We are going to show from beginning to end how to wire up a generic 8-bit machine. This machine will use a 2-operand format, meaning that instructions are of the time A=A+B. So, for example, "Add r0, r1" is r0=r0+r1.

The machine is byte-addressable. Offsets are sign-extended, and jumps are done by adding the sign-extended offset field to the PC. Immediates are not sign-extended.

The machine has 3 different instruction formats: A, B, and C.

A-type:	Opcode	ds	s	extra
	7-4	3	2	1-0

B-type:	Opcode	ds	Immediate
	7-4	3	2-0

C-type:	Opcode	Offset
	7-4	3-0

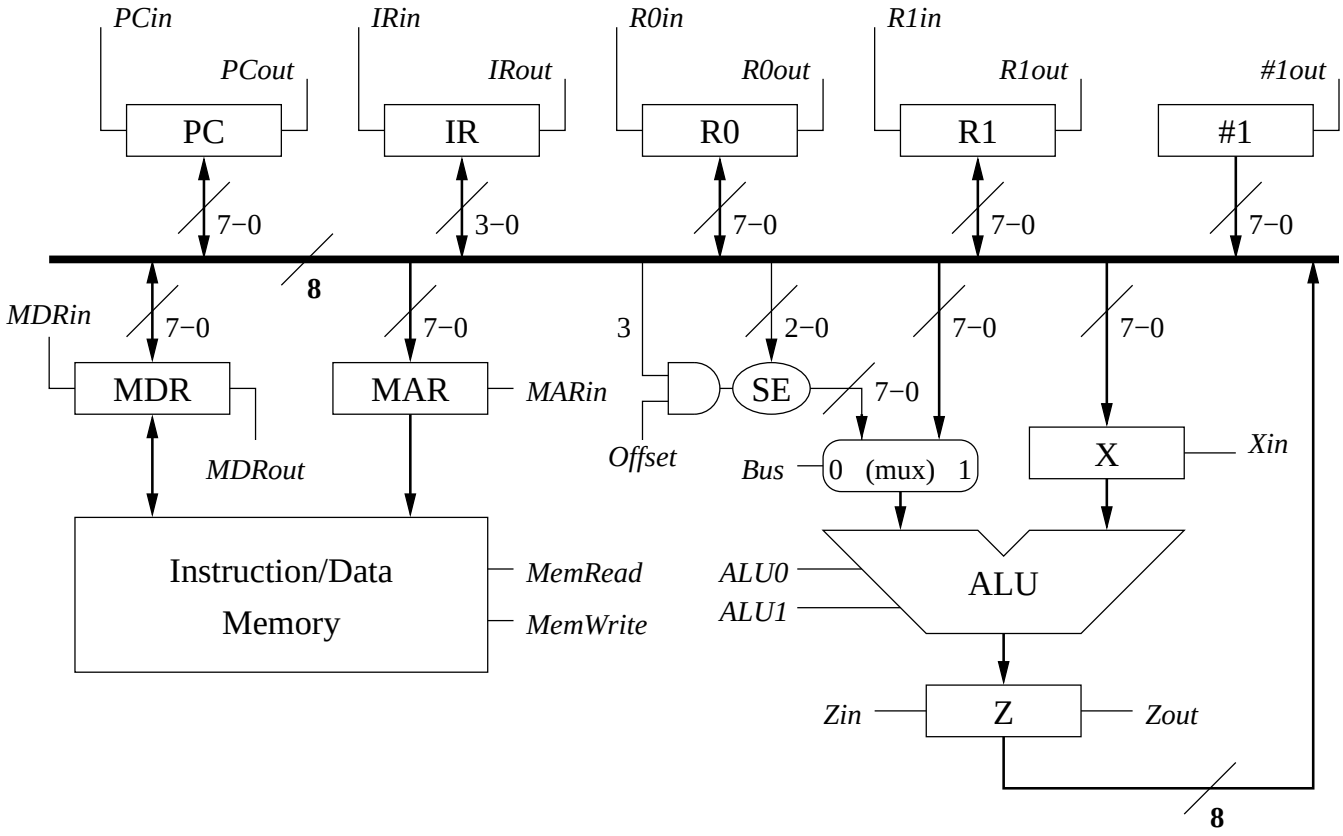
The ALU can perform 4 functions:

Operation	**ALU0**	**ALU1**
Add	1	1
Or	1	0
And	0	1
Not A	0	0

Here are a few of the instructions that have been defined:

Name	**Opcode**	**Operation**
add	0001	rds=rds+rs
addm	0010	rds=rds+mem[rs]
addi	0011	rds=rds+imm
jmp	1111	PC=PC+offset (offset is sign extended)

On the following page is a diagram of the machine. The control signals are in italics. MAR does not need an "out" signal, because the MemRead and MemWrite signals control whether or not memory looks at the MAR signal. The X register does not need an "out" signal either - the ALU will always perform an operation, which we can ignore if we want. The "#1" register contains the number 1, which is used to increment the PC. The SE block is sign-extend logic, and needs some further explanation. The problem is that the ALU requires an 8-bit value, but the Immediate field is only 3 bits wide, and the offset field is only 4 bits wide and also needs to be sign-extended. The sign extend logic creates a 5-bit value which matches the contents of bit 3, so that the 8-bit value generated looks like 33333210 (instead of 76543210). The AND gate is necessary because an immediate instruction is not sign-extended, and there needs to be a way to ensure that the top 5 bits are all set to zero.



Here are the 21 control signals.

PCin	PCout	IRin	IRout	R01	R0out	#1out
R1in	R1out	MDRin	MDRout	Zin	Zout	MARin
Xin	ALU0	ALU1	MemRead	MemWrite	Bus	Offset

In order to execute an instruction, we must make sure that each of these control signals is set to the appropriate value at the appropriate time. For example, here are the steps necessary to fetch an instruction from memory:

Step	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
2	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
3	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Once the instruction has been fetched, it must be executed. Here is how R0 = R0 + R1 would be performed:

Step	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
2	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Here is how a Jump would be performed (0's are removed for legibility reasons):

Step	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0		1												1							
1				1							1					1	1				1
2	1											1									

