- The assembly program that you write is translated into machine code by "hand" (lab1). for automatically using an assembler. tool (lab2)

- The processor fetches (reads) one machine code instruction at a time into DIN from memory

- The instruction is held in the IR register, The FSM controls Sel, R0in, R1in, ... Ain, Gin, ... to execute the instruction in IR over multiple clock cycle

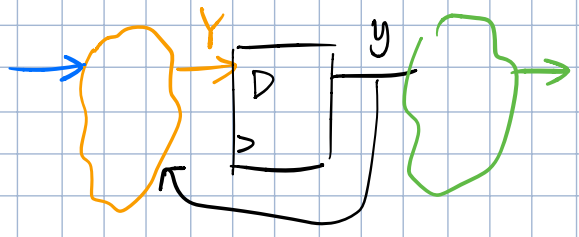
	T_0	T_1	T_2	T_3
mv	IRin	Sel=rY rXin Done		
mvi	IRin	sel=DIN rXin. Done		
add	IRin	sel=rY, Ain	sel=rX Gin Add/sub=0	sel=G rXin Done
sub	IRin	sel=rY, Ain	sel=rX Gin Add/sub=1	sel=G rXin Done

(*) For mvi, the immediate data has to be on DIN in clock cycle T1

Verilog:

```
module proc (DIN, clock, Resetn, Run);
  ...
  parameter T0 = 3'b000, T1 = 3'b001, ...;
(省略) //declare signals for IRin, R0in, R1in, ... Ain, Gin, Sel...
  wire [1:9] = IR;
  reg [2:0] y, Y; //FSM
  parameter mv = 3'b000, mvi = 3'b001, ...;
```

```
reg IR_reg (DIN, IRin, clock, IR); //instantiate the IR register.
```




```
// FSM state table
always @ (*)
    case(y)
        T0: if (!Run) Y = T0;
            else Y = T1;
        T1: if(Done) Y = T0;
            else Y = T2;
        T2: Y = T3;
        T3: Y = T0;
        default: Y = 3'bxxx;
    endcase

// FSM FFs
always @(posedge clock)
    if (! Resetn) y <= T0;
    else y <= Y;

// FSM outputs.
always @ (*)
    begin
        IRin = 0, R0in = 0, R1in = 0 .... Ain = 0, Gin = 0, ... //defaults.
        case (y)
```

```
T0: IRin=1;
T1: case (IR[1:3])
    mv: begin
        sel=rY, rXin=1, Done=1;
        end
    mvi:
        ⋮
T2: ⋮
T3: ⋮

end.
// instantiate registers r0, r1, r2 ...
// define adder, subtractor module...
// define bus wire multiplexer.

endmodule.
```