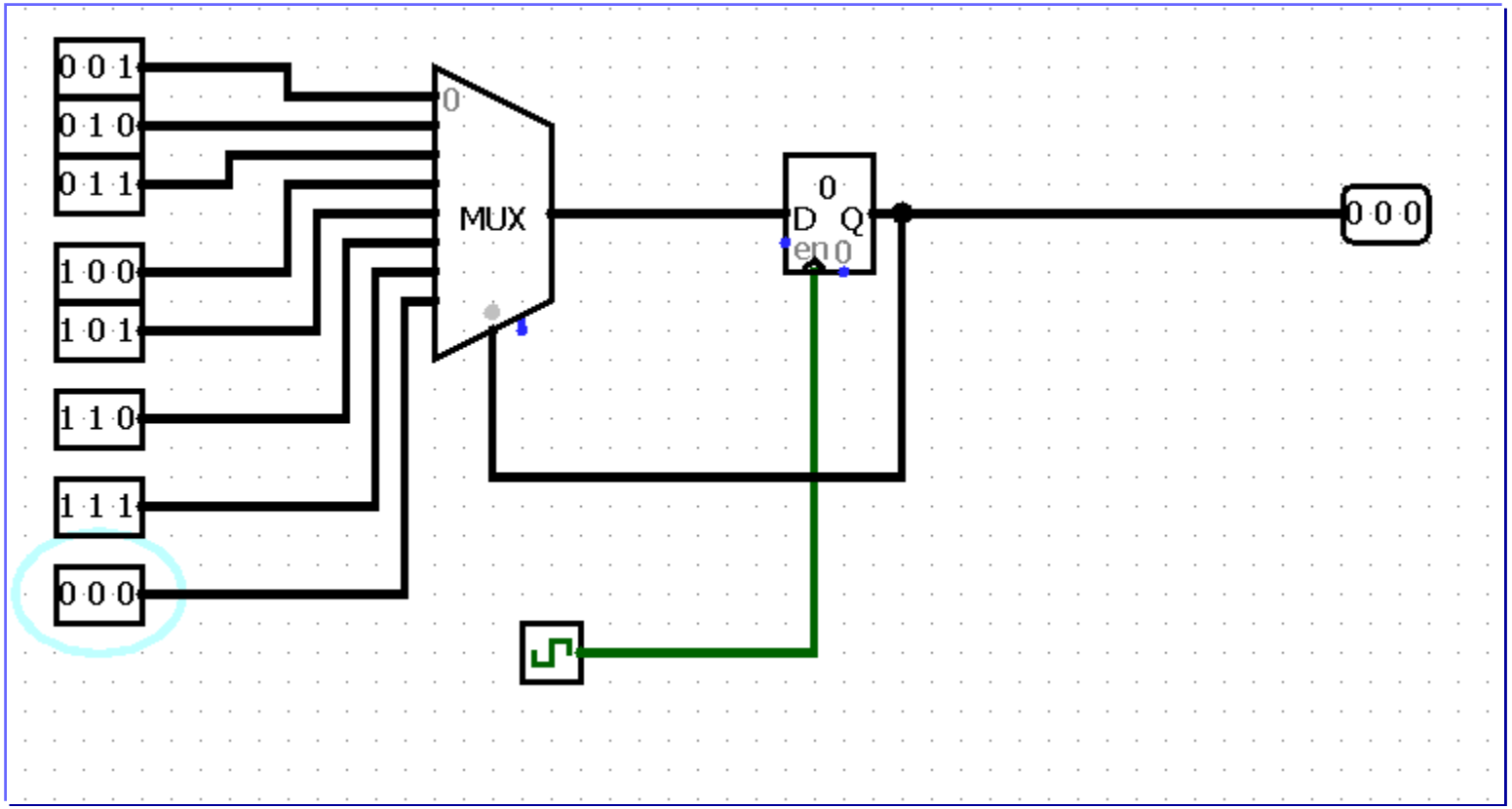
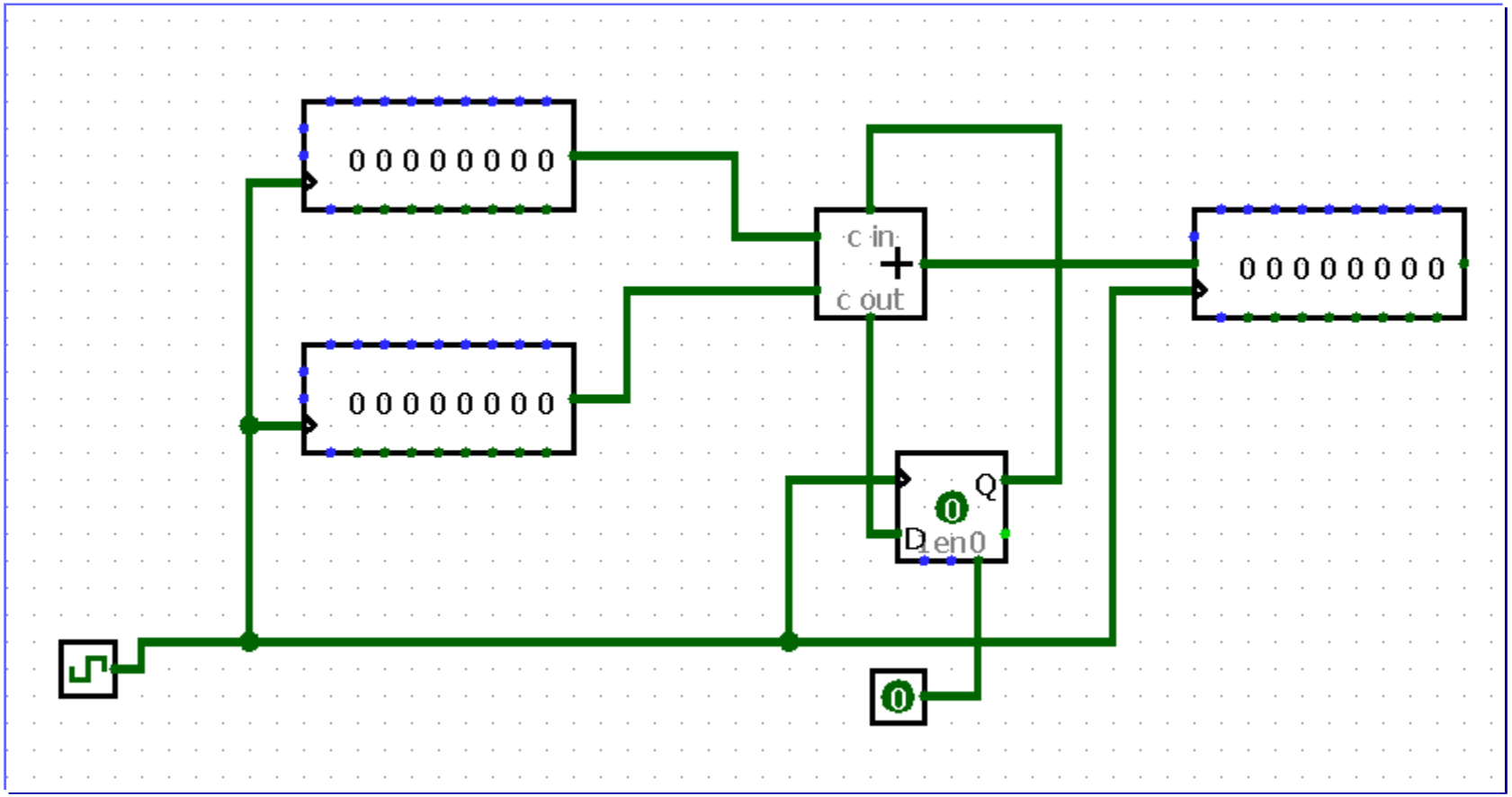
Finite-State Machines (FSMs) and Controllers