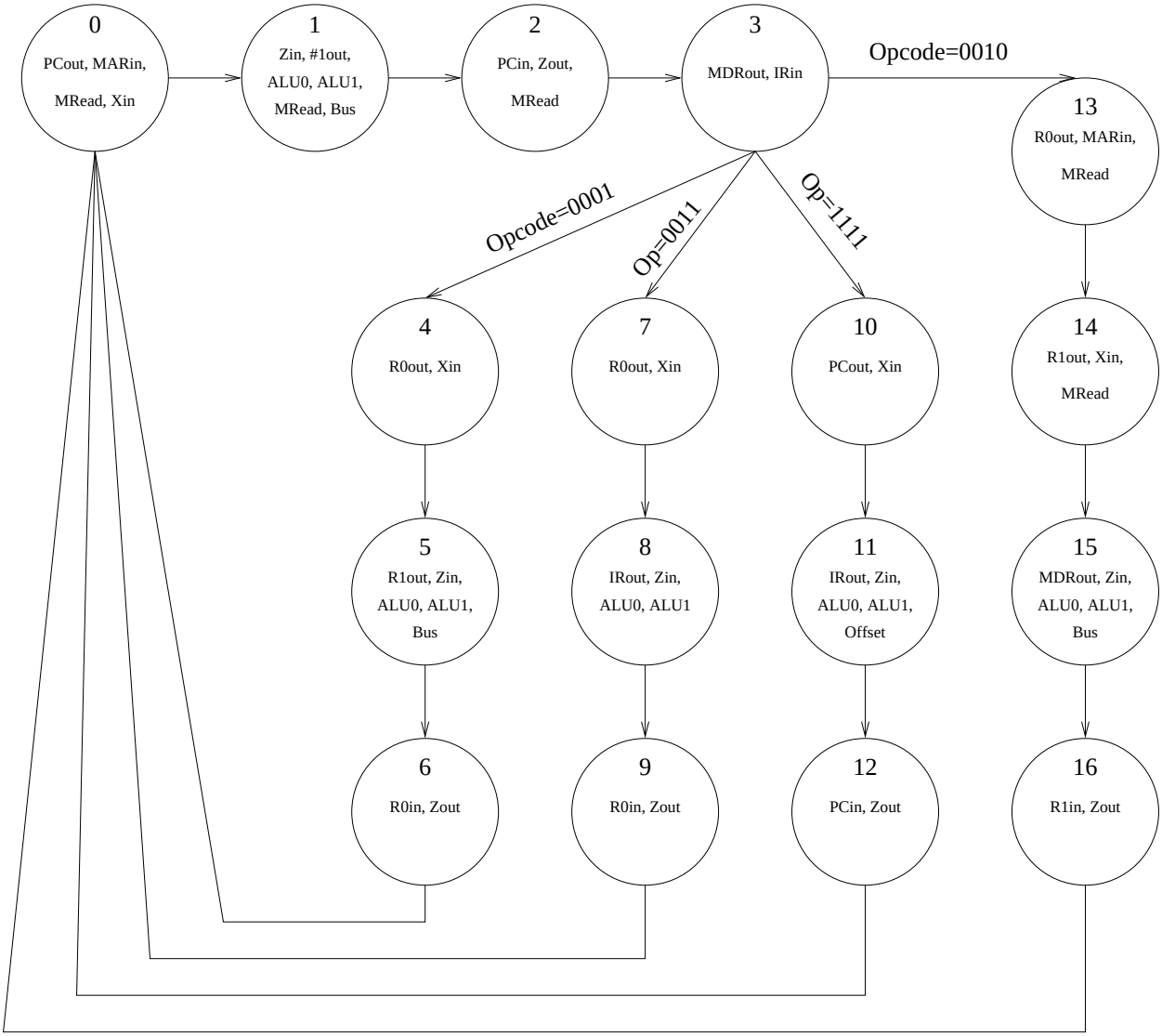
Here is an Immediate (R0 = R0 + immd) instruction:

Step	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0						1								1							
1				1							1					1	1				
2					1							1									

A memory access (R1 = R1 + mem[R0]):

Step	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0						1							1					1			
1								1						1				1			
2										1	1					1	1			1	
3							1					1									

Once every instruction has been fully specified in this manner, one can create sequential logic circuits which will cause each control signal to be set to the right value at the right time. In the following diagram, the State number is in the top center of each circle, and inside the circle are the signals that need to be asserted that cycle.



From this diagram we can create the following table, which describes the exact boolean equation for every control signal. Since there are 17 states, we will need 5 state variables, Y4-Y0. In this table, "State0" = 00000, "State1" = 00001, etc. Remember, when doing sequential circuits, both the current outputs and the next state values must be calculated. In this table, the NextState values are calculated based on the current state and the current inputs. (The current inputs are I7-I4, which are the opcode bits and come from the Instruction Register). This is a Moore model, since the outputs are a function of the current state only.

PCin=	State2 + State12	**PCout=**	State0 + State10
IRin=	State3	**IRout=**	State8 + State11
R0in=	State6 + State9	**R0out=**	State4 + State7 + State13
R1in=	State16	**R1out=**	State5 + State14
MDRin=		**MDRout=**	State3 + State15
Zin=	State1 + State5 + State8 + State11 + State15		
Zout=	State2 + State6 + State9 + State12 + State16		
MARin=	State0 + State13		
Xin=	State0 + State4 + State7 + State10 + State14	**#1out=**	State1
ALU0=	State1 + State5 + State8 + State11 + State15		
ALU1=	State1 + State5 + State8 + State11 + State15		
Mread=	State0 + State1 + State2 + State13 + State14	**Mwrite=**	
Bus=	State1 + State5 + State15	**Offset**	State11
NextState0=	State6 + State9 + State12 + State16	**NextState1=**	State0
NextState2=	State1	**NextState3=**	State2
NextState4=	State3 * !I7 * !I6 * !I5 * I4	**NextState5=**	State4
NextState6=	State5	**NextState7=**	State3 * !I7 * !I6 * I5 * I4
NextState8=	State7	**NextState9=**	State8
NextState10=	State3 * I7 * I6 * I5 * I4	**NextState11=**	State10
NextState12=	State11	**NextState13=**	State3 * !I7 * !I6 * I5 * !I4
NextState14=	State13	**NextState15=**	State14
NextState16=	State15		

