[4] 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?

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

[4] MIPS has 3 instruction formats - R, I and J. If you want to do a store instruction, which format would you use?

[4] Given the instruction **load R5, 8(R9)**
If R9 contains the value 0x12, what memory address is generated when executing the load instruction?

[4] What is the difference between fixed and hybrid instruction encodings?

[4] Registers are faster than main memory. What is another advantage to using registers? What is one disadvantage?

[4] Is it true that more powerful instructions (instructions that do more) mean higher performance? Explain your answer.

[4] Given the value **0x12345678** and the addresses

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100

Write below each of the above addresses the appropriate value, assuming the data is being stored in little endian format starting at location 0010 (there may be more addresses than you need). What type of alignment would this be an example of?

[4] We want to use microcode to provide the control signals for a given machine which has 18 control signals and a 6-bit opcode. Assuming there are 14 states in the State Transition Diagram and you are using 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).

[8] In the MIPS processor, there are branches and there are jumps.

(a) for a jump, explain exactly how the new address to be placed in the PC is created/calculated.

(b) for a branch, explain exactly how the new address to be placed in the PC is created/calculated (assume the branch is taken).

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

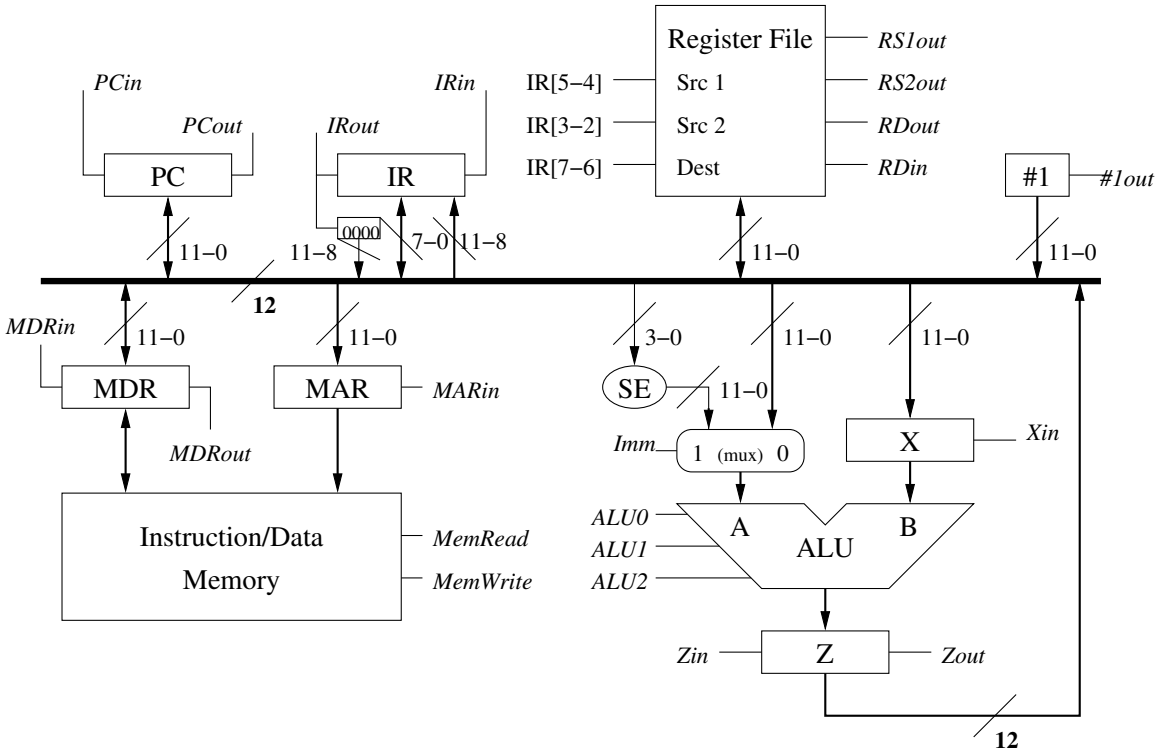
Here are some of the instructions that have been defined:

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	#1out	ALU0	ALU1	ALU2	Xin

Here is a diagram of the machine.