FSM design -examples



FSM design -examples



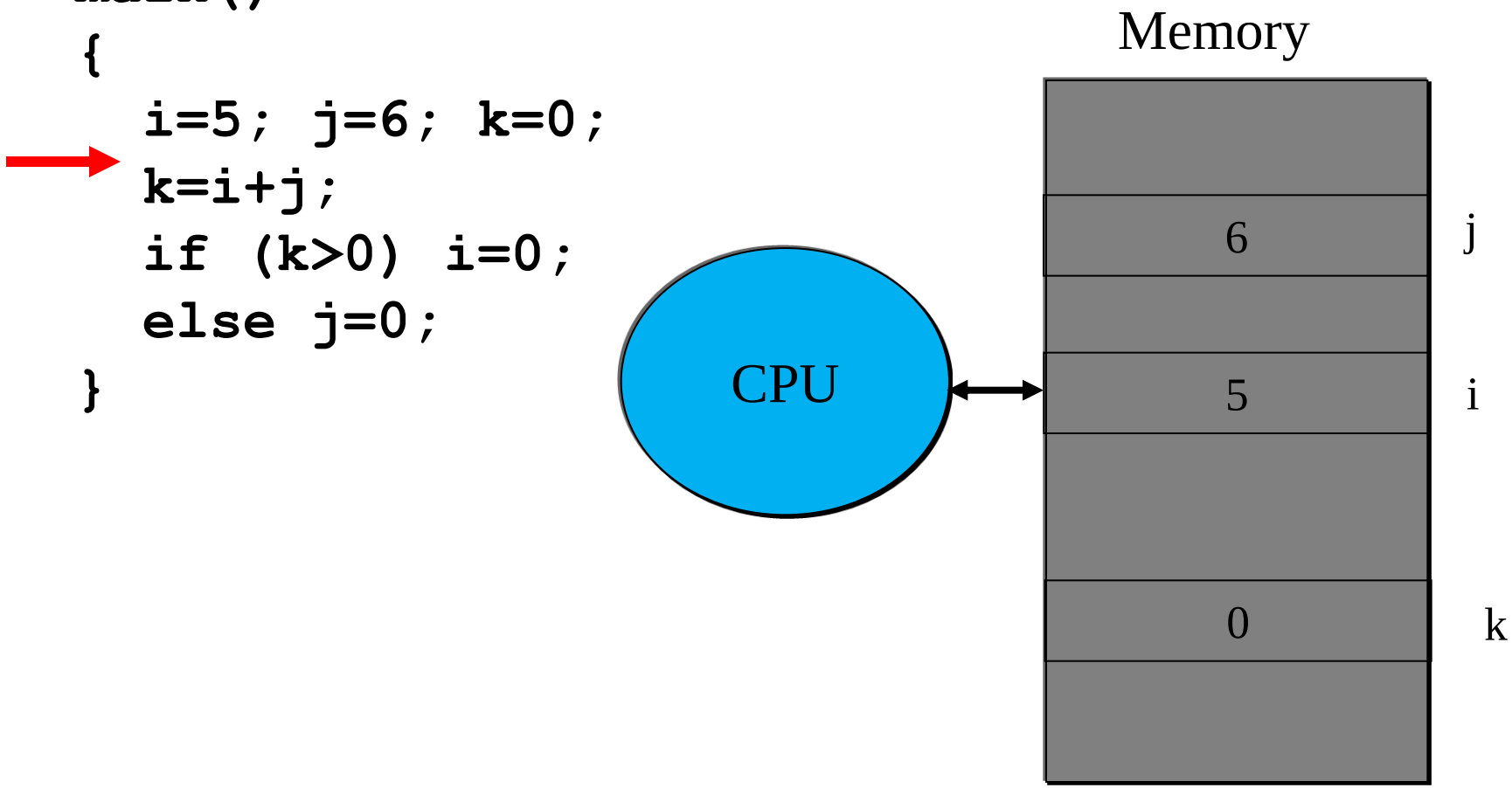
Consider the Program

```
main()
{
   i=5; j=6; k=0;
   k=i+j;
   if (k>0) i=0;
   else j=0;
}
```

- Can this program be modeled with a FSM?

