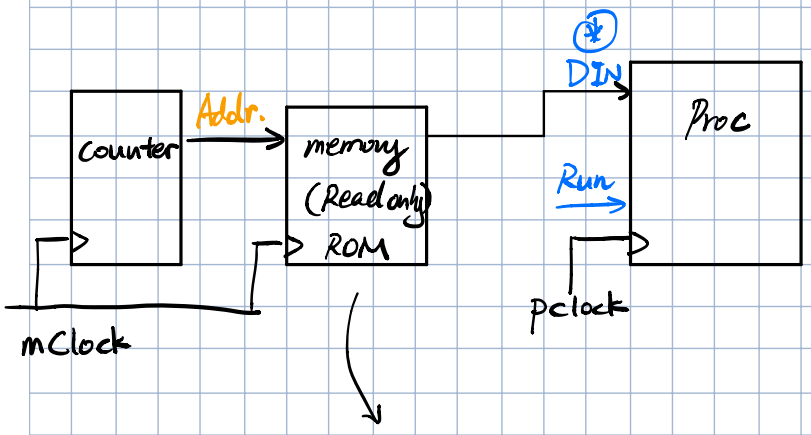
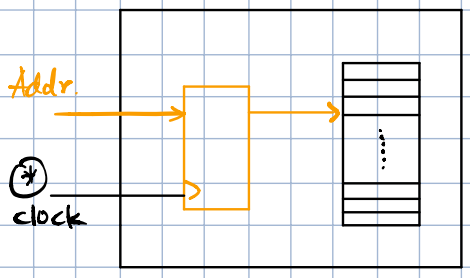
Add an external counter & memory (Lab 1, part 2)



* usually DIN is used for instructions. (IIIXXXYYY) But for "mvi", DIN is also used for the data.



* the memory that you created. using Quartus has a register to hold Addr. Thus, a clock edge is needed in the memory before it provides data.

Enhanced Processor (lab 2)

- change all registers to 16-bits.
- add an interface btw the processor and memory.
- add 2 instructions for reads/write of memory & 2 instructions for loop control.
- change R7 into a program counter (pc) register. The PC register always holds the address in memory of the next instruction to be executed.

New Instructions

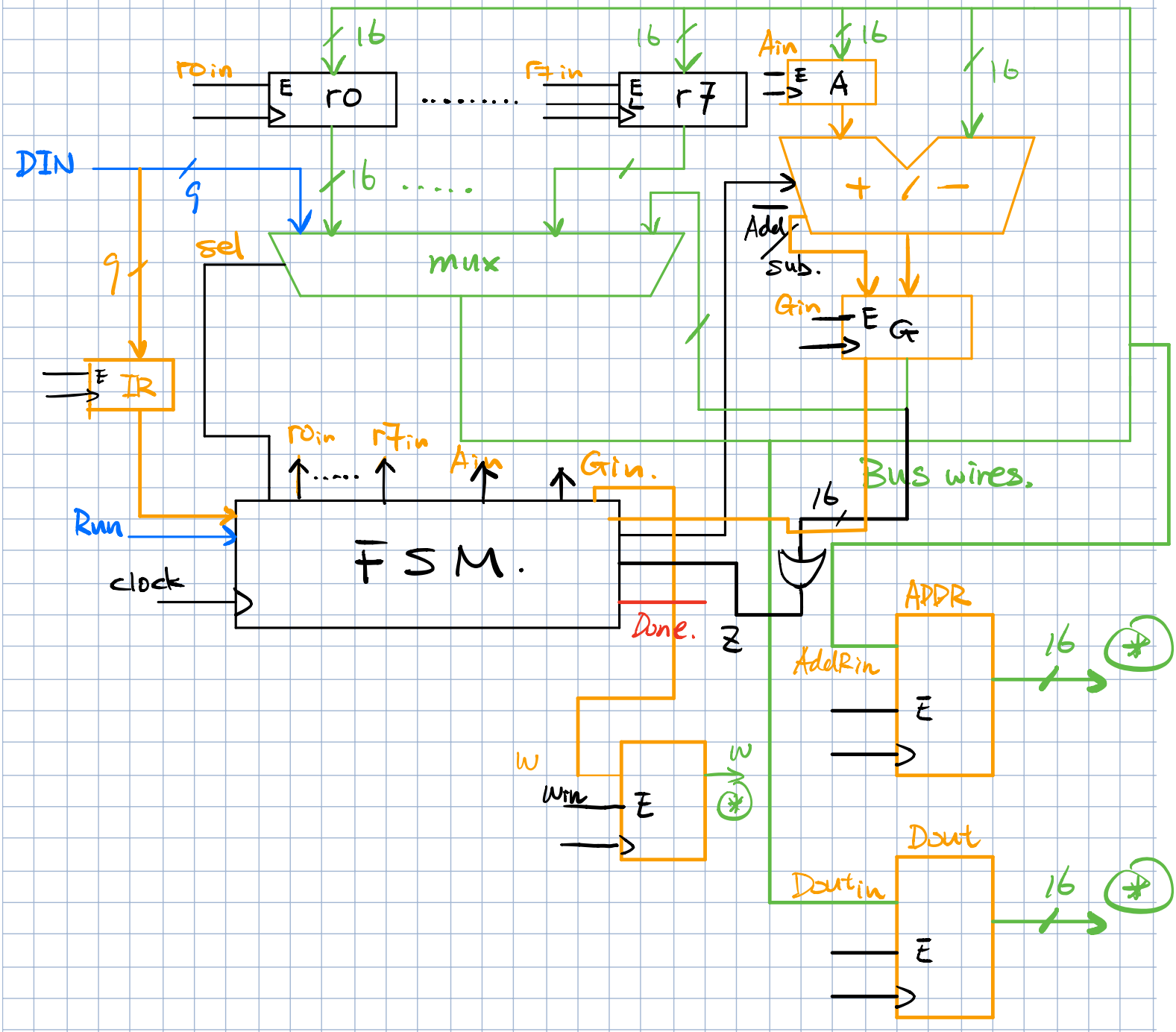
`ld rX, [rY]` rX ← contents of memory using rY for the Address.

