Mealy machines (6.3)

A Mealy machine is one where the outputs depend directly on the inputs. That has significantly more implications than you'd think.

- First of all, it means that the outputs will change soon after the inputs change and won't wait for the next rising edge of the clock. This can be handy (fast response time) and annoying (recall our traffic light from lab—you really don't want the light changing from green to yellow to green again).
- Secondly it means that we can sometimes reduce the number of states needed.
- It also means that the outputs need to show up on edges (arcs) of the state diagram rather than in the states. (Why is that?)

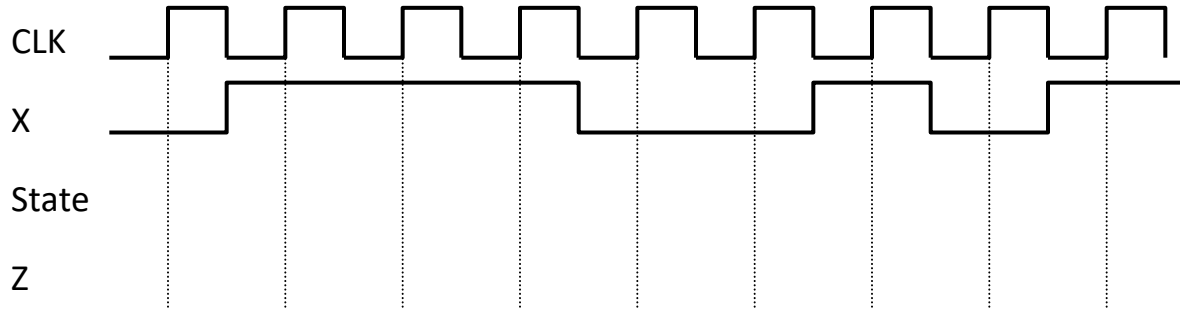
Draw the high-level diagram for a Moore machine (next state, state, OL). What what gets added for a Mealy machine?

Problem:
Using a Moore machine which takes one input X and generates one output Z, design it so that Z goes high iff X has been high for the last two cycles.

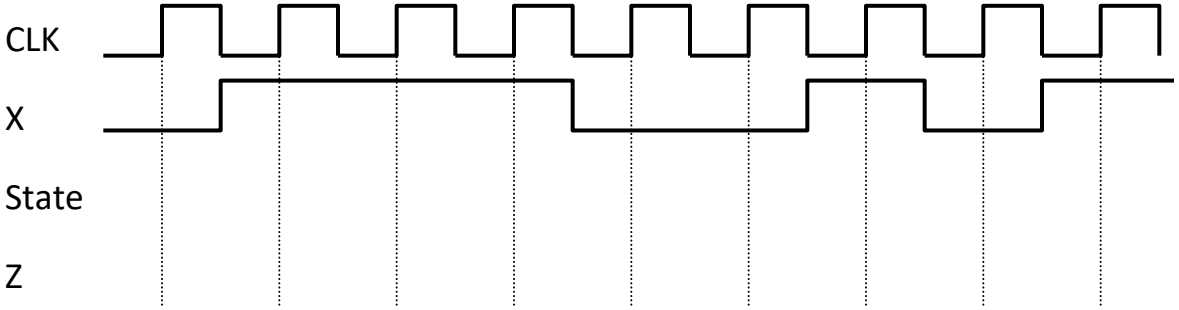
Now solve the same problem with a Mealy machine.

Now, let's look at the timing diagram associated with each machine.

Moore:



Mealy:



And now a processor…

Instruction name	Opcode	Effect
halt	0	PC = PC+4 stop executing instructions
add	1	PC = PC+4 memory[addr0] = memory[addr1] + memory[addr2]
sub	2	PC = PC+4 memory[addr0] = memory[addr1] - memory[addr2]
mult	3	PC = PC+4 memory[addr0] = memory[addr1] * memory[addr2]
div	4	PC = PC+4 memory[addr0] = memory[addr1] / memory[addr2]
cp	5	PC = PC+4 memory[addr0] = memory[addr1]
and	6	PC = PC+4 memory[addr0] = memory[addr1] & memory[addr2]
or	7	PC = PC+4 memory[addr0] = memory[addr1] | memory[addr2]
not	8	PC = PC+4 memory[addr0] = ~memory[addr1]

be	9	if (memory[addr1] == memory[addr2]) {
		  PC = addr0
		} else {
		  PC = PC+4
		}

bne	10	if (memory[addr1] != memory[addr2]) {
		  PC = addr0
		} else {
		  PC = PC+4
		}

blt	11	if (memory[addr1] < memory[addr2]) {
		  PC = addr0
		} else {
		  PC = PC+4
		}
		
		Comparisons take into account the sign of the number. E.g., 16'hffff (-1) is less than 16'h0000 (0).

All instructions are encoded over 4 memory locations. First is the opcode. Next three are addr0, addr1, addr2, addr3.