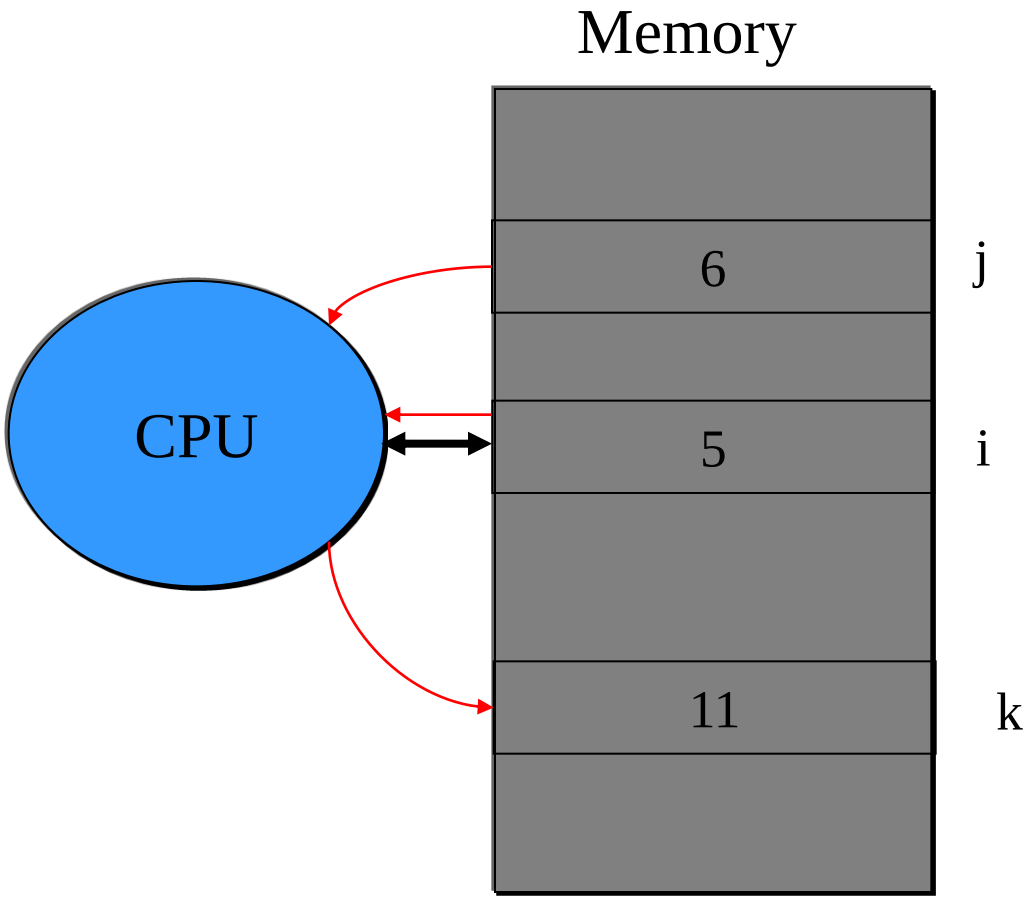
Initial State

```
main()
{
   i=5; j=6; k=0;
   k=i+j;
   if (k>0) i=0;
   else j=0;
}
```



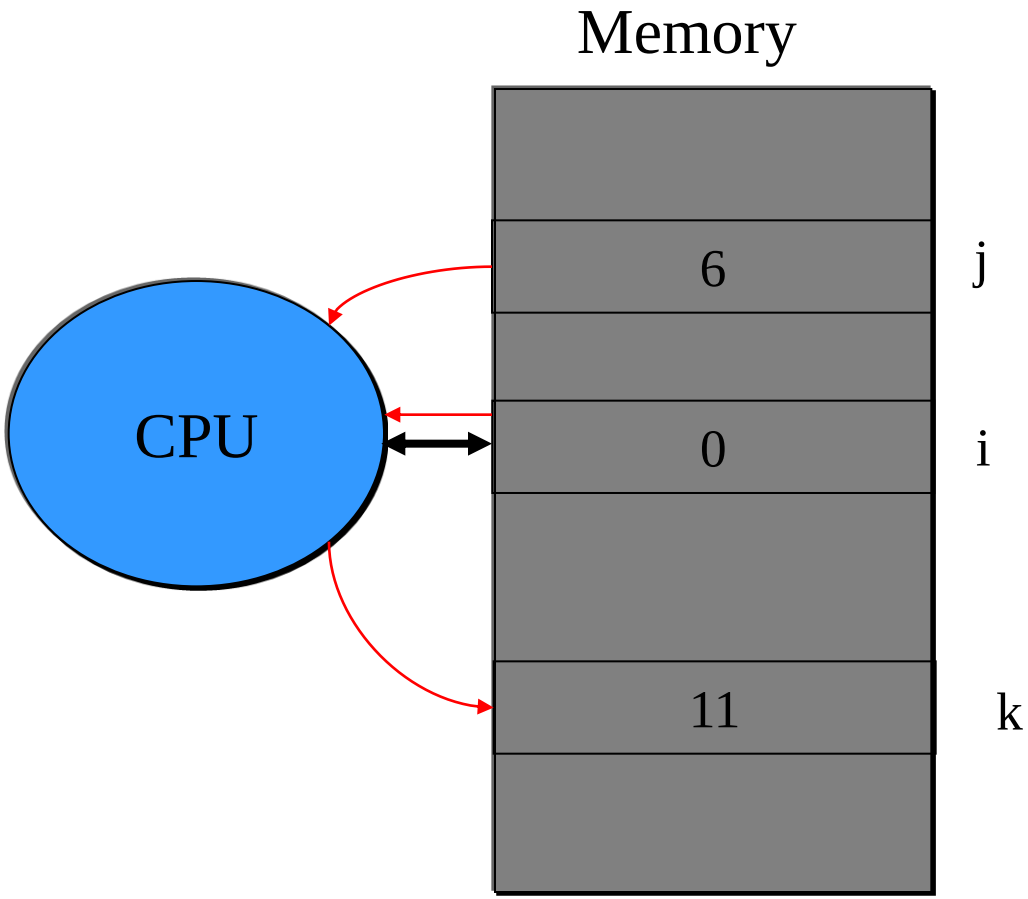
State 1

```
main()
{
   i=5; j=6; k=0;
   k=i+j;
   if (k>0) i=0;
   else j=0;
}
```