`st rX, [rY]` store rX into memory at address in rY

mvnz rX, rY rX ← rY iff register G != 0

mvnc rX, rY rX ← rY iff last add/sub did not produce a carry-out.

Modified Proc.



Connecting the processor to memory & Input/output devices

- Assume that the memory contains 256 words of

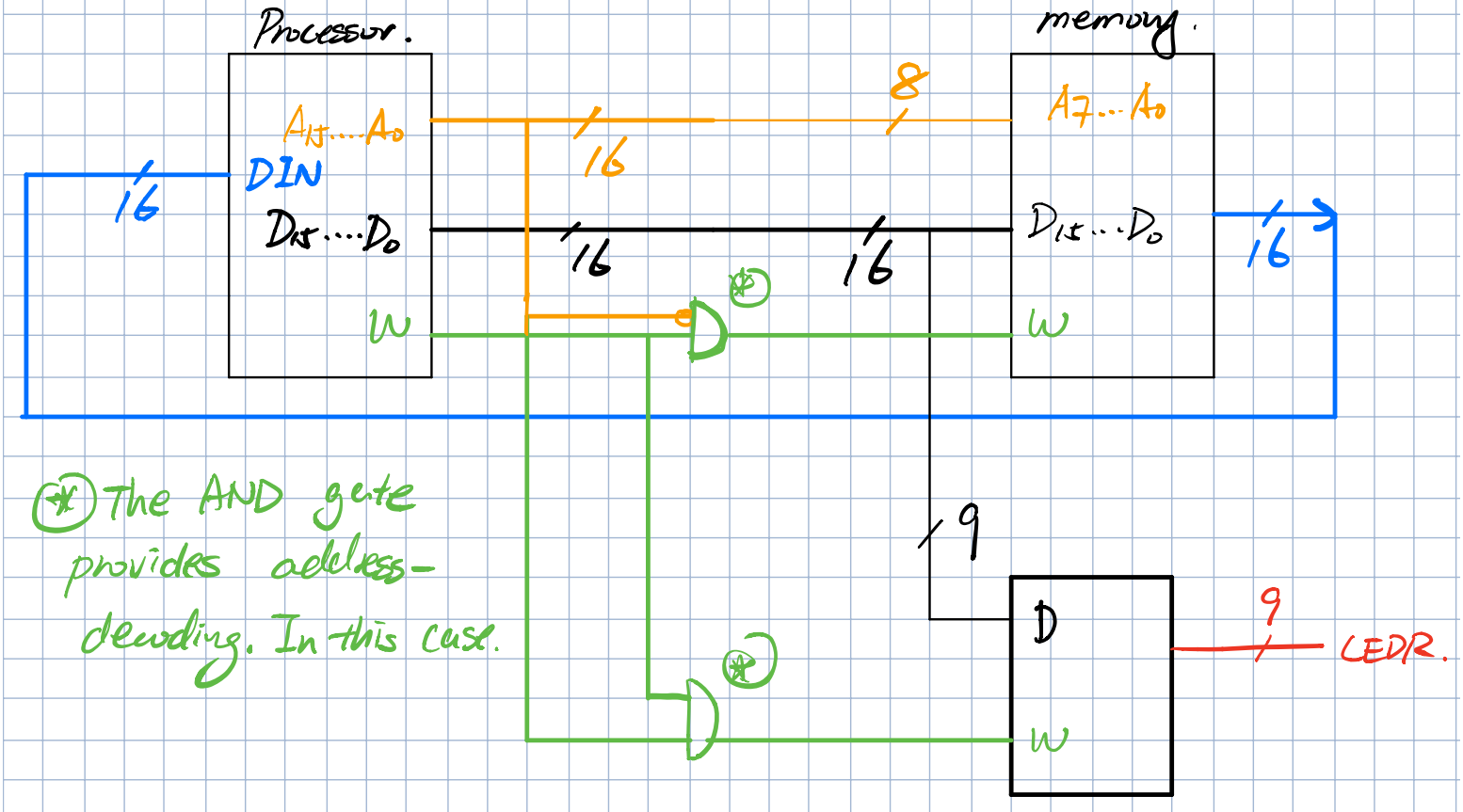
data (instructions)

