Finite State Machines (FSMs) and RAMs and inner workings of CPUs

3/27/2008

COS 116

Instructor: Sanjeev Arora

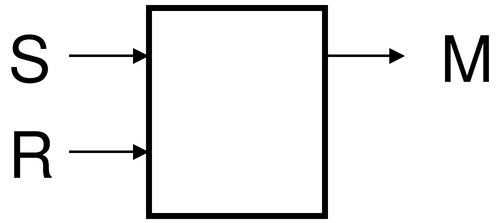
Recap

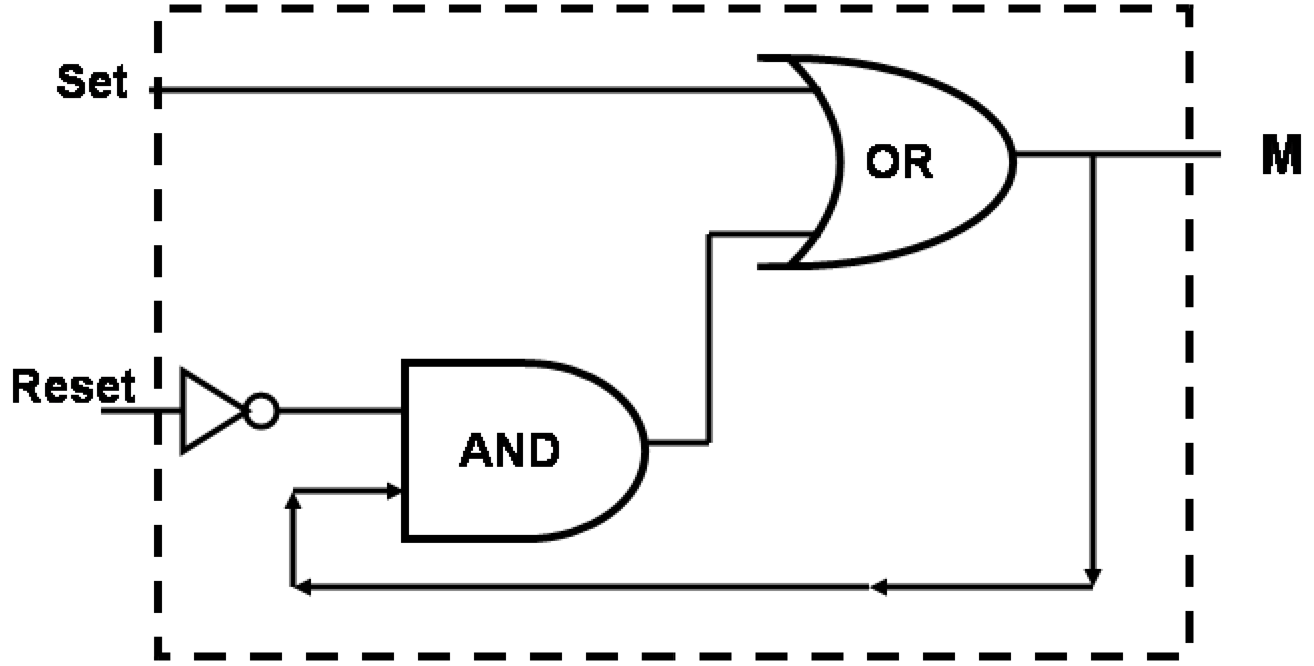
- Combinational logic circuits: no cycles, hence no “memory”

- Sequential circuits: cycles allowed; can have memory as well as “undefined”/ambiguous behavior

- Clocked sequential circuits: Contain D flip flops whose “Write” input is controlled by a clock signal

R-S Flip-Flop (corrected slide)



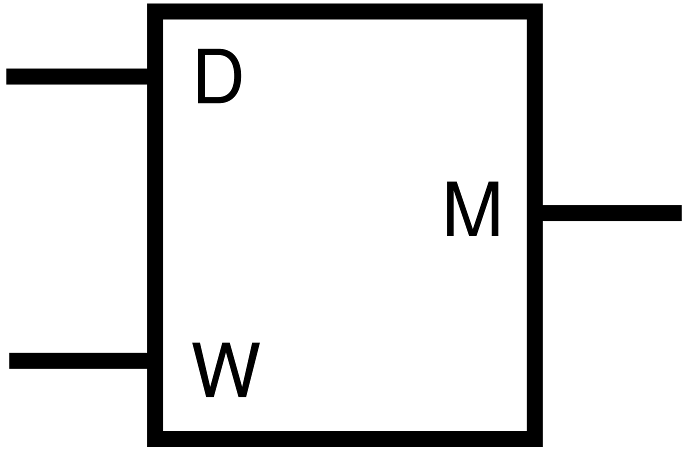


- M becomes 1 if Set is turned on
- M becomes 0 if Reset is turned on
- Otherwise (if both are 0), M just remembers its value

Forbidden to turn off both Set and Reset simultaneously (value is "ambiguous")

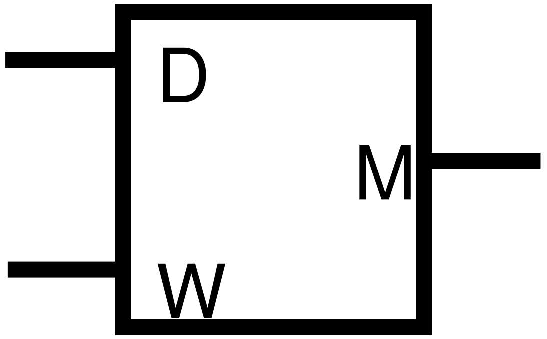
Recap: D Flip Flop

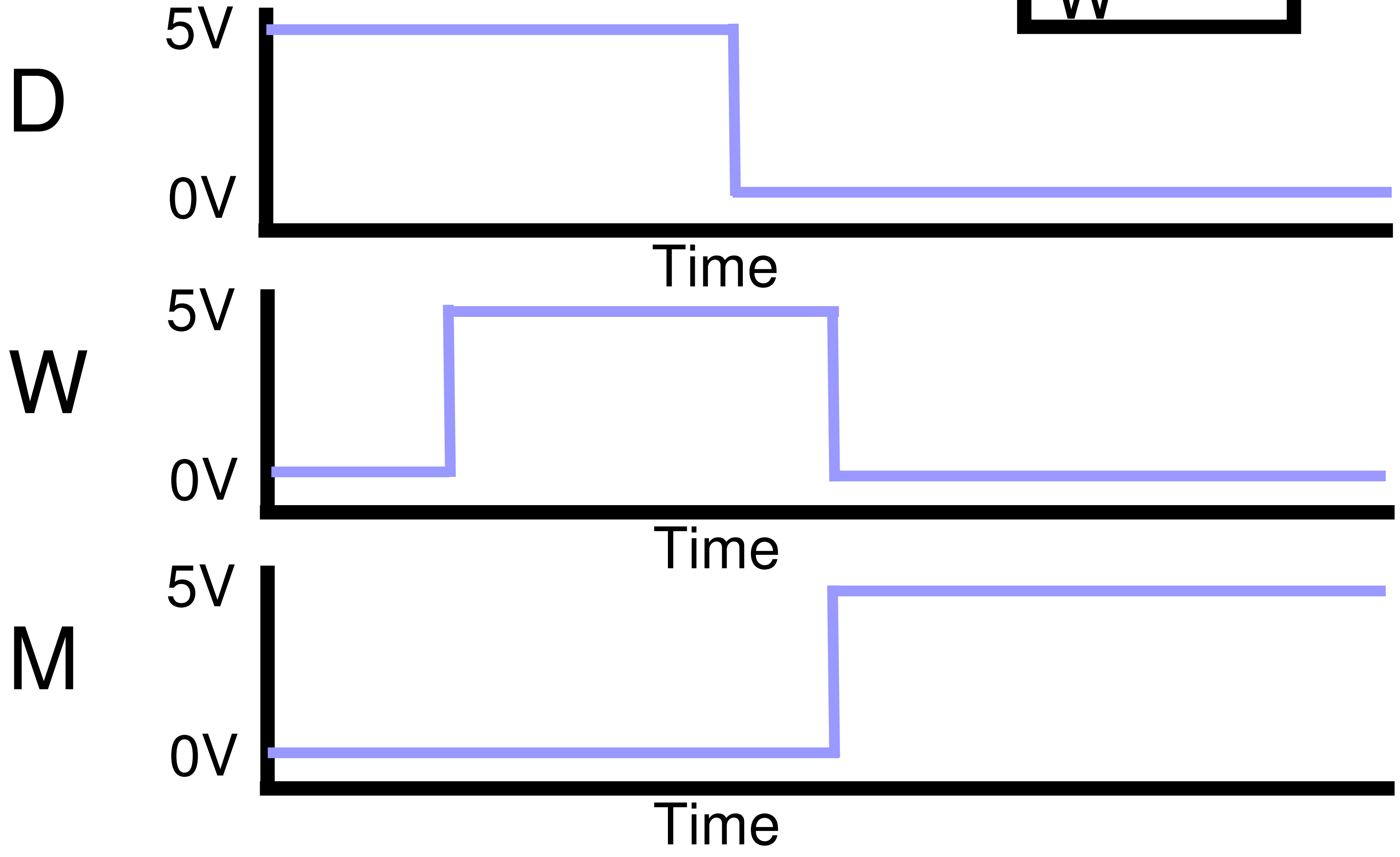
Basic Memory Block – stores 1 bit.



