

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

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

3. (3) What is an "unbalanced" pipeline?

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

5. (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?

6. (3) Is it true that more powerful instructions (instructions that do more) will result in better performance? Why or why not?

7. (3) Do benchmark programs remain valid indefinitely? Why or why not?

8. (4) What does a correlating predictor attempt to do? What piece of hardware does it use to accomplish this?

9. (5) You are in charge of 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 128 different operations, use a 2-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.

10. (10) The standard MIPS has a 5-stage pipeline, and uses a branch delay slot. If the machine is redesigned to be an 8-stage pipeline, with the following stages:

F D RR E1 E2 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 E2 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 the RR stage?

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? (Assume data is needed at the beginning of the E1 stage - nothing can forward into E2.)

What is necessary in order to ensure that this pipeline does not have to worry about WAW hazards?

11. (17) Here is a code sequence.

sw R9, 0(R11)

lw R1, 0(R10)

add R2, R3, R1

lw R3, 8(R1)

add R4, R3, R1

sw R4, 20(R0)

lw R5, 4(R2)

add R5, R5, 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 [01], ADD [11], OR [10], NOT [00]

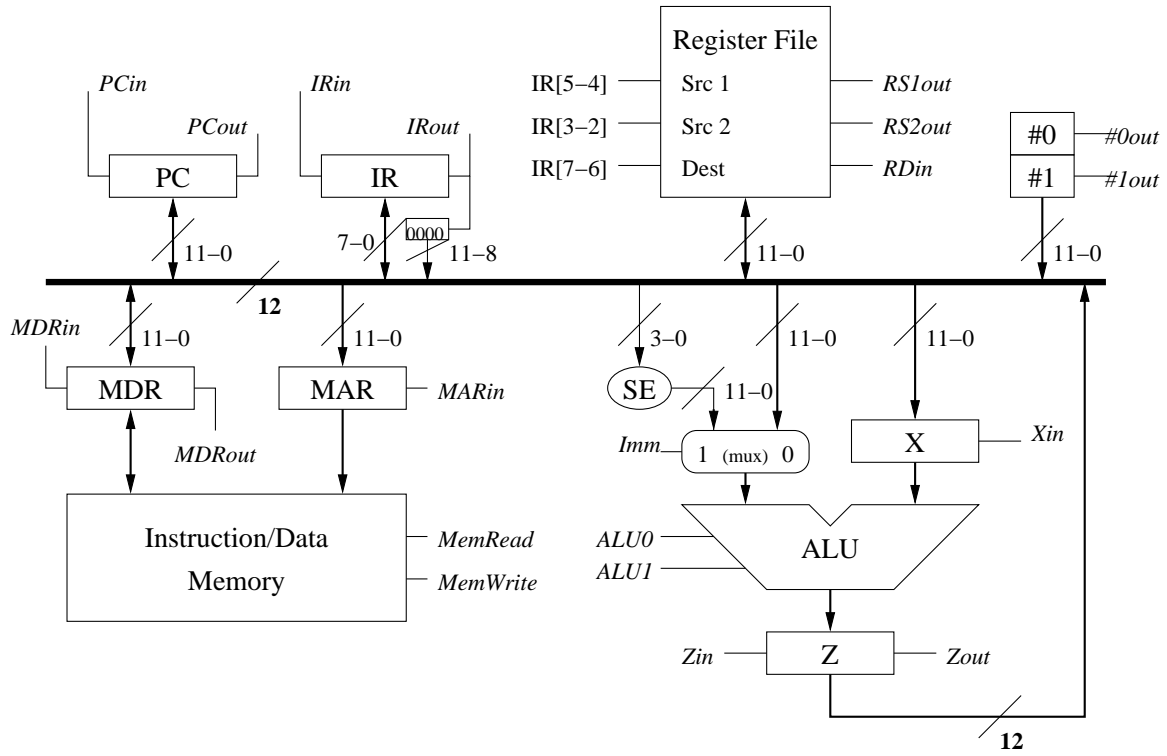
Here are some of the instructions that have been defined:

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.



12. (8) Fill in the microcode steps necessary to perform an instruction fetch. (Incrementing the PC is considered part of the fetch).