∴ the memory rows have addresses from 0 to 255.

In binary, 0000 0000 to 1111 1111

∴ the memory has 8 address input. $A_7 \dots A_0$

- Also assume there's an output port connected to some LEDs



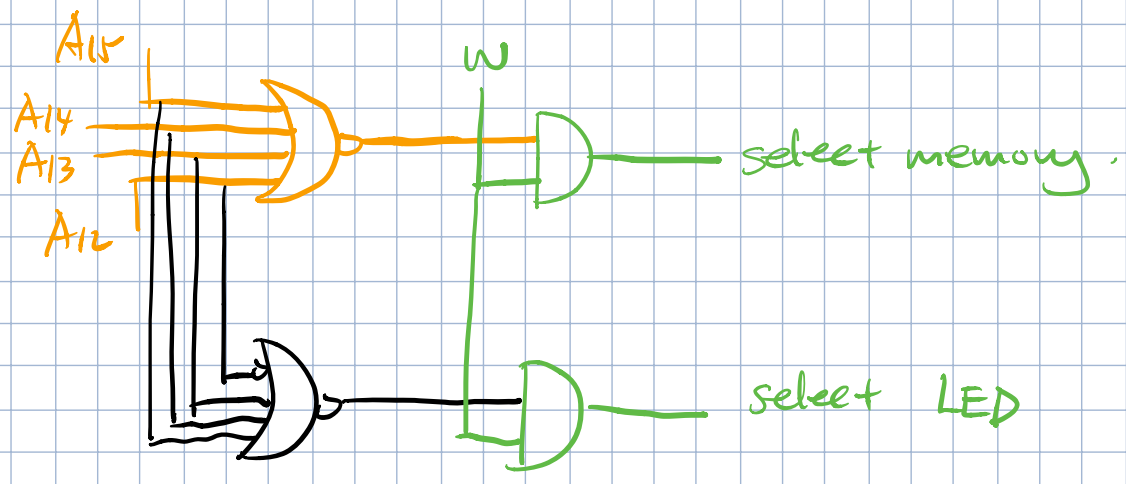
* The AND gate provides address-decoding. In this case.

- memory is selected by any address w/ $A_{12}=0$

- LED output port is selected by any address w/ $A_{12}=1$

- Using only A_{12} for address-decoding is not very flexible.

In lab 2 we use $A_{11} \dots A_{12}$ for address decoding.


A15
A14
A13
A12
W
select memory.
select LED