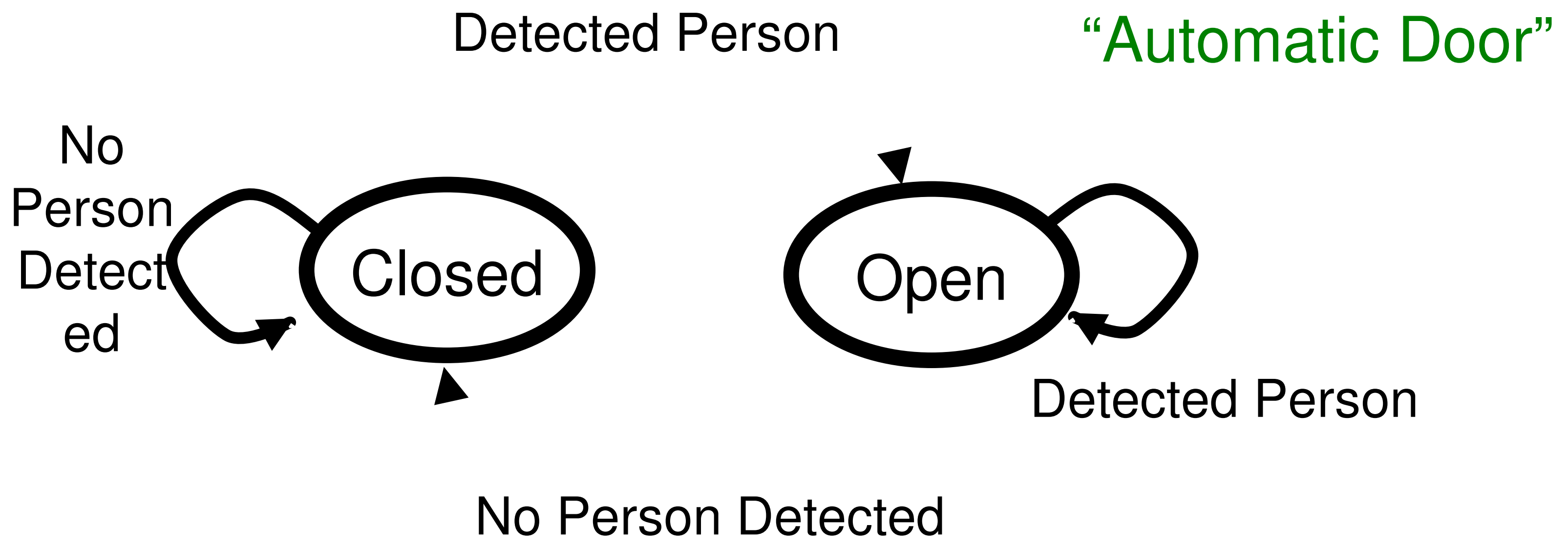
If we “toggle” the write input (setting it 1 then setting it 0) then M acquires the value of D.

“Timing Diagram”





FSMs (of Moore Type)



- Finite number of states
- Machine can produce outputs, these depend upon current state only
- Machine can accept one or more bits of input; reading these causes transitions among states.

Discussion Time

What are some examples of FSMs in the Hayes article?

How can we implement a FSM using logic gates etc.?

- If number of states = 2^k then represent “state” by k boolean variables.
- Identify number of input variables
- Write truth table expressing how “next state” is determined from “current state” and current values of the input.
- Express as clocked synchronous circuit.

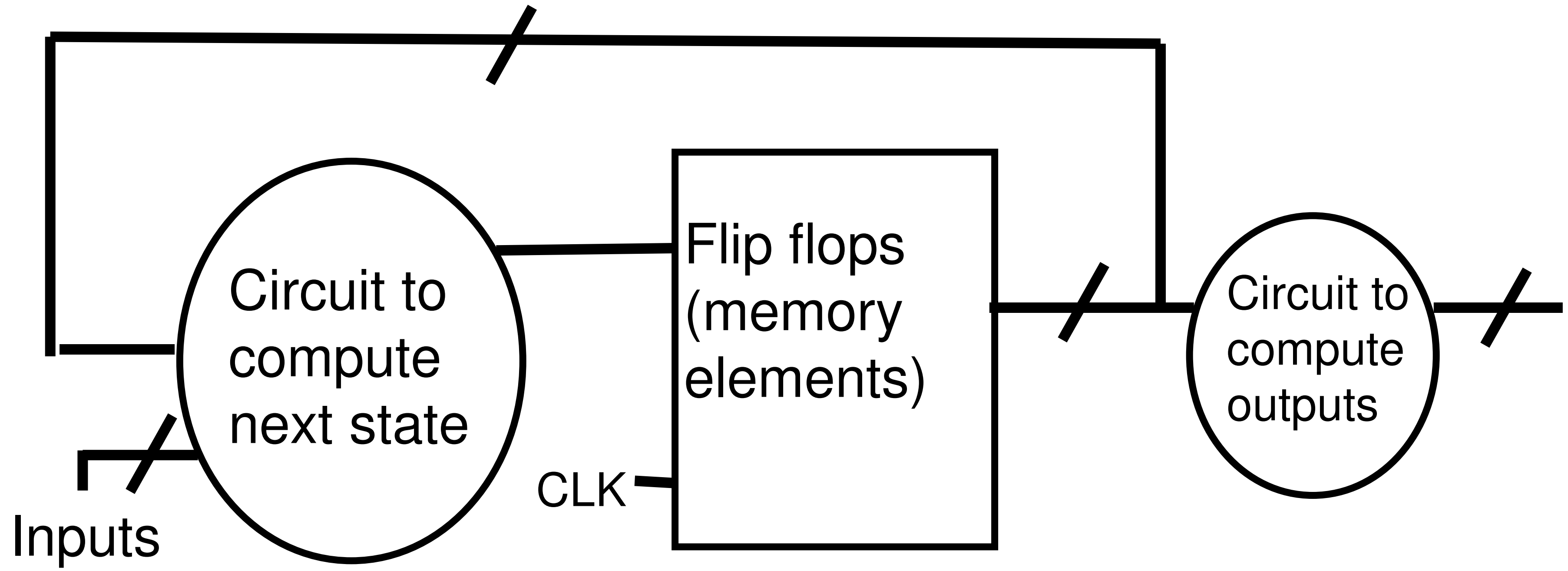
Example: 4-state machine; 1 bit of input; No output

State variables: P, Q
Input variable: D

Next value of P = (P+Q). D
Next value of Q = P

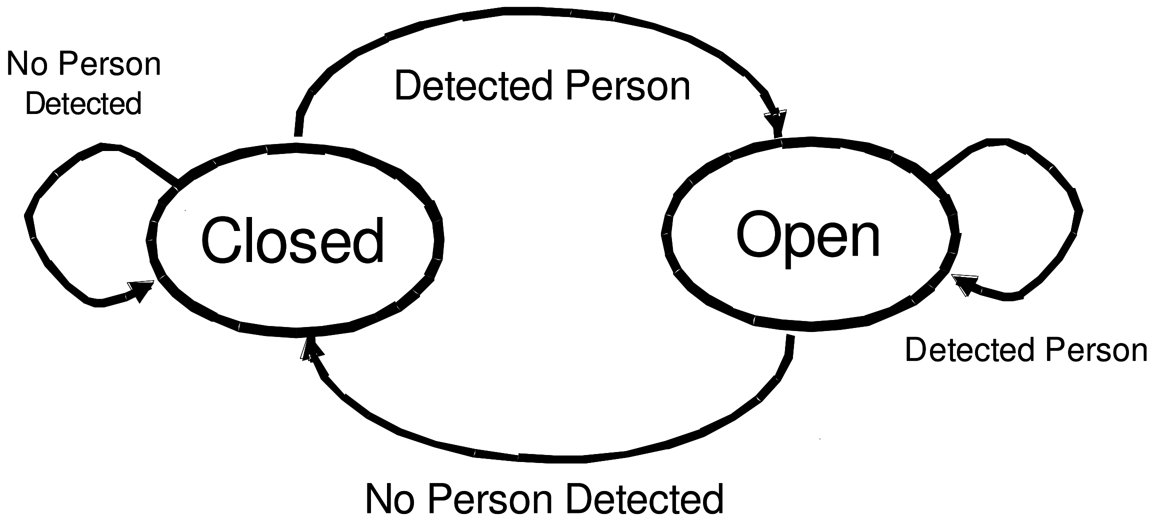
What is its state diagram?