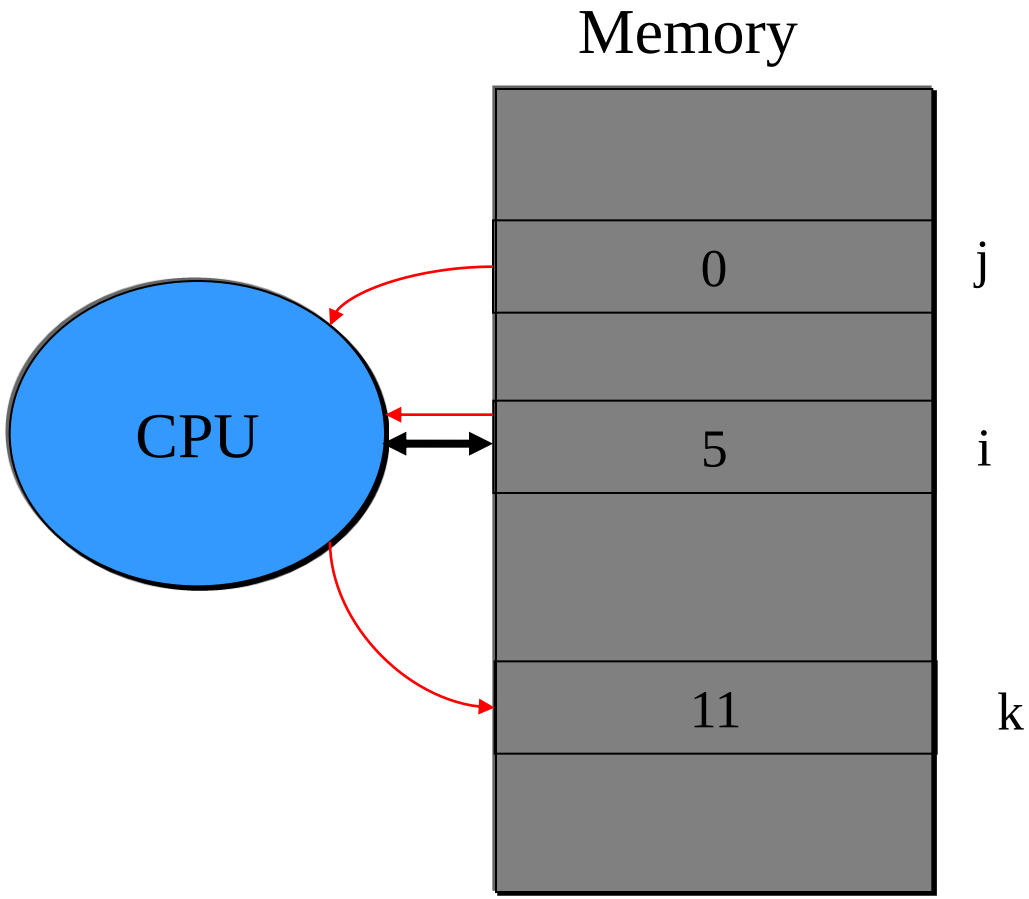
State 2

```
main()
{
   i=5; j=6; k=0;
   k=i+j;
   if (k>0) i=0;
   else j=0;
}
```

