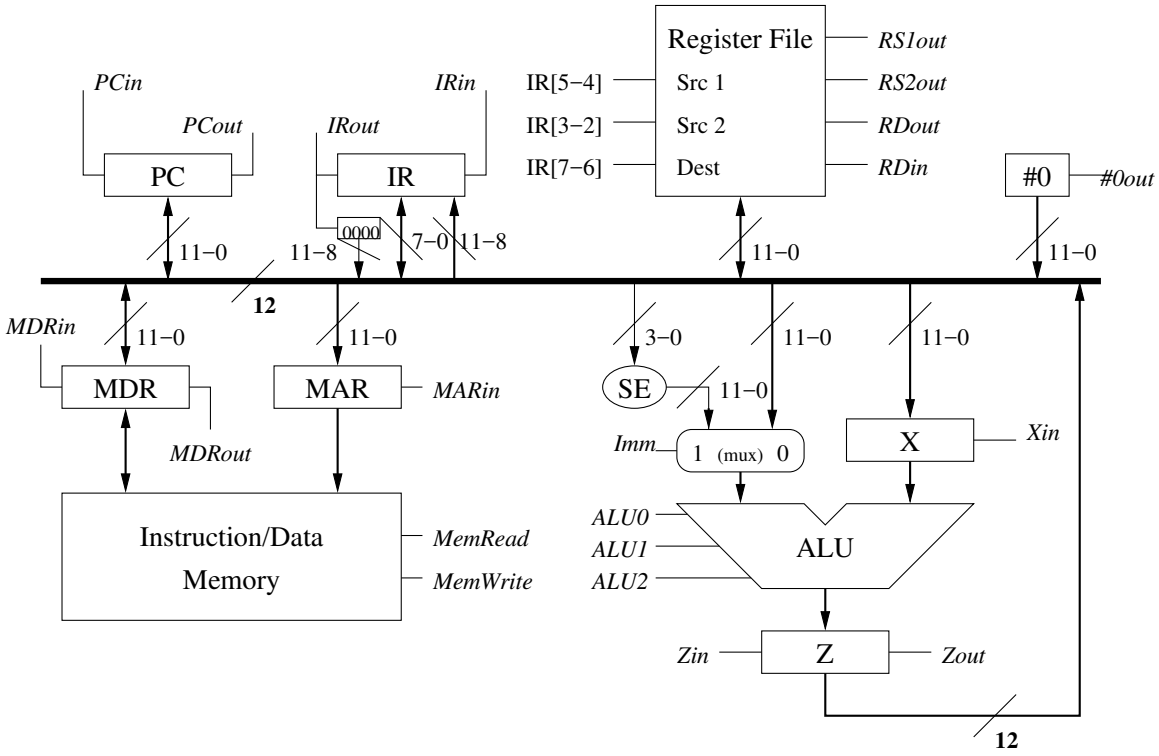
Here are some of the instructions that have been defined:

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.



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

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																					

[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: **XOR R0,R1,R2**

[8] Now that you have done the instruction fetch, fill in the microcode steps necessary to perform the following instruction: **SW R0, 15[R2]** Assume memory can respond in a single cycle.