Implementation: General Schematic


Circuit to compute next state
Flip flops (memory elements)
Circuit to compute outputs
CLK
Inputs

K Flip flops allow FSM to have 2^K states

Implementing door FSM as synchronous circuit



INPUT

0 = No Person Detected
1 = Person Detected

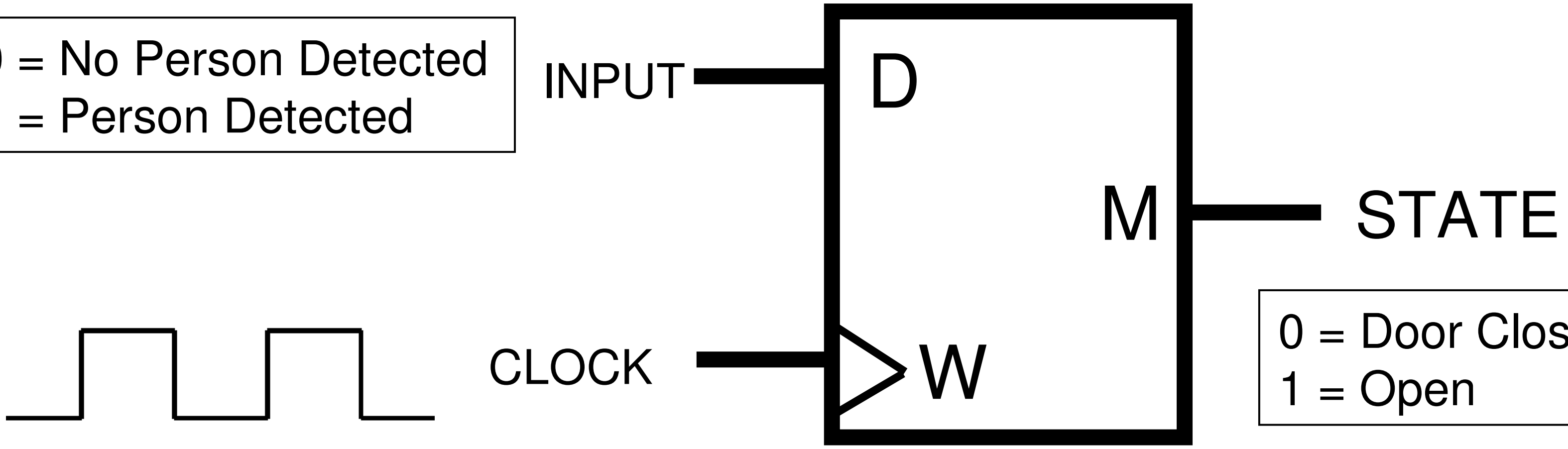
STATE

0 = Door Closed
1 = Open

Input	Present State	Next State
0	0	0
1	0	1
0	1	0
1	1	1

Implementation of door FSM (contd)

0 = No Person Detected
1 = Person Detected



0 = Door Closed
1 = Open

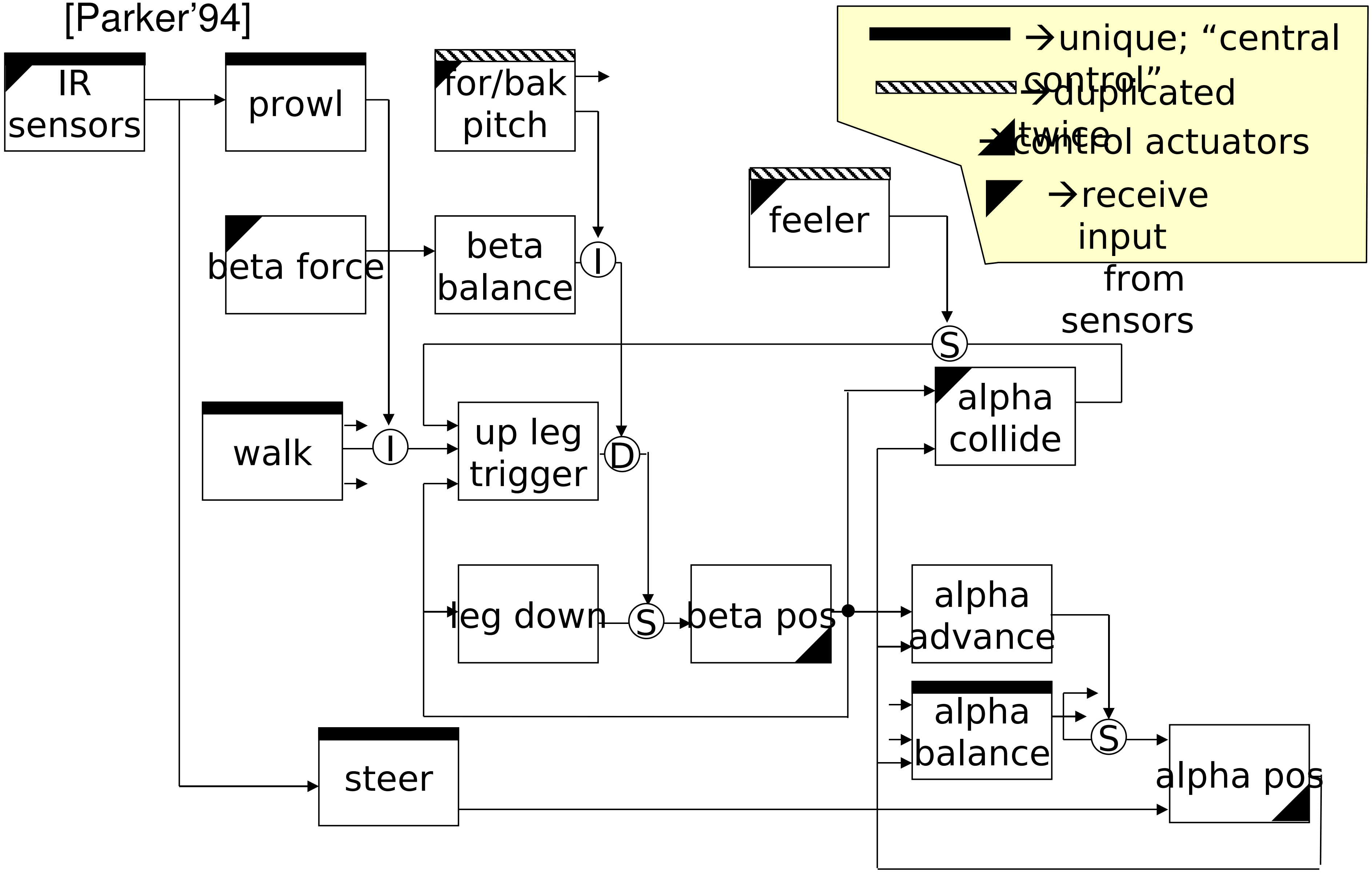
Other examples of FSMs

- Sisyphus

- Brook’s Genghis (51 FSMs) (see p. 46 in our text)
- Human Soul a la Aquinas (see Handout)

Aside: Portion of Genghis AFSM Network

[Parker'94]