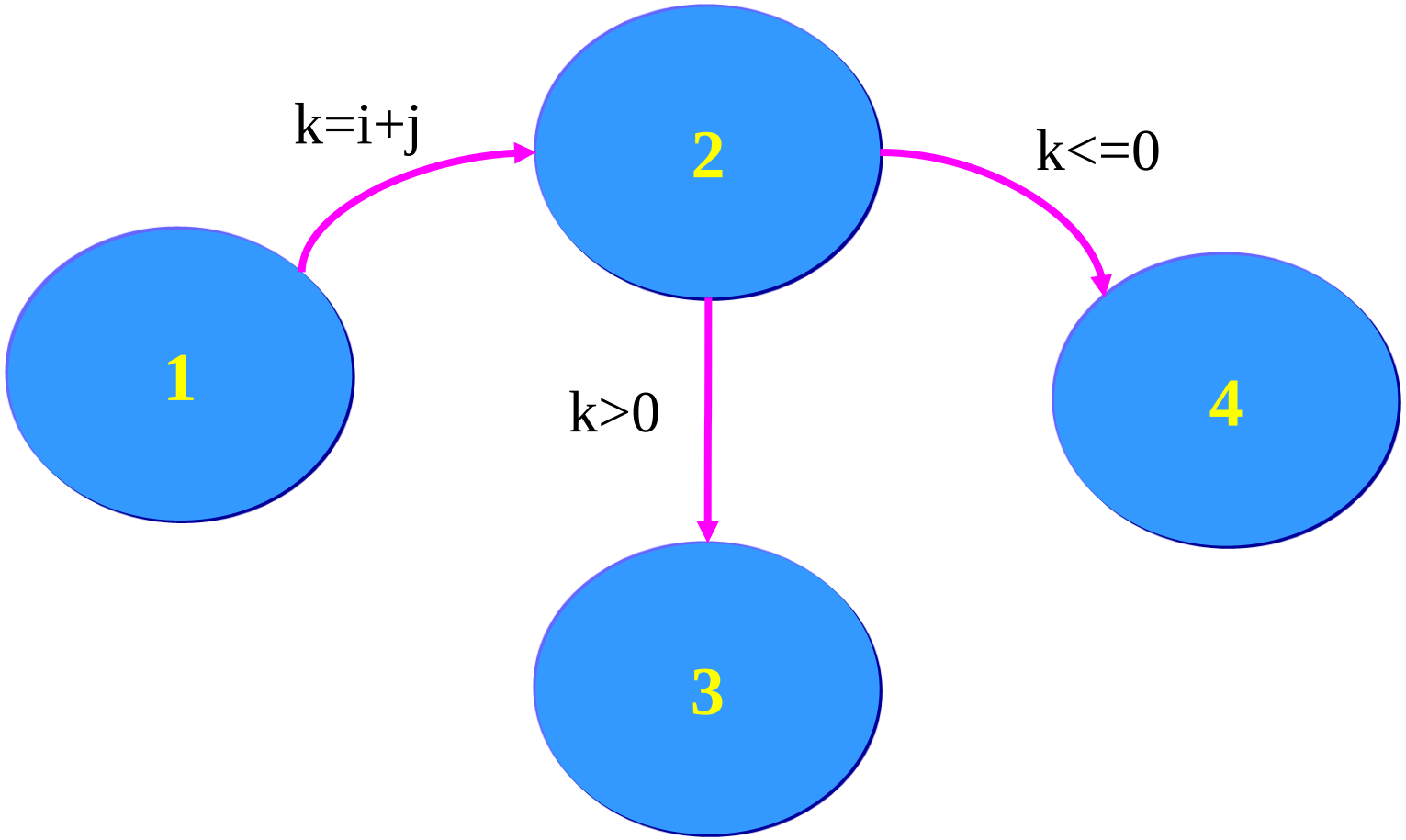


State 3

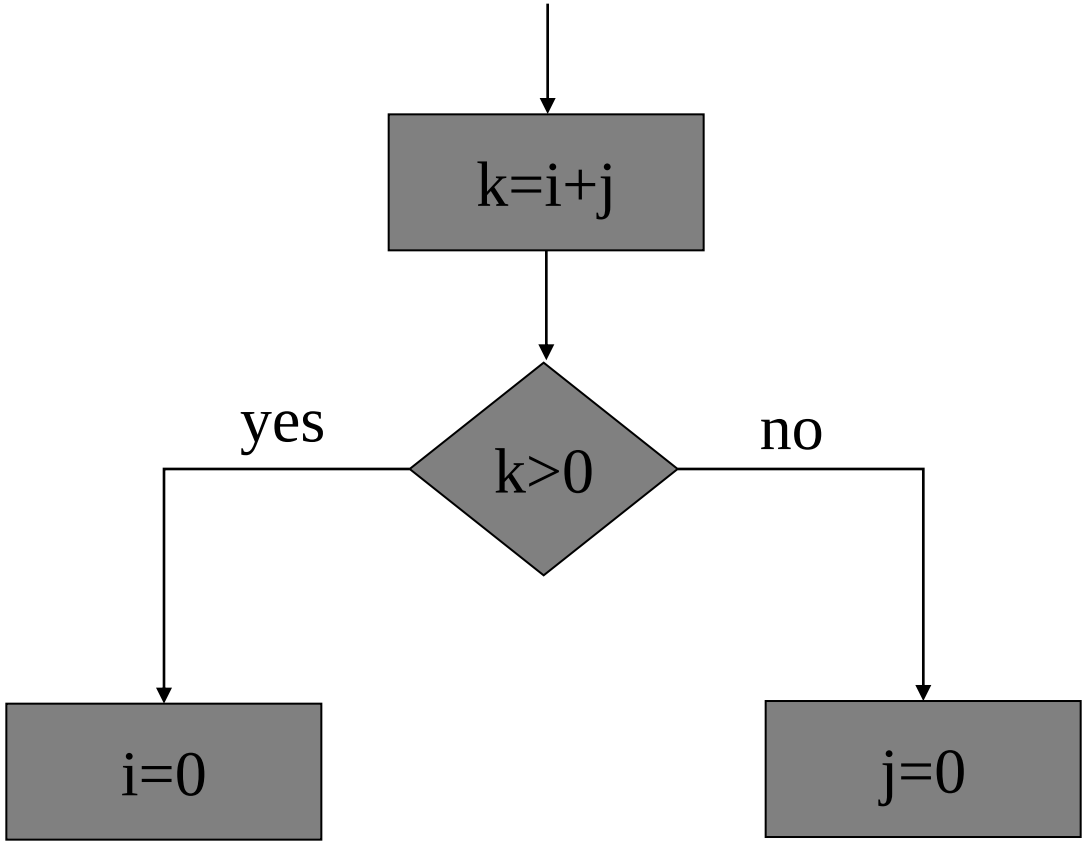
```
main()
{
   i=5; j=6; k=0;
   k=i+j;
   if (k>0) i=0;
   else j=0;
}
```