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

Step	Xin	Zin	MARin	MDRin	RDin	IRin	PCin	IRout	PCout	MDRout	Zout	RDout	RS1out	RS2out	#1out	ALU2	ALU1	ALU0	Mread	Mwrite	Imm
0																					
1																					
2																					
3																					
4																					
5																					

[6] 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: **SUB R1,R2,R3**

[8] Now that you have done the instruction fetch, fill in the microcode steps necessary to perform the following instruction: **OR R0,R1,R3**

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

[8] Assuming you have done the instruction fetch, fill in the microcode steps necessary to perform the following instruction: **SW R2, 4(R3)**

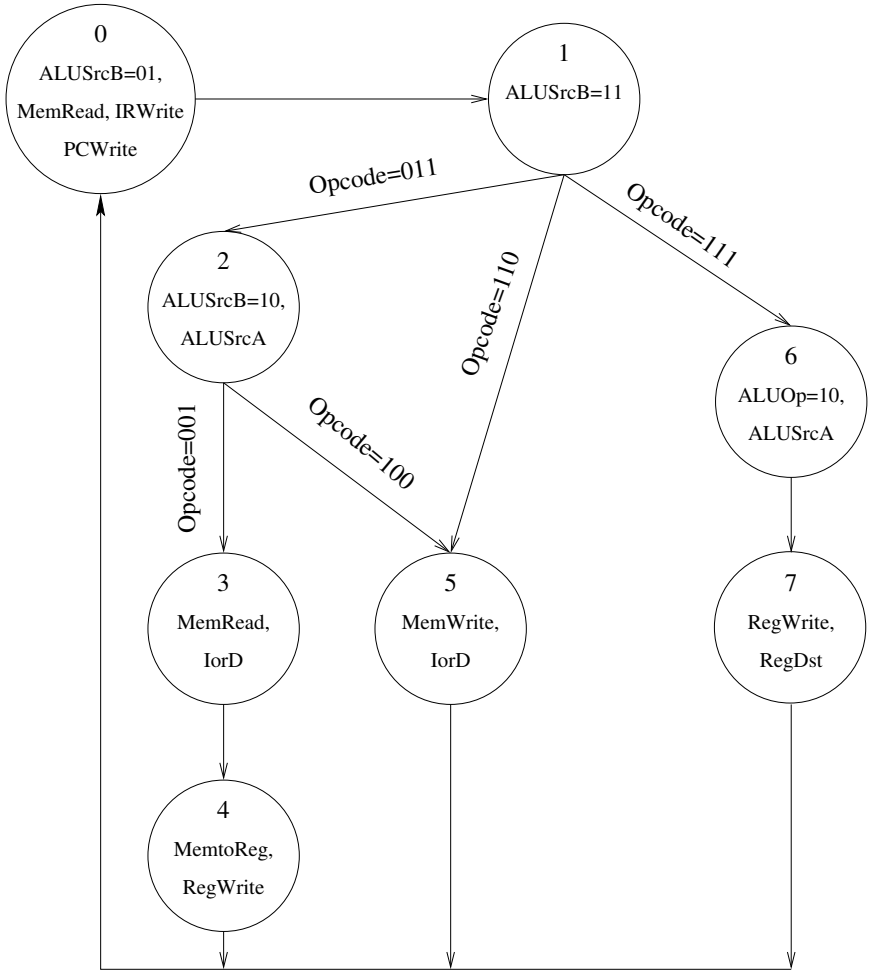
Step	Xin	Zin	MARin	MDRin	RDin	IRin	PCin	IRout	PCout	MDRout	Zout	RDout	RS1out	RS2out	#1out	ALU2	ALU1	ALU0	Mread	Mwrite	Imm
0																					
1																					
2																					
3																					
4																					

[2] Assuming you have done the instruction fetch, fill in the microcode steps necessary to perform the following instruction: **J 0x23**

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

[8] You have been asked to design an instruction set which has a fixed size of 15 bits. You would like to support 32 different operations, use a 3-operand instruction format, and support 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 two different things you could do to fix the problem and draw each solution. (For a number of reasons, the size of the instruction cannot be changed, nor can the number of instructions.)

Here is the state diagram for a "random" machine:



[4] Assuming there are 3 state variables (Y2-Y0), that State0 = $\overline{Y2}$*$\overline{Y1}$*$\overline{Y0}$ (000) and State7 = Y2*Y1*Y0 (111), write down the exact boolean equation for the **MemRead** signal.

[4] Assuming the same situation as in the previous question, write down the exact boolean equation for **NextState6**.

[4] If you were using a minimized microcode configuration for this machine, circle on the diagram where the dispatch rom points are.