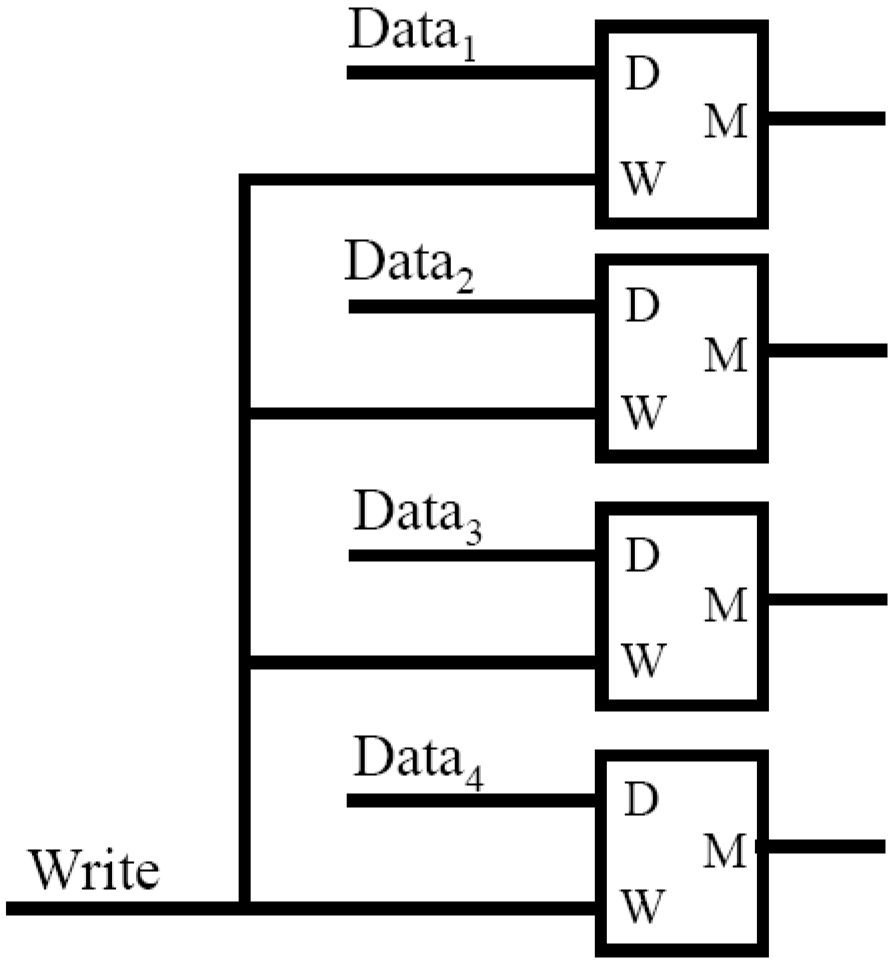


Next….

Random Access Memory (RAM)

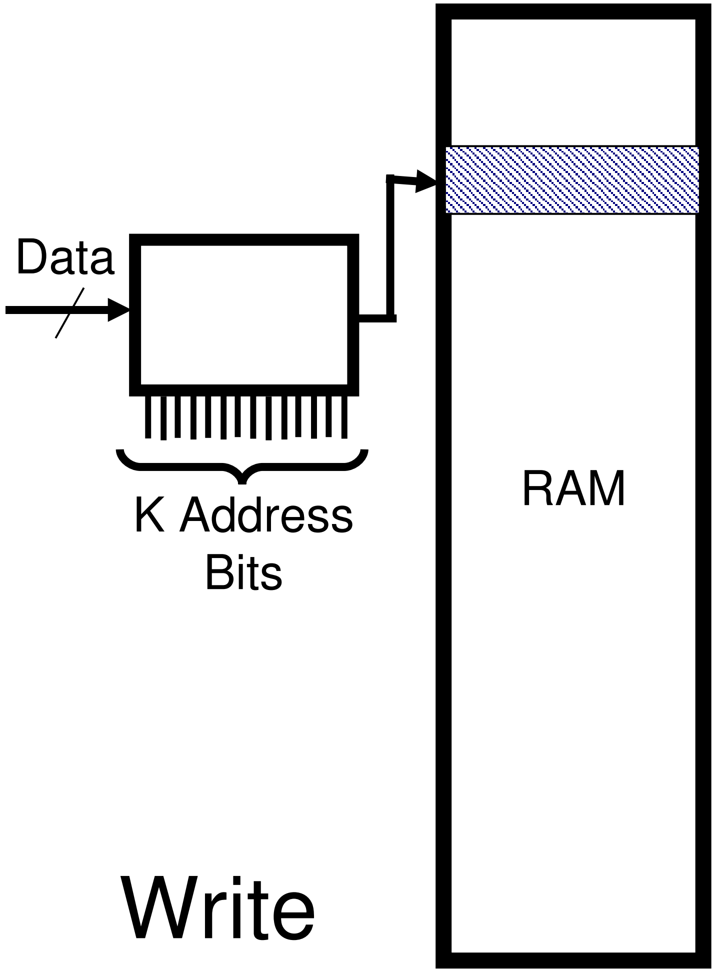
Memory where each location has an address

Recall from last lecture: "Register" with 4 bits of memory

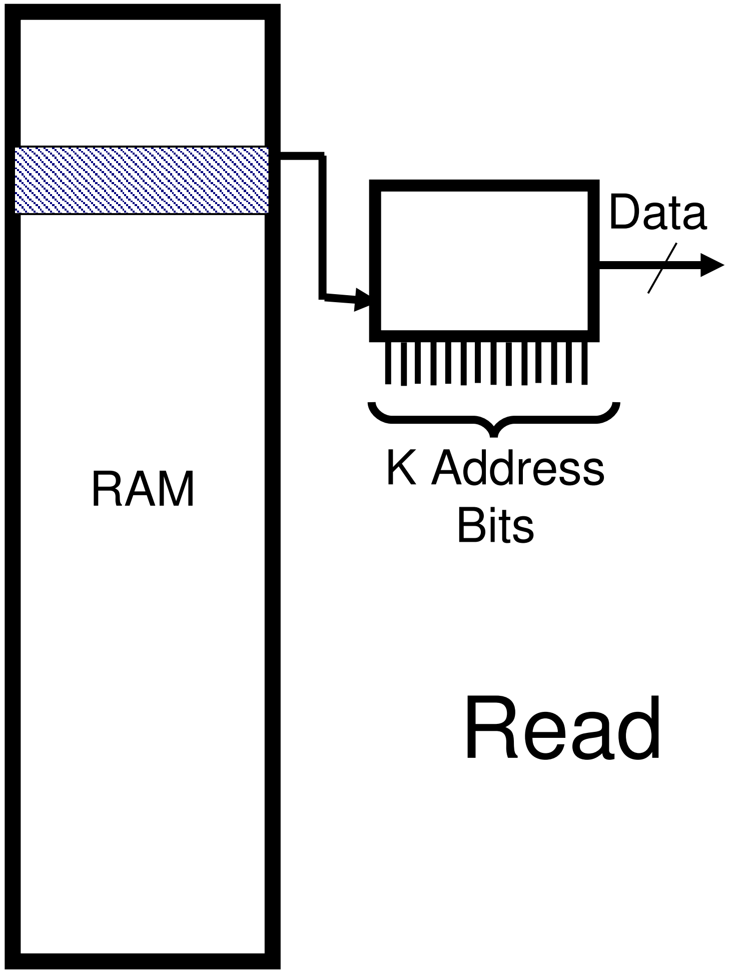


How can you set up an addressing system for large banks of memory?