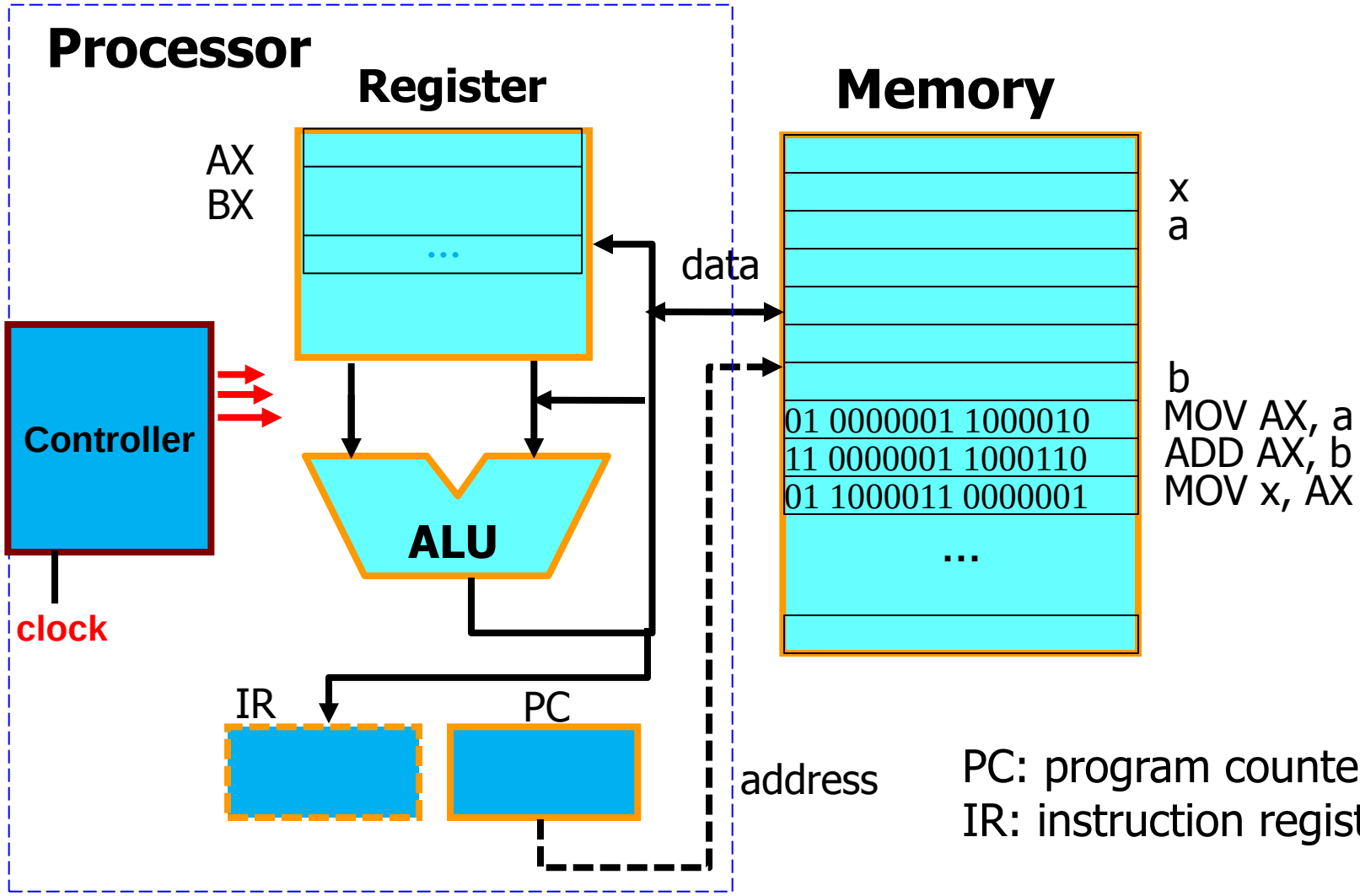
FSM Representing the Program



Compare with Flow Chart

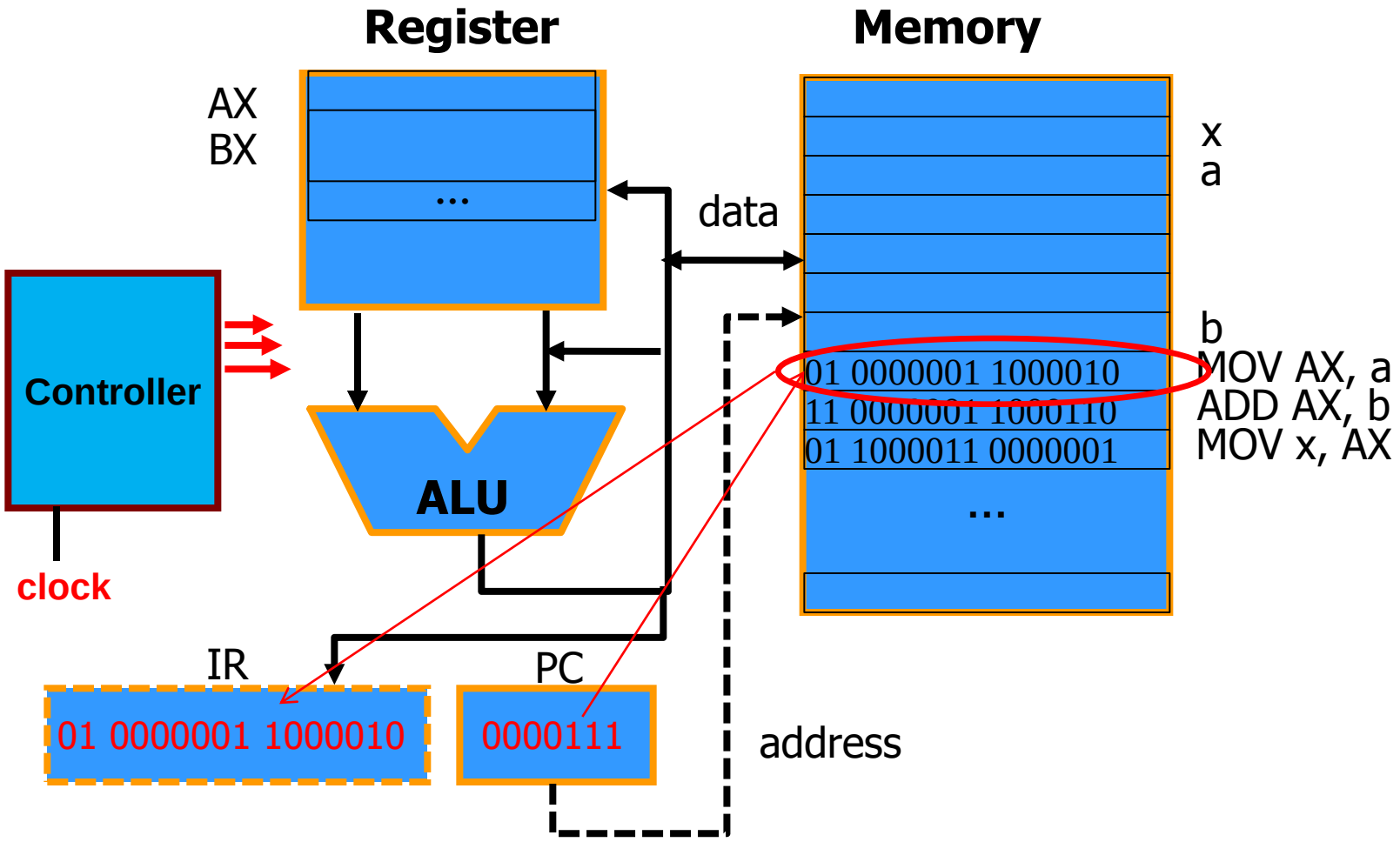


FSM