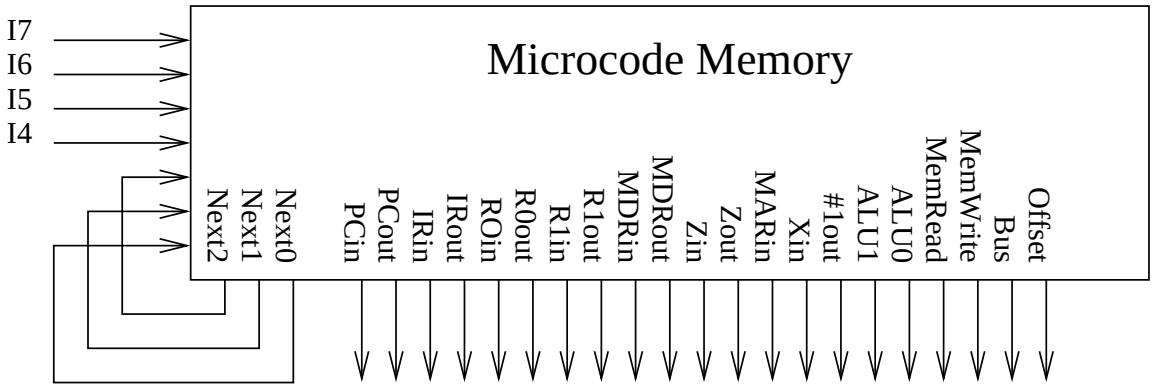
Here, for example, are the exact boolean equations for a couple of signals:

PCin = (!Y4 * !Y3 * !Y2 * Y1 * !Y0) + (!Y4 * Y3 * Y2 * !Y1 * !Y0)

R1in = (Y4 * !Y3 * !Y2 * !Y1 * !Y0)

NextState4 = !Y4 * !Y3 * !Y2 * Y1 * Y0 * !I7 * !I6 * !I5 * I4

If there are any changes to the design (additions or modifications), then it will obviously take substantial effort to generate an entire new set of sequential logic circuits. It is much easier to create a small memory which will contain the values of the control signals during each cycle, and then all we have to do is cycle through the memory correctly. For example, suppose I create a memory which is 24 bits wide, and addressed by using the top 4 bits of the opcode and the left most 3 bits of the memory output. It would look like this:

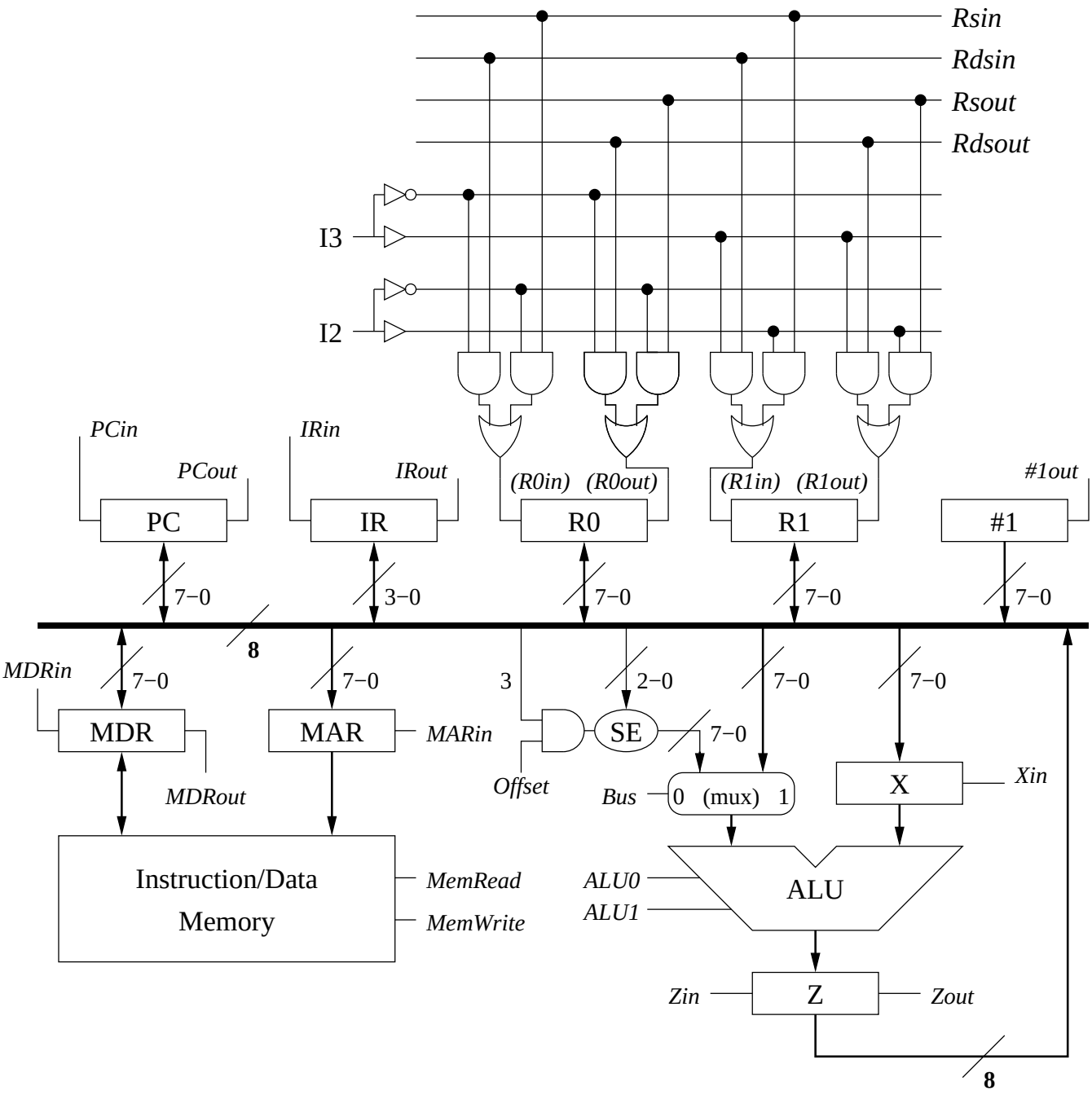


Using this approach, the control memory would look something the table on the following page. What happens is the following: The top 4 bits of the IR provide the location in the memory where the steps to execute that instruction reside. The Next bits step us through the sequence of operations. Once the sequence has completed, we return to the xxxx000 location and perform the instruction fetch. After the 4th step of the instruction fetch, the IR will be loaded with a new instruction, and we will then go to that part of the control memory and perform the appropriate steps.

Thus, in this case, the magic "decode" stage amounts to nothing more than jumping to the correct part of the memory! The table on the following page shows some of the contents of the microcode memory.

Control Memory Address **IIII NNN 7654 210**	Next2	Next1	Next0	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

As I discussed in class, the actual diagram needs to be modified a bit in order to incorporate the fact that the instruction specifies which register is to be the source and which is to be the destination. The modified diagram would look like this:

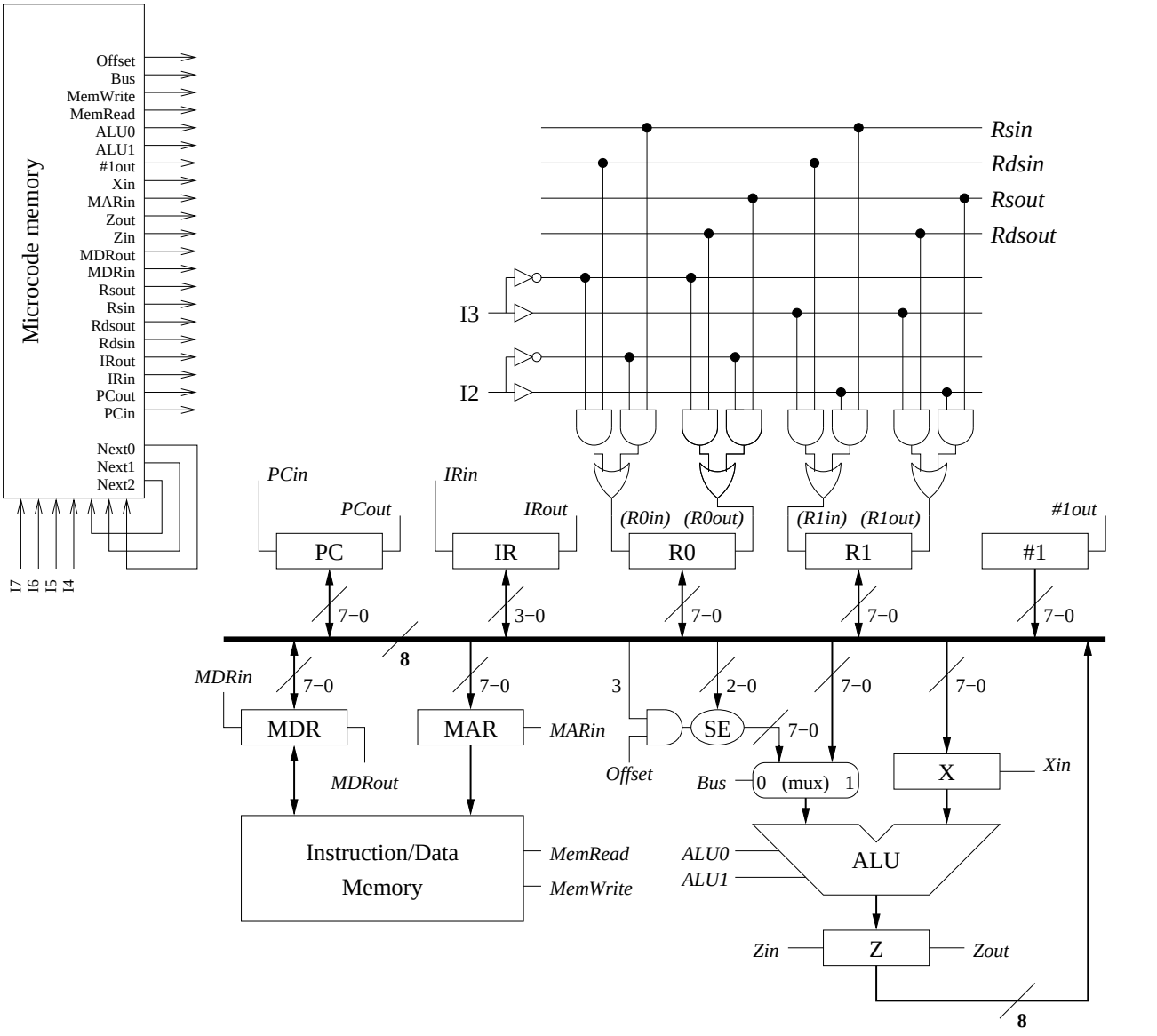


This would not require any new signals, we would just need to rename 4 of them. *R0in, R0out, R1in,* and *R1out* would become *Rdsin, Rdsout, Rsin,* and *Rsout.*

The microcode steps in order to perform an add would look like this:

Step	PCin	PCout	IRin	IRout	Rdsin	Rdsout	Rsin	Rsout	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0						1								1							
1								1			1					1	1			1	
2					1							1									

The machine, including the microcode, would look like this:



Suppose the IR currently contains 0 0 0 1 1 0 0 0, and will contain 1 1 1 1 0 1 1 0 next. Let's see if we can walk through what is happening in the microcode. Here is step 0 of fetch:

Control Memory Address **IIII NNN 7654 210**	Next 2	Next 1	Next 0	PC in	PC out	IR in	IR out	R0 in	R0 out	R1 in	R1 out	MDR in	MDR out	Z in	Z out	MAR in	X in	#1 out	ALU 0	ALU 1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Suppose the IR currently contains 0 0 0 1 1 0 0 0, and will contain 1 1 1 1 0 1 1 0 next. Let's see if we can walk through what is happening in the microcode. Here is step 1 of fetch:

Control Memory Address **IIII NNN** **7654 210**	Next2	Next1	Next0	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Suppose the IR currently contains 0 0 0 1 1 0 0 0, and will contain 1 1 1 1 0 1 1 0 next. Let's see if we can walk through what is happening in the microcode. Here is step 2 of fetch:

Control Memory Address **IIII NNN** **7654 210**	Next2	Next1	Next0	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Suppose the IR currently contains 0 0 0 1 1 0 0 0, and will contain 1 1 1 1 0 1 1 0 next. Let's see if we can walk through what is happening in the microcode. Here is step 3 of fetch:

Control Memory Address **IIII NNN** **7654 210**	Next2	Next1	Next0	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

At this point, we have fetched the new instruction, and IR contains 1 1 1 1 0 1 1 0. We will perform the first step of the jump instruction.

Control Memory Address **IIII NNN 7654 210**	Next2	Next1	Next0	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

At this point, we have fetched the new instruction, and IR contains 1 1 1 1 0 1 1 0. We will perform the second step of the jump instruction.

Control Memory Address **IIII NNN 7654 210**	Next2	Next1	Next0	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

At this point, we have fetched the new instruction, and IR contains 1 1 1 1 0 1 1 0. We will perform the third step of the jump instruction.

Control Memory Address **IIII NNN** **7654 210**	Next2	Next1	Next0	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

At this point, we have finished the jump instruction, and we will now begin fetching the next instruction. This is step 0 of that fetch.

Control Memory Address **IIII NNN 7654 210**	Next 2	Next 1	Next 0	PC in	PC out	IR in	IR out	R0 in	R0 out	R1 in	R1 out	MDR in	MDR out	Z in	Z out	MAR in	X in	#1 out	ALU 0	ALU 1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

At this point, we have finished the jump instruction, and we will now begin fetching the next instruction. This is step 1 of that fetch.

Control Memory Address **IIII NNN 7654 210**	Next 2	Next 1	Next 0	PCin	PCout	IRin	IRout	R0in	R0out	R1in	R1out	MDRin	MDRout	Zin	Zout	MARin	Xin	#1out	ALU0	ALU1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

At this point, we have finished the jump instruction, and we will now begin fetching the next instruction. This is step 2 of that fetch.

Control Memory Address **IIII NNN 7654 210**	Next 2	Next 1	Next 0	PC in	PC out	IR in	IR out	R0 in	R0 out	R1 in	R1 out	MDR in	MDR out	Z in	Z out	MAR in	X in	#1 out	ALU 0	ALU 1	Mread	Mwrite	Bus	Offset
0000 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0000 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0000 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0000 011	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0000 1xx	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0001 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0001 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0001 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0001 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0001 101	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0
0001 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0010 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0010 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0010 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0010 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0010 101	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
0010 110	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0
0010 111	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0011 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0011 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
0011 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0011 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0011 100	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0011 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
0011 110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0001 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1111 000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1111 001	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0
1111 010	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1111 011	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1111 100	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1111 101	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
1111 110	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1111 111	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

At this point, we have finished the jump instruction, and we will now begin fetching the next instruction. This is the final step of the fetch. After this cycle, the IR will contain a new value, and we will go to the appropriate entry in the table based on the top 4 bits in the IR, and repeat the entire process. Forever. Or at least until it crashes. :-)