RAM



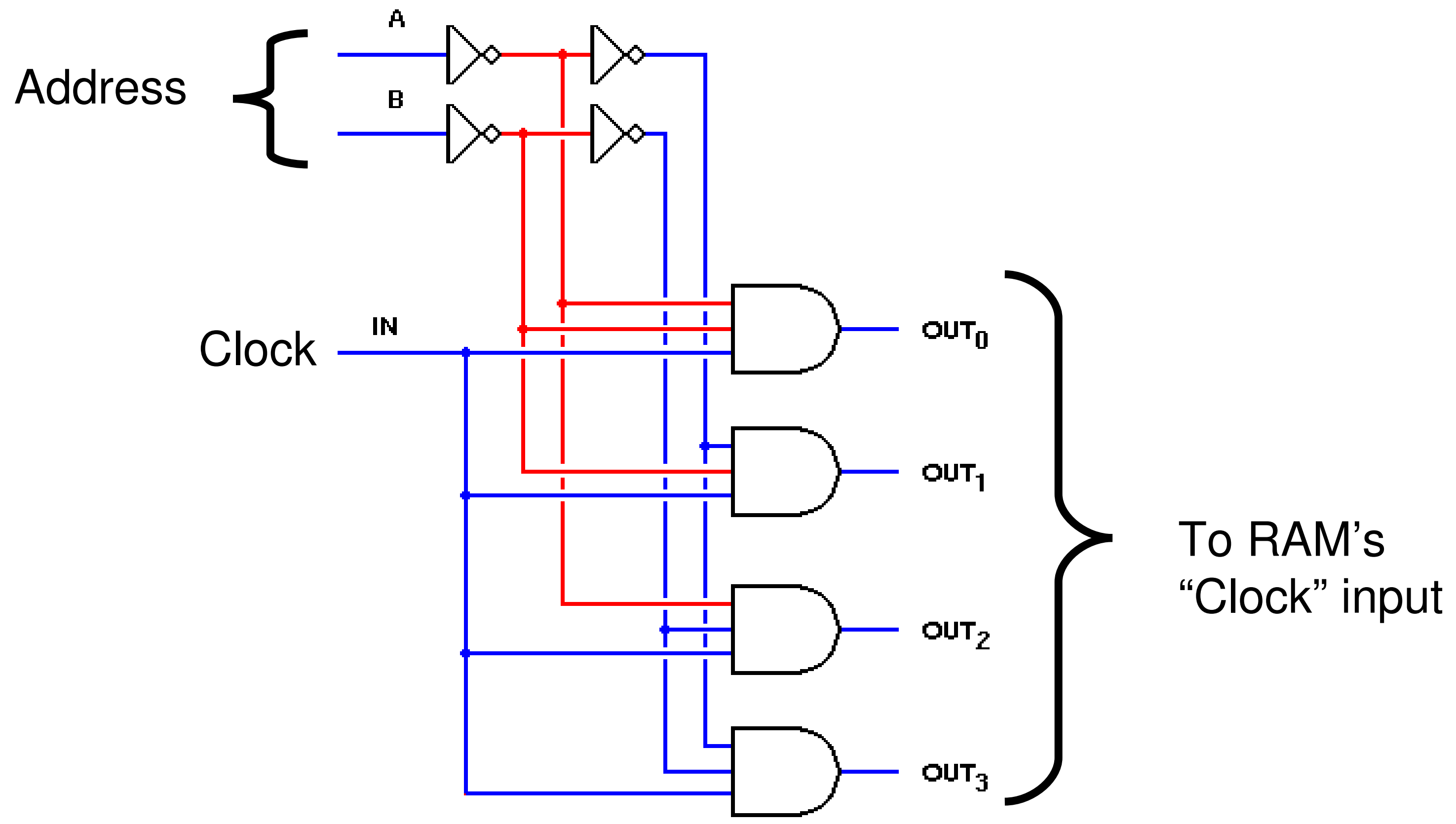
Write



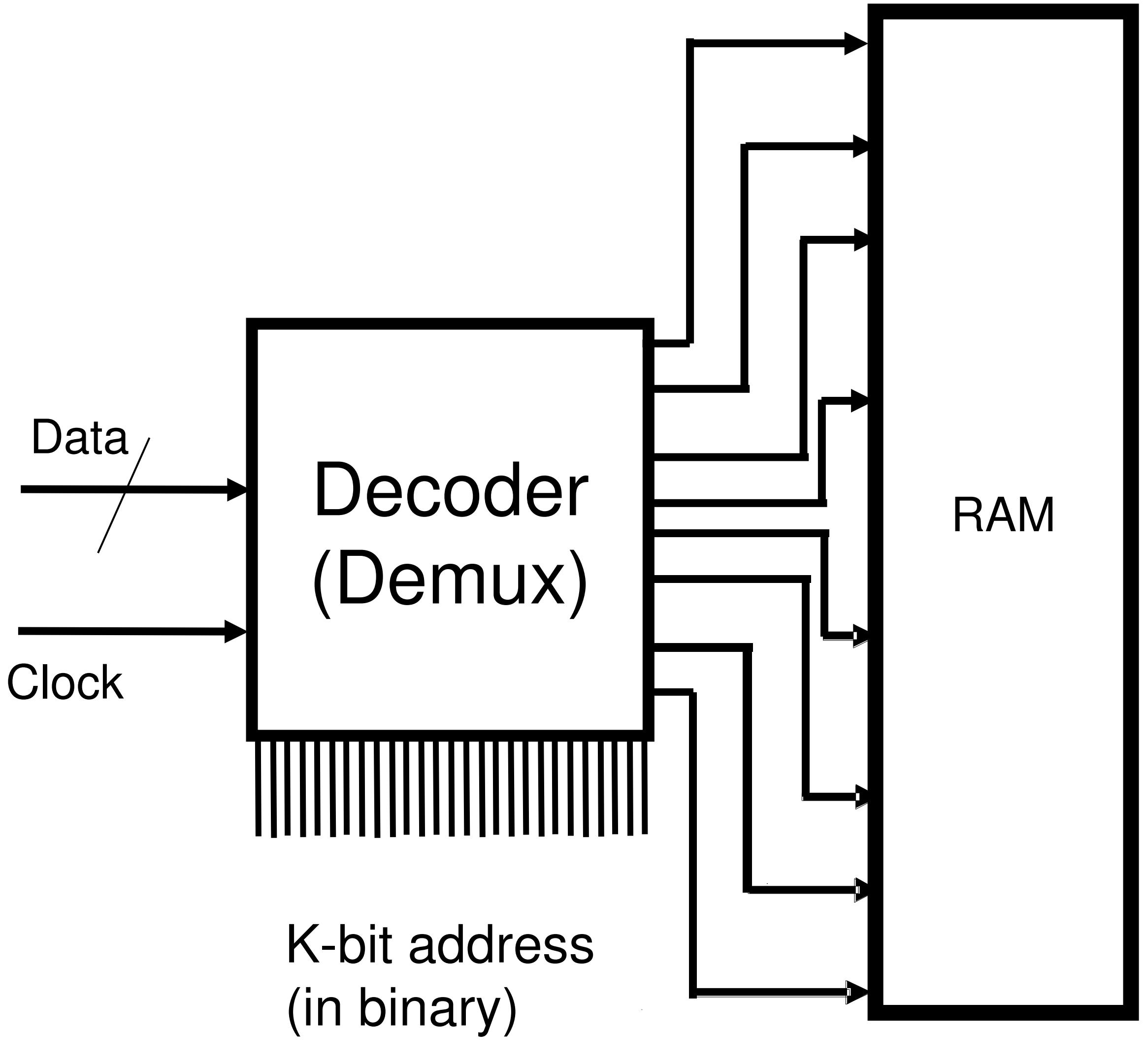
Read

2^K bits; bank of flipflops

If 4 locations, “address” has 2 bits



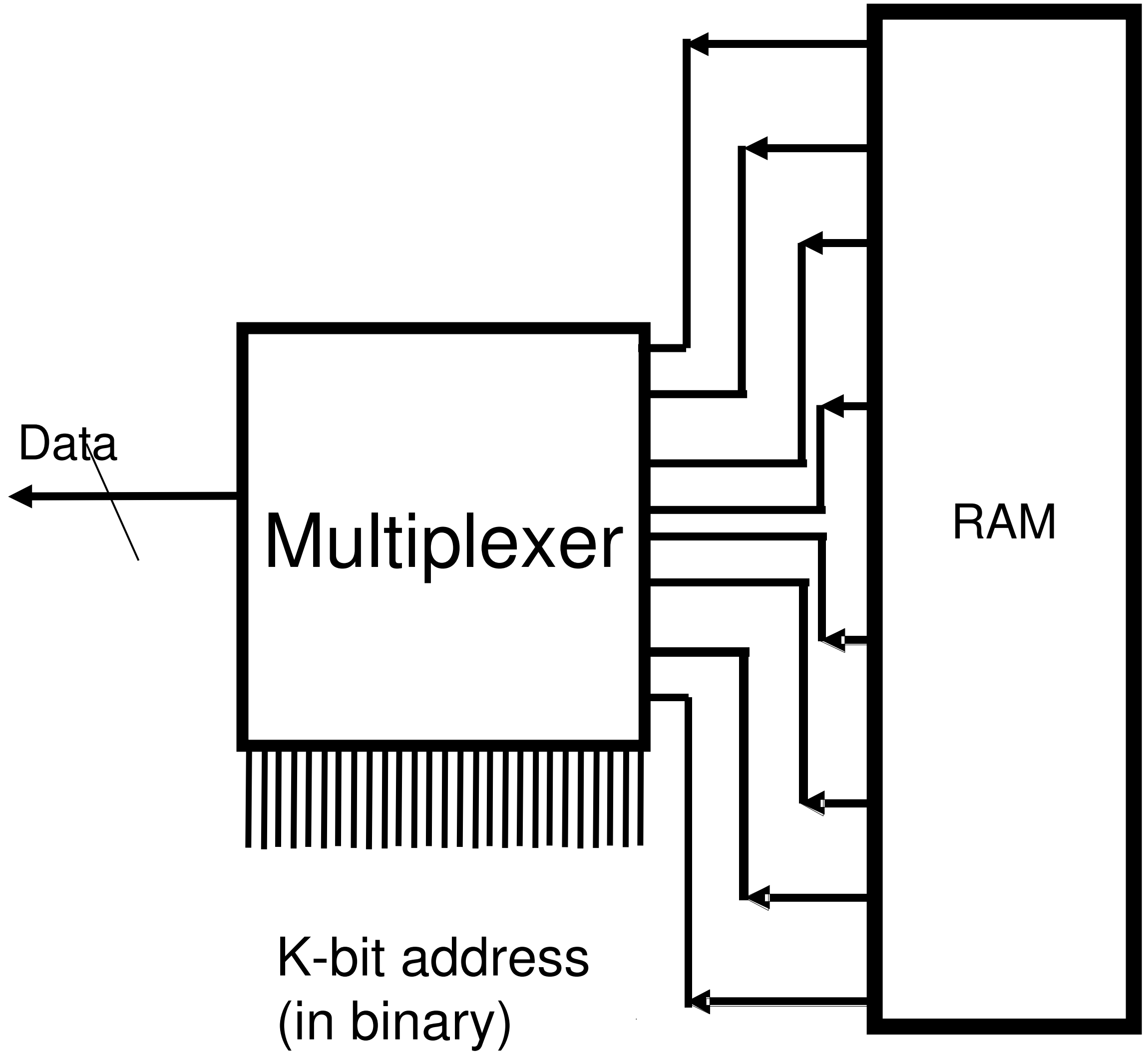
RAM: Implementing “Write”