Short program that computes $x=a^2+b^2$ where x is memory location 100, a is location 101 and b is location 102

```
Mult    101     101     101
Mult    102     102     102
Add     100     101     102
Halt
```

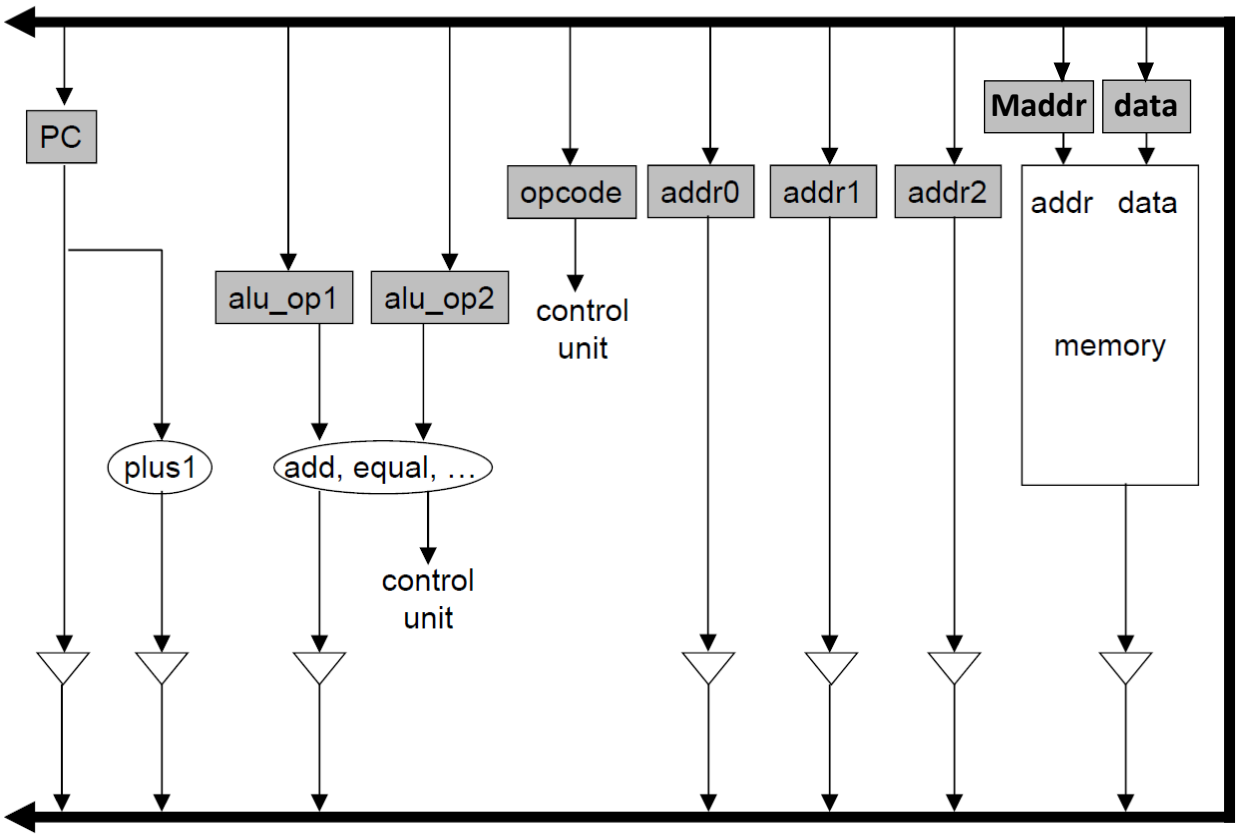
Write a program that computes the average of locations 101, 102, and 103 and puts that average in location 100. You can assume memory location 99 has a "3" in it.

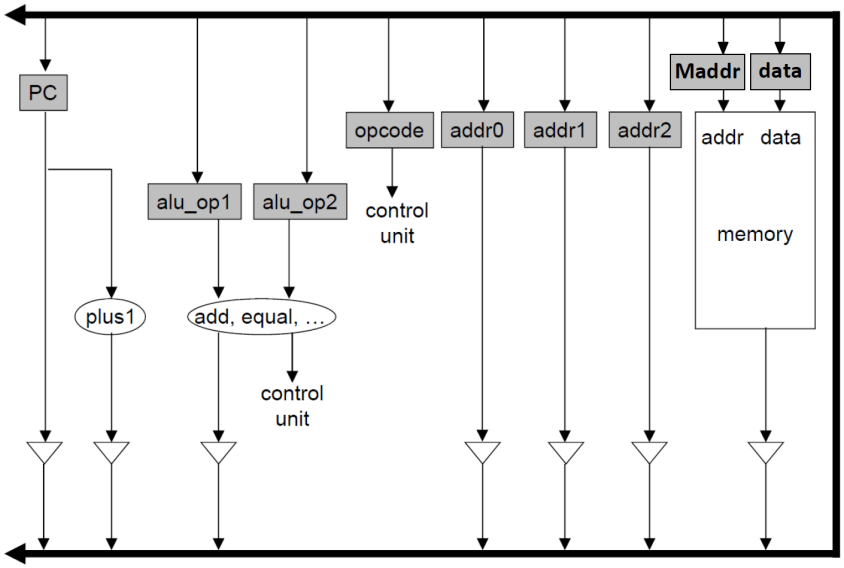
One thing to keep in mind is that each instruction takes up 4 memory locations and we start at location 0. So, for the program below, what memory location does the Halt instruction start at?

What does the program below do?

```
Blt     12      101     102
Cp      100     101
Be      16      0       0
Cp      100     102
Halt
```

Write a program that sums the numbers 1 to 200 and puts that result in location 100. You may assume that location 100 is initially 0 and that location 99 is initially 200.


PC
Maddr
data
opcode
addr0
addr1
addr2
addr
data
alu_op1
alu_op2
control
unit
memory
plus1
add, equal, …
control
unit


PC
Maddr
data
opcode
addr0
addr1
addr2
addr data
alu_op1
alu_op2
control unit
memory
plus1
add, equal, …
control unit