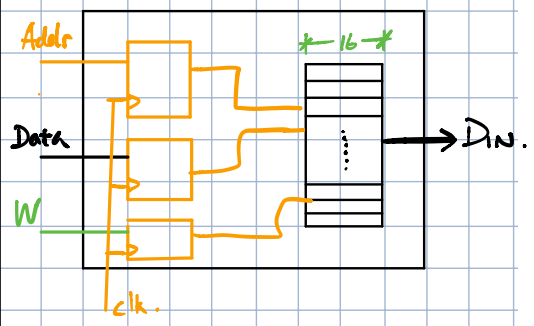
- In processor FSM, new timing signals are needed b/c of the program counter (pc, aka r7 in the diagram), and due to the memory interface. The processor always fetches its next instruction using pc

	T_0	T_1	T_2	T_3	T_4	T_5
mv	sel = pc ADDRin, PCincrement. read memory at address in pc, and increament pc	sel=pc, ADDRin read memory at address in pc, in case the instru. turns out to be mvi	IRin	same as before.		
mvi			IRin			
add	...	...	IRin			
sub	...		IRin.			
ld	...	...	IRin	T3: sel=rY, ADDRin	T4: // don't do anything, wait for a clock cycle -	T5: sel=DIN, rXin, Done.
st	...	...	IRin	sel=rY, ADDRin	sel=rX, Doutin, W	Done.



Example Assembly Language Program.

- Assume LED output port that is mapped to address 0X1000, and an input port connected to SW switches mapped to address 0X3000.

- write a program loop that continuously display the SW value

on the LEDs.

```
.define   LED_ADDRESS   0x1000
.define   SW_ADDRESS    0x3000

        mvi   r3, #LED_ADDRESS.
        mvi   r4, #SW_ADDRESS.
MAIN:   ld    r0, [r4]
        st    r0, [r3]
        mvi   r7, #MAIN.
```

loop back.

Address	0000 000 III XXX YYY	
0	0000000 001 011 000	→ ignored for mvi
1	0001 0000 0000 0000	) mvi r3, 0x1000
2	0000 0000 001 100 000	) mvi r4, 0x3000
3	0011 0000 0000 0000	
"MAIN" 4	0000 0001 0000 0100	) ld r0, [r4]
5	0000 0010 1000 0011	) st r0, [r3]
6	0000 0000 0111 1000	) mvi r7, #MAIN.
7	0000 0000 0000 0100	