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																					

---

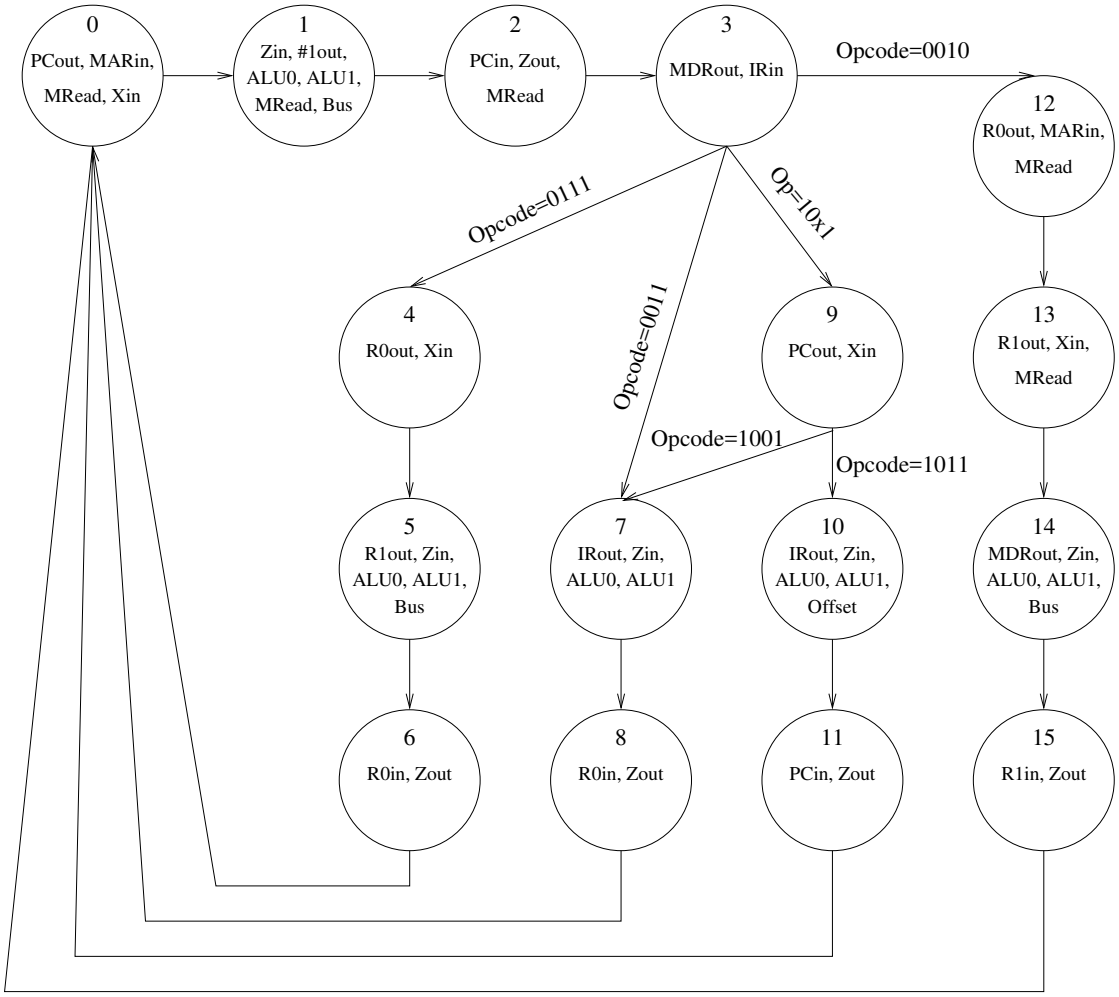
[2] The multicycle MIPS uses a single memory, but the pipelined MIPS design has two. Why is that?

[4] In your first design of a 5-stage pipeline (F,D,E,M,W), F takes 24 time units, D takes 26, E takes 25, M takes 20, and W takes 5.

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?

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 **IRout** signal.

[4] Assuming the same situation as in the previous question, write down the exact boolean equation for **NextState10**.

[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 6 stage pipeline):

	1	2	3	4	5	6	7	8	9	
i	IF	ID	EX	M1	M2	WB	<- Interrupt detected			
i+1		IF	ID	EX	M1	M2	WB	<- Instruction Squashed		
i+2			IF	ID	EX	M1	M2	WB	<- Trap Handler fetched	
i+3				IF	ID	EX	M1	M2	WB	
i+4					IF	ID	EX	M1	M2	WB

Fill out the following table if instruction i+1 experiences a fault in the EX stage (Overflow, for example) and we are assuming **imprecise** interrupts:

	1	2	3	4	5	6	7	8	9	10	
i	IF	ID	EX	M1	M2	WB					
i+1		IF	ID	EX	M1	M2	WB				
i+2			IF	ID	EX	M1	M2	WB			
i+3				IF	ID	EX	M1	M2	WB		
i+4					IF	ID	EX	M1	M2	WB	
i+5						IF	ID	EX	M1	M2	WB

Assuming we are supporting **precise** interrupts, what happens in this case?

	1	2	3	4	5	6	7	8	9	10		
i	IF	ID	EX	M1	M2	WB	<- M1 stage has Page Fault					
i+1		IF	ID	EX	M1	M2	WB					
i+2			IF	ID	EX	M1	M2	WB	<- Illegal Instruction detected			
i+3				IF	ID	EX	M1	M2	WB			
i+4					IF	ID	EX	M1	M2	WB		
i+5						IF	ID	EX	M1	M2	WB	
i+6							IF	ID	EX	M1	M2	WB
i+7								IF	ID	EX	M1	M2

---

[3] What is the maximum number of exceptions that could happen at one time in the above machine? Why?