The decoder selects which cell in the RAM gets its “Write” input toggled

(simple combinational circuit; see logic handout)

Ram: implementing "Read"



The multiplexer is connected to all cells in the RAM; selects the appropriate cell based upon the k-bit address

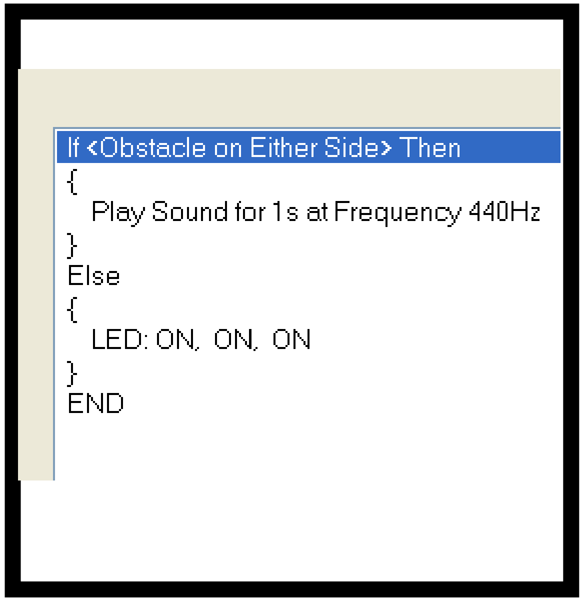
(simple combinational circuit; see logic handout)

Next, the secret revealed...

How computers execute programs.

CPU = Central Processing Unit

Scribbler Control Panel Program



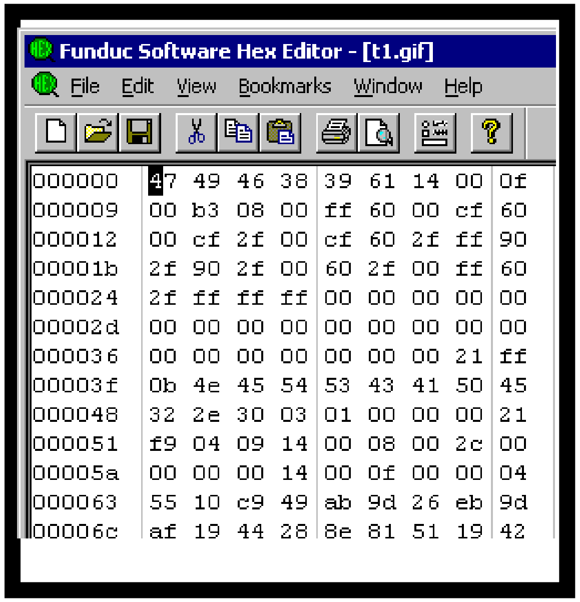
```
If <Obstacle on Either Side> Then
{
   Play Sound for 1s at Frequency 440Hz
}
Else
{
   LED: ON, ON, ON
}
END
```

F5

“Download to Robot”

(Compilation)

Machine Executable Code



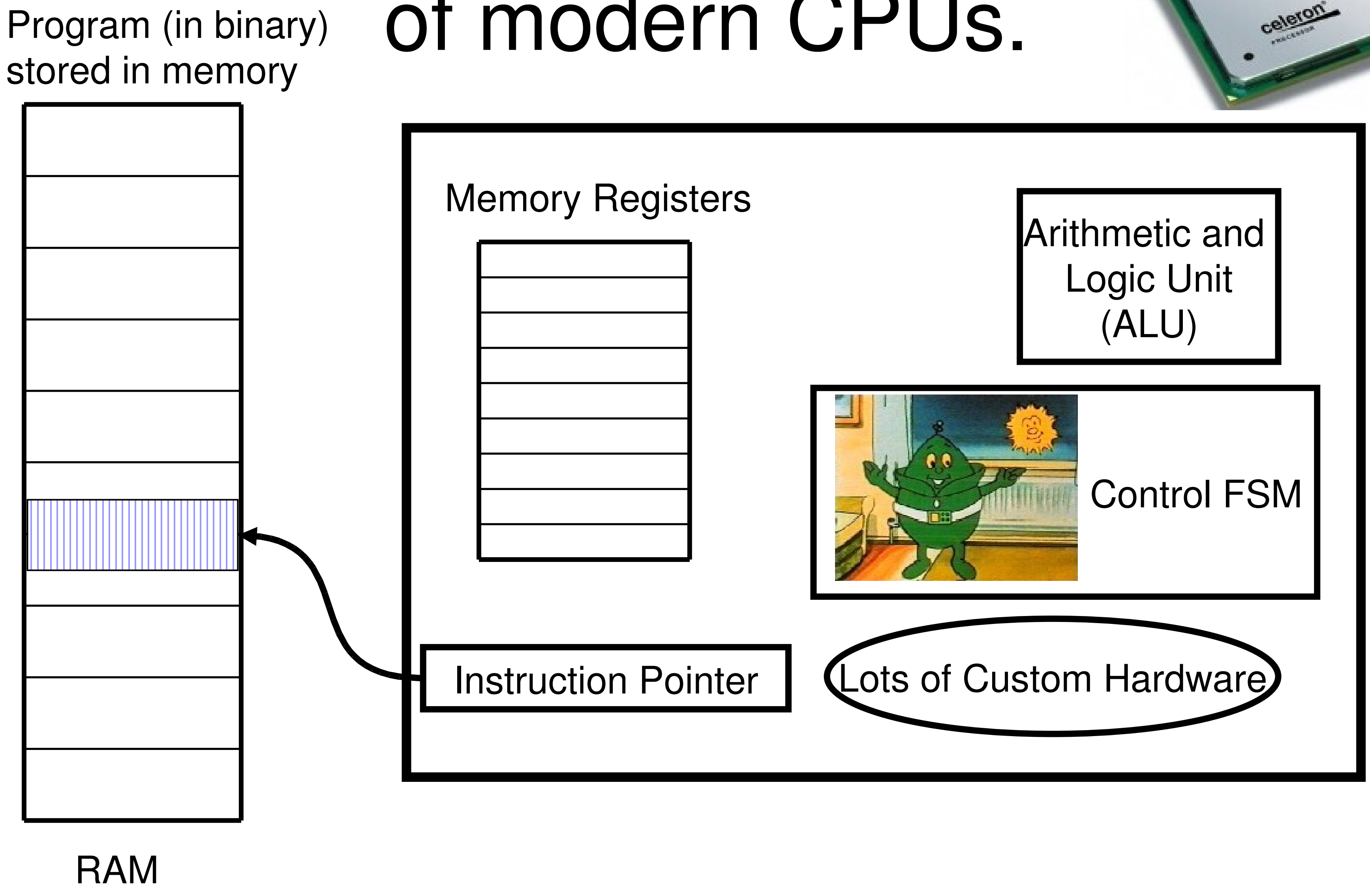
```
000000  47 49 46 38 39 61 14 00 0f
000009  00 b3 08 00 ff 60 00 cf 60
000012  00 cf 2f 00 cf 60 2f ff 90
00001b  2f 90 2f 00 60 2f 00 ff 60
000024  2f ff ff ff 00 00 00 00 00
00002d  00 00 00 00 00 00 00 00 00
000036  00 00 00 00 00 00 00 21 ff
00003f  0b 4e 45 54 53 43 41 50 45
000048  32 2e 30 03 01 00 00 00 21
000051  f9 04 09 14 00 08 00 2c 00
00005a  00 00 00 14 00 0f 00 00 04
000063  55 10 c9 49 ab 9d 26 eb 9d
00006c  af 19 44 28 8e 81 51 19 42
```

Point 1: Programs are “translated” into “machine language”; this is what’s get executed.

