1.

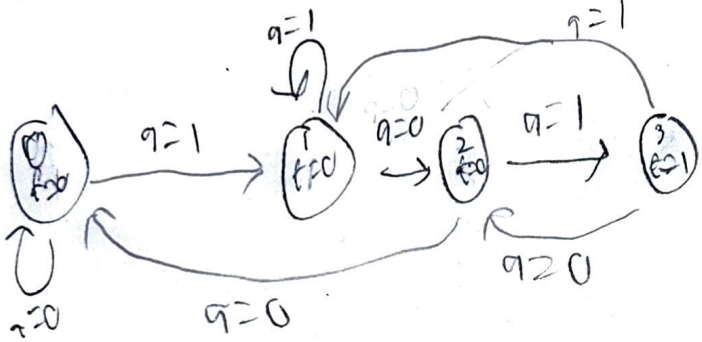
a. it's a moore machine b/c the output is changed at the current state and not in between state changes

b. 1st always block: sets up the reset function and the changing of currState

2nd always block: sets up our moore logic for state-switching

3rd always block: sets up output for each state

c.



detects 101

d. State 0

e. detects 101

2. x86 is a CISC b/c the Add instruction supports many types of operands where as RISC only supports registers ^ and maybe sometimes constants. Professor Yang also stated Intel was CISC and Intel uses x86.

b. You would first have to load the 32-bit ~~#~~ from memory to a register. Then, you would use the add instruction.

Ex:

```
ldr r4, =num2
add r32,r4 // assume r32 has its value already

num2 dcd 0x12345678
```

I used ARM assembly for my example instead of RISC-V, but the point I'm making still stands.

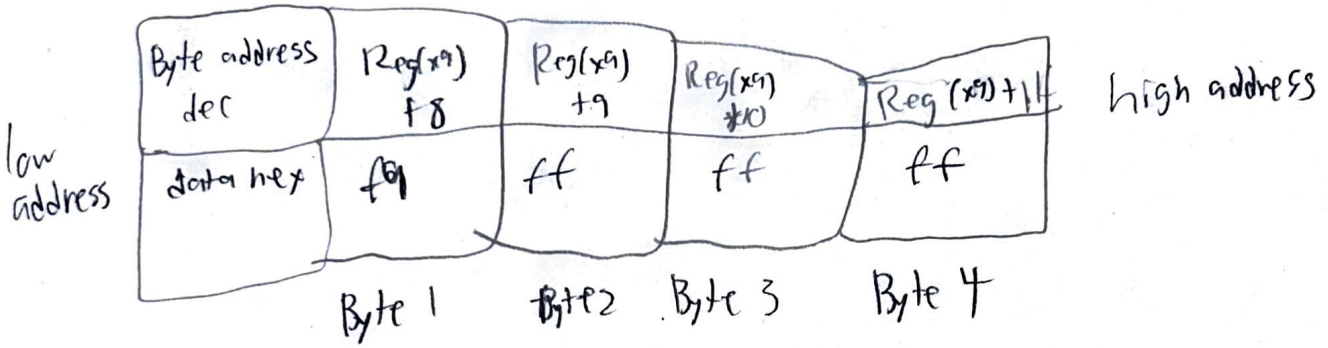
3.

-7 for 32 bit

7 = 0x00000007 → 0111 →

-7 = 0xfffffff9 → 1001

0xfffffff9
B4 B3 B2 B1



b.

Add x19, x19, x19 // 9*2

LW x7, 8(x9) // -7 loaded

Add x20, x20, x7 // 14 -7

Add x18, x19, x20 // 9*2 + (14-7) = f

For this to work, I'm going to assume I can load -7 from x9 given that this is still the same problem