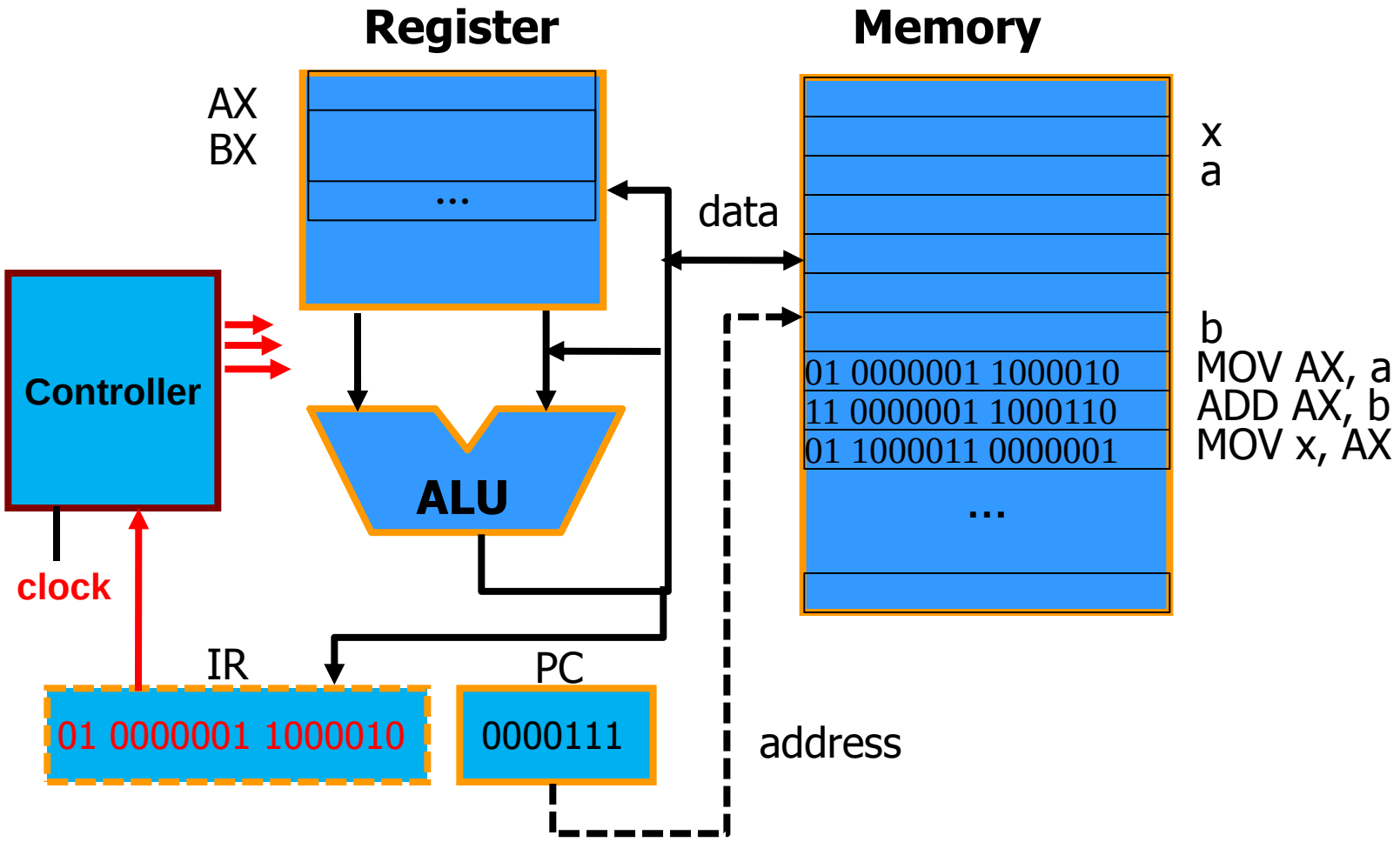


PC: program counter
IR: instruction register

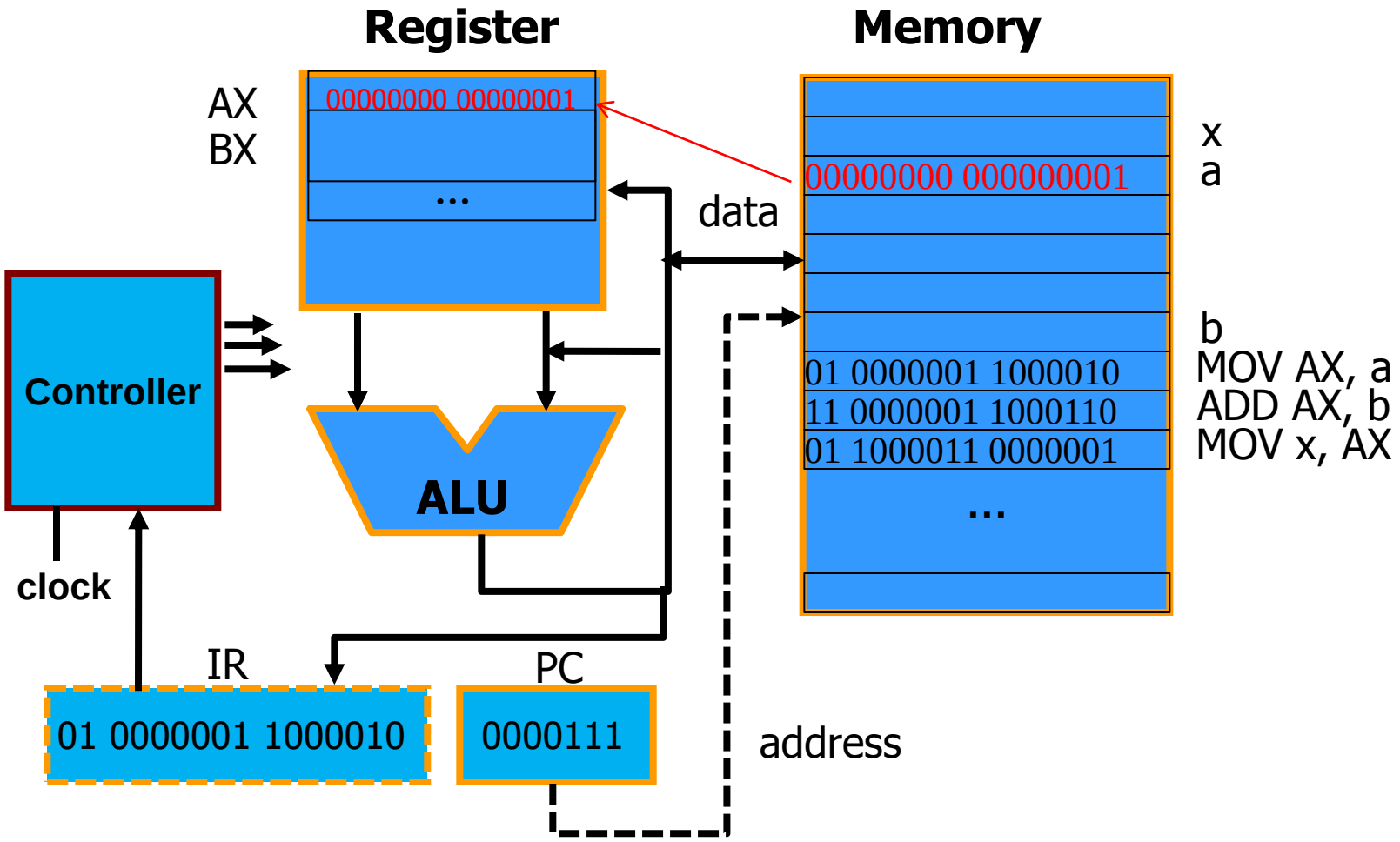
Step 1: Fetch (MOV AX, a)