Similar to:

- T-P programs represented in binary
- .exe files in the Wintel world

Greatly simplified view of modern CPUs.



Examples of Machine Language Instructions

`ADD 3 7 12`	Add contents of Register 3 and Register 7 and store in Register 12
`LOAD 3 67432`	Read Location 67432 from memory and load into Register 3
`JUMP 4 35876`	If register 4 has a number > 0 set IP to 35876

Stored in binary (recall Davis's binary encoding of T-P programs)

Different CPUs have different machine languages

- Intel Pentium
- Power PC
- Palmpilot, etc.

“Backwards Compatibility” – Pentium 4’s machine language extends Pentium 2’s machine language

Machine languages now allow complicated calculations (eg for multimedia, graphics) in a single instruction

Example:
Intel press release

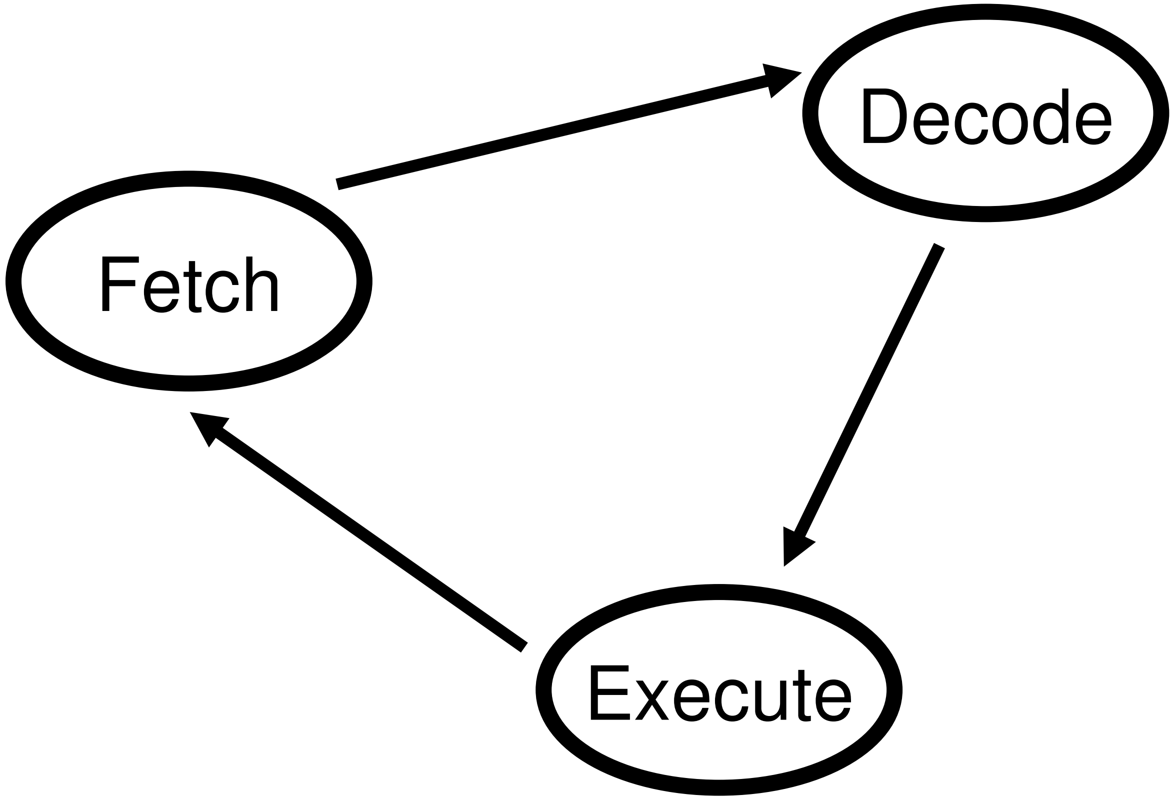
SANTA CLARA, Calif., June 28, 2004 - Intel Corporation today announced availability of a new Intel® Xeon™ processor-based platform and a host of new products and technologies for its Intel Xeon processor family that significantly boost performance, memory and graphics capabilities for workstation platforms. Workstations will benefit from rich set of new technologies that address the increasingly data-hungry systems and software applications that crave performance for a range of functions such as financial and scientific data modeling to digital filmmaking and design automation.

Main Insight

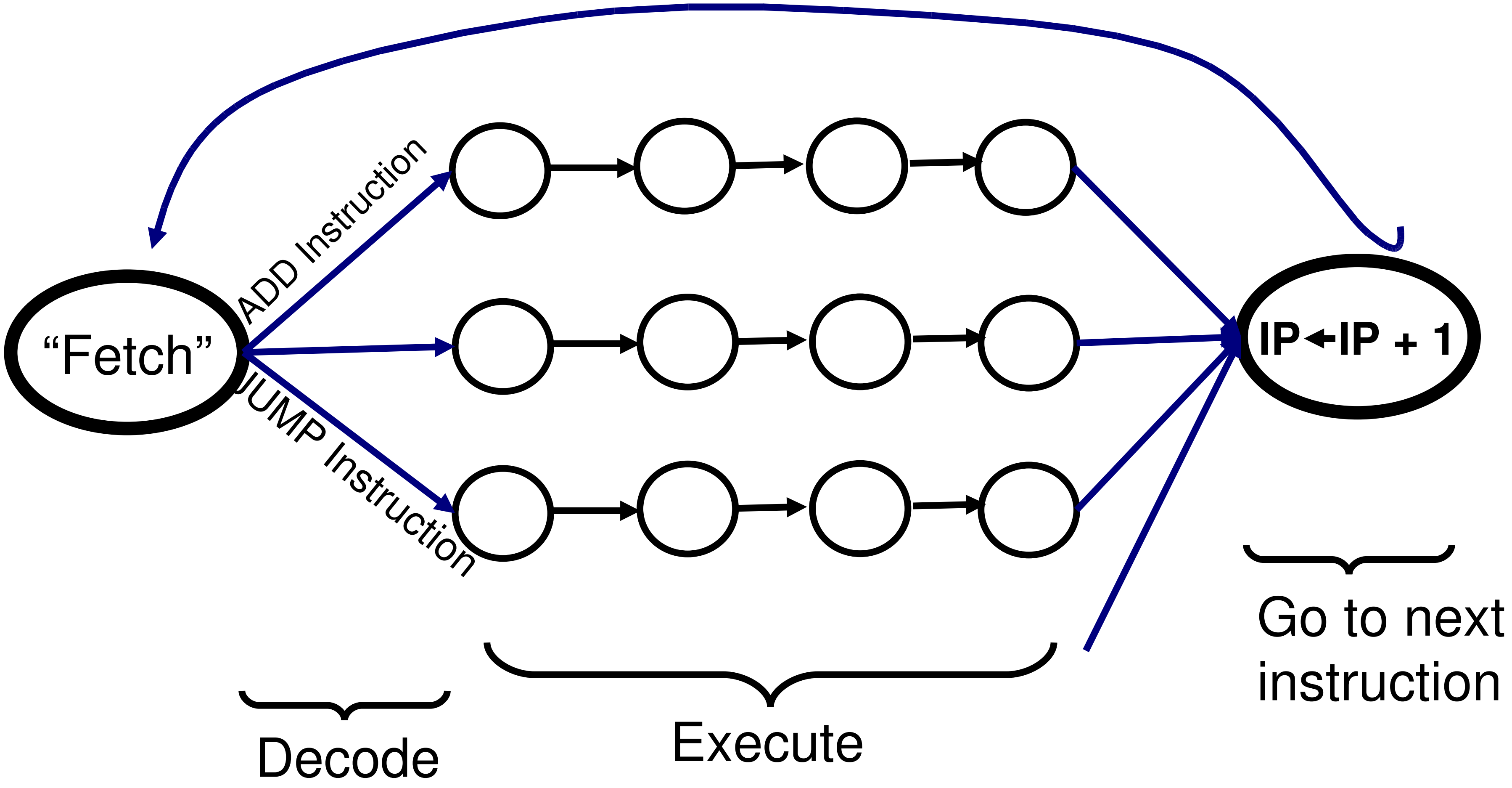
Computer = FSM controlling a larger (or infinite) memory.