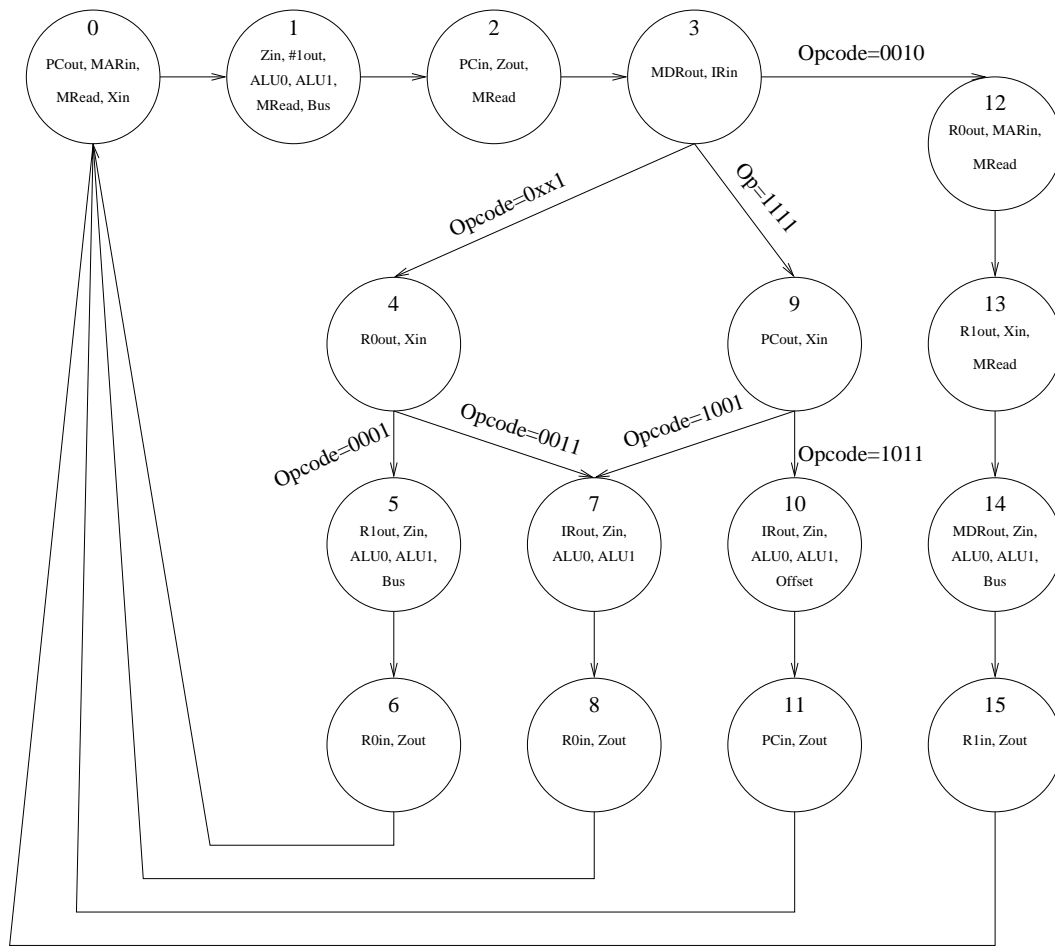
S t e p	P C i n	I R i n	M A R i n	M D R i n	M D R o u t	X i n	Z i n	P C o u t	I R o u t	M D R o u t	Z o u t	R D i n	R S 1 o u t	R S 2 o u t	# 0 o u t	# 1 o u t	A L U 0	A L U 1	M e m r e a d	M e m w r i t e	I m m
0																					
1																					
2																					
3																					
4																					

13. (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: J 0x2A

14. (4) Now that you have done the instruction fetch, fill in the microcode steps necessary to perform the above instruction (J 0x2A)

S t e p	P C i n	I R i n	M A R i n	M D R i n	R D i n	X i n	Z i n	P C o u t	I R o u t	M D R o u t	Z o u t	R D o u t	R S o u t	R S o u t	# o u t	# o u t	A L U 0	A L U 1	M r e a d	M w r i t e	I m m	
0																						
1																						
2																						
3																						
4																						
5																						

Here is the state diagram for a random machine:



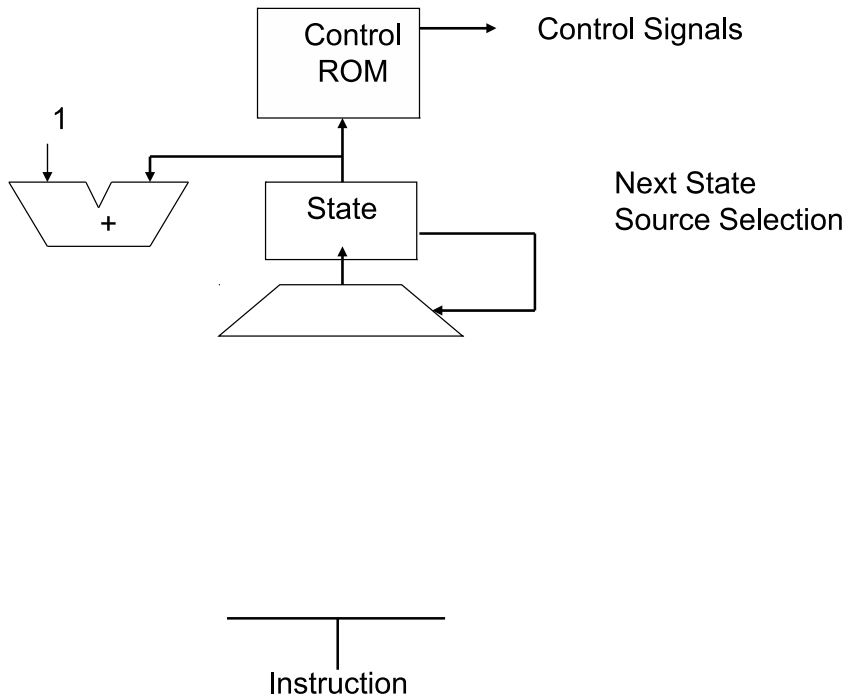
15. (4) Assuming there are 4 state variables (Y_3 - Y_0), that $\text{State0} = \neg Y_3 \neg Y_2 \neg Y_1 \neg Y_0$ (0000) and $\text{State15} = Y_3 Y_2 Y_1 Y_0$ (1111), write down the exact boolean equation for the MARin signal.
16. (4) Assuming the same situation as in the previous question, write down the exact boolean equation for NextState7 .

17. (4) We want to use microcode to provide the control signals for the machine on the previous page. Assuming there are 23 control signals, the machine has a 5 bit opcode field, and you can use the simplest microcode configuration,

a. How many entries would the microcode memory have?

b. How wide would each entry be?

18. (10) Here is an incomplete picture of the minimized microcode configuration. Fill in the missing pieces.



a. How many entries are there in the control rom?

b. How wide is each entry?

c. How many entries in the state rom?

d. How wide is each entry?

19. (4) 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+1 experiences a fault in the EX stage:

	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

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