Step 2: Decode (MOV AX,a)



Step 3: Execute (MOV AX,a)



Concept of the State Machine

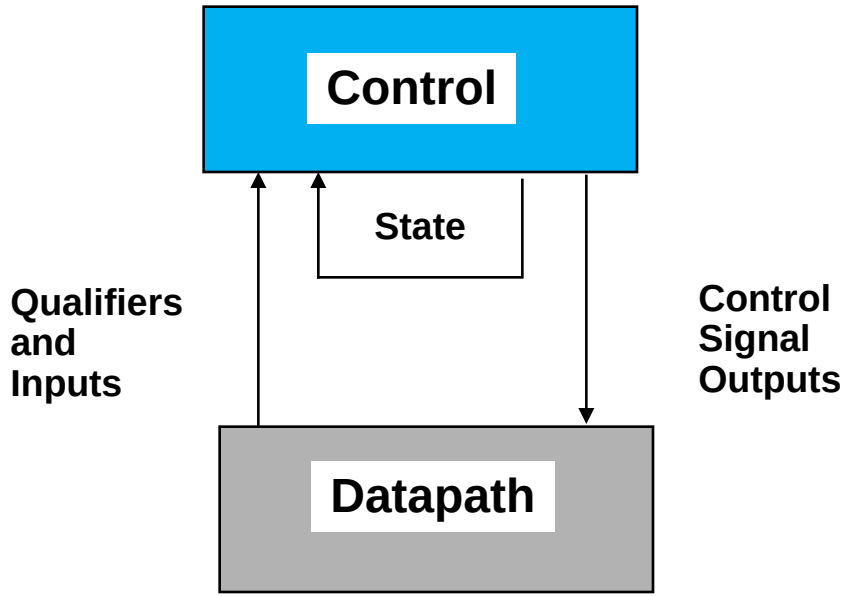
Computer Hardware = Datapath + Control

Registers
Combinational Functional Units (e.g., ALU)
Busses

Qualifiers

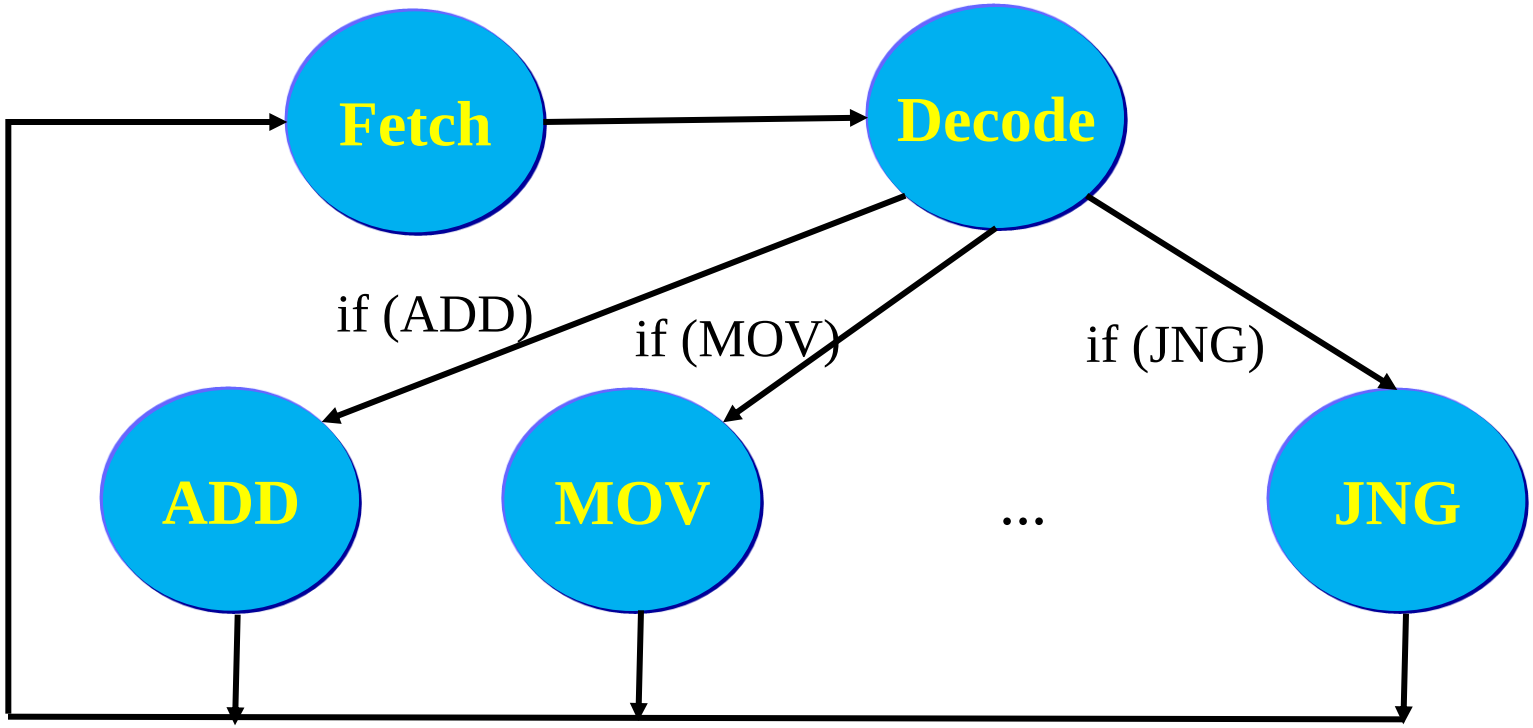
Control

FSM generating sequences of control signals
Instructs datapath what to do next



FSM of the Computer

- For this highly simplified computer, the controller can be described by a FSM



Each state will generate certain control signals to control the datapath