Meet the little green man…

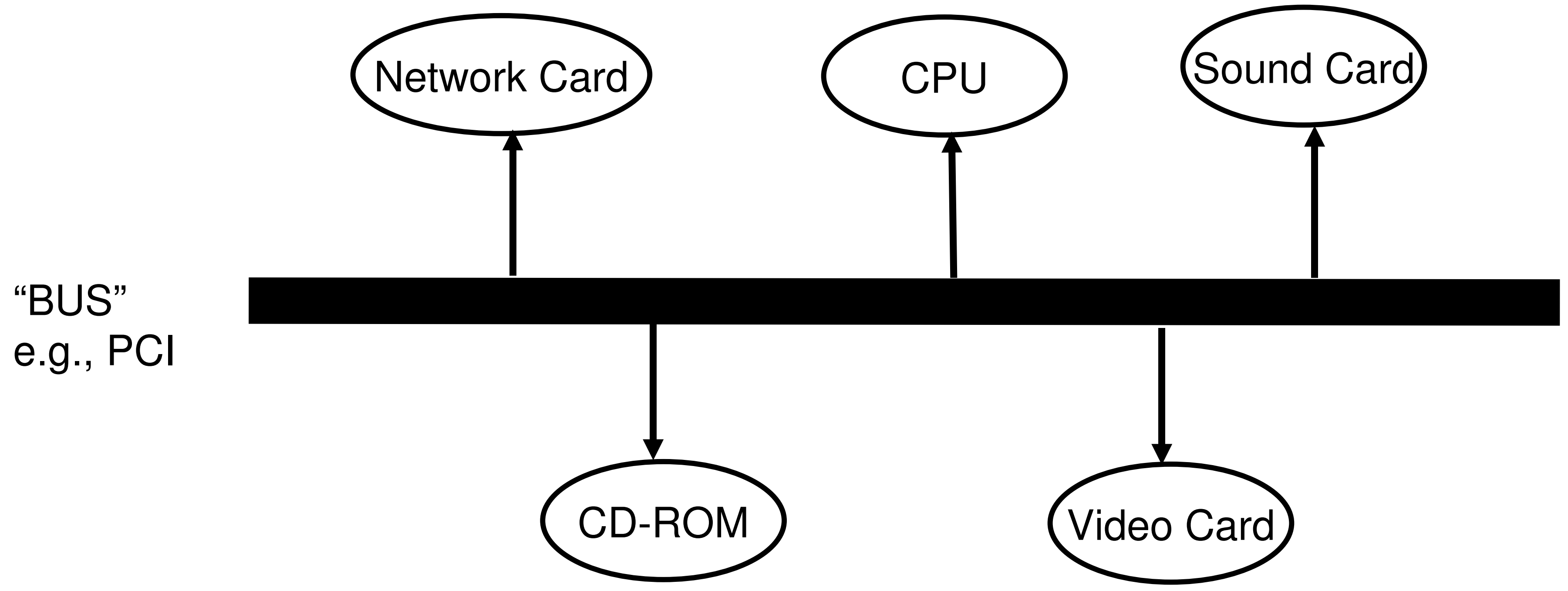
The Fetch – Decode – Execute FSM



Fetch – Decode – Execute FSM



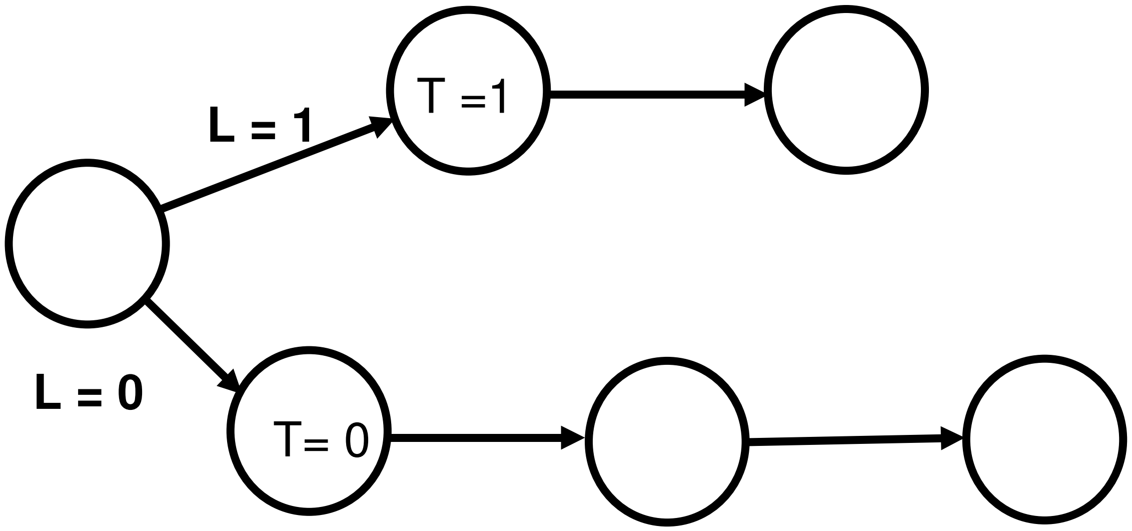
CPU as a conductor of a symphony



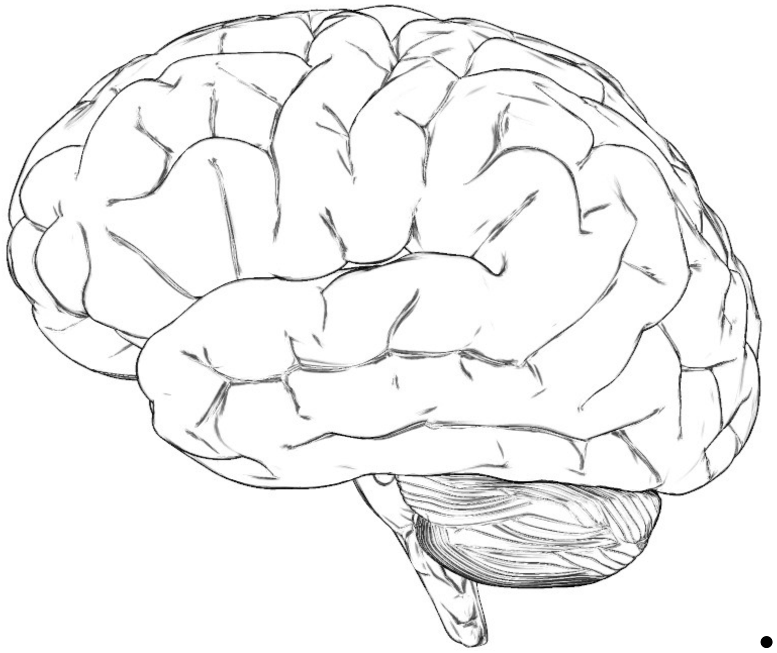
Bus: “Everybody hears everybody else”

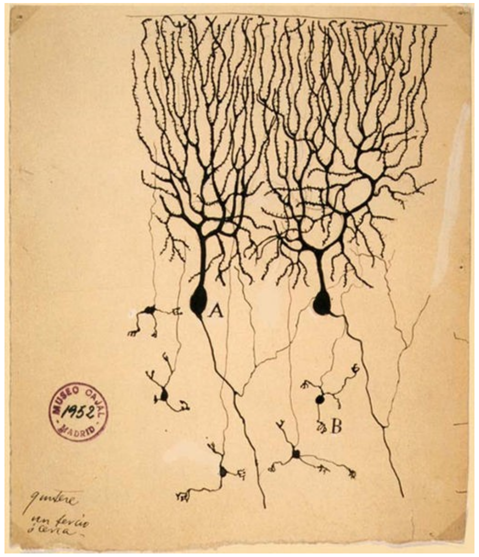
How an FSM does “reasoning”

“If left infrared sensor detects a person, turn left”



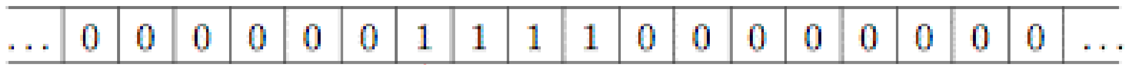
Speculation: Brain as FSM?





- Network ("graph") of 100 billion neurons; each connected to a few thousand others
- Neuron = tiny Computational Element; "switching time" 0.01 s
- Neuron generates a voltage spike depending upon how many neighbors are spiking.

Discussion: How would you implement a Turing-Post program with a digital circuit?



1.PRINT 0
2. GO RIGHT
3. GO TO STEP 1 if 1 SCANNED
4. GO TO STEP 1 if 0 SCANNED
5. STOP

Assume “PRINT” and “SCAN” as basic operations