Consider the following piece of C code (or almost Verilog code):

```
a=getInput(0); // assume this reads a bit from some input wire
b=getInput(1);
c=getInput(2);
o1=a&b | b&c | a&c;
o2=a^b^c;
putOutput(1,o2); // and this places a bit to some output wire
putOutput(0,o1);
```

Reminder `a^b` is xor – equivalently `((a&!b)|(!a&b))`;

1. What operation does this perform?

2. How many 2-input gates (AND, OR, XOR) does it require?

We could write equivalent C broken into primitive operations:

```
a=getInput(0);
b=getInput(1);
c=getInput(2);
t1=a&b;
t2=b&c;
t1=t1|t2;
t2=a&c;
o1=t1|t2;
t1=a^b;
o2=t1^c;
putOutput(1,o2);
putOutput(0,o1);
```

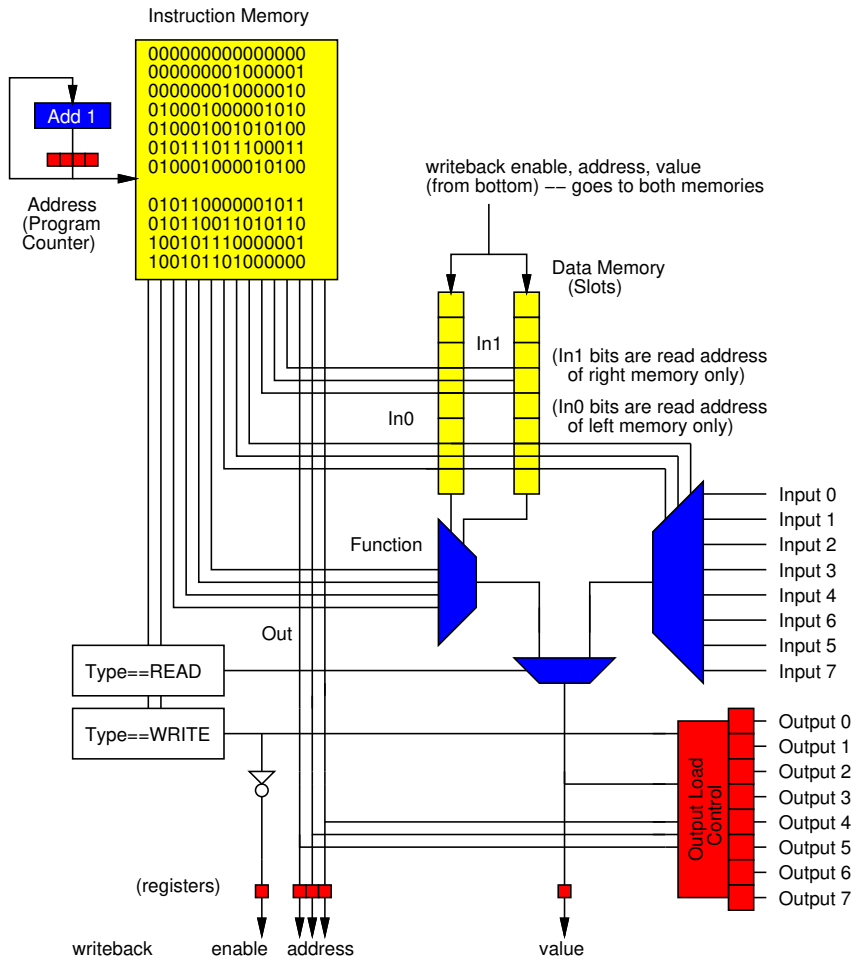

Diagram is relevant, but don't need to understand to complete problem on other side.

(Continue other side)

Assign the variables to slots in a memory:

0	1	2	3	4	5	6	7
a	b	c	t1	t2	o1	o2	

READ=00; GATE=01; WRITE=11;
AND=0001; OR=0111; XOR=0110; NONE=0000; SEL0=0101

Hint: colors show how Instruction Fields correspond to bits in encoding.

3. Complete missing entries in table (one per column);
This is an implementation of the primitive operation version on previous page. Read through entries that are there to see what it's doing. Then complete the missing table entries.

C	Description	Instruction Fields					Bit Enocde
		Type	Function	In0	In1	Out	
a=getInput(0);	read input 0 and put in slot 0	READ	NONE	0	0	0	000000000000000
b=getInput(1);	read input 1 and put in slot 1	READ	NONE	1	0	1	000000001000001
c=getInput(2);	read input 2 and put in slot 2	READ	NONE	2	0	2	000000010000010
`t1=a&b;`	read value in slot 0 and value in slot 1, perform an AND on the values, and store into slot 3	GATE	AND	0	1	3	010001000001011
	read value in slot 1 and value in slot 2, perform an AND on the values, and store into slot 4	GATE	AND	1	2	4	010001001010100
`t1=t1|t2;`	read value in slot 3 and value in slot 4, perform an OR on the values, and store into slot 3	GATE	OR	3	4	3	010111011100011
`t2=a&c;`		GATE	AND	0	2	4	010001000010100
`o1=t1|t2;`	read value in slot 3 and value in slot 4, perform an OR on the values, and store into slot 5	GATE	OR	3	4	5	
`t1=a^b;`	read value in slot 0 and value in slot 1, perform an XOR on the values, and store into slot 3	GATE	XOR	0	1	3	010110000001011
`o2=t1^c;`	read value in slot 3 and value in slot 2, perform an XOR on the values, and store into slot 6						010110011010110
putOutput(1,o2);	read value in slot 6 and write to output 1	WRITE	SEL0	6	0	1	100101110000001
putOutput(0,o1);	read value in slot 5 and write to output 0	WRITE	SEL0	5	0	0	100101101000000