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

COS 116, Spring 2010
Guest: Szymon Rusinkiewicz
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 on 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”

D

5V

0V

Time

W

5V

0V

Time

M

5V

0V

Time
Finite State Machines (FSMs)

- 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.

Diagram:

- Closed state
- Open state
- Detected Person transition
- No Person Detected transition

“Automatic Door”
Discussion
Time

What are some examples of FSMs?

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

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

<table>
<thead>
<tr>
<th>Input</th>
<th>Present State</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Implementation of door FSM (contd)

0 = No Person Detected
1 = Person Detected

0 = Door Closed
1 = Open
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

Address

Clock

To RAM’s “Clock” input
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

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

F5

“Download to Robot” (Compilation)

Machine Executable Code

Similar to:
• T-P programs represented in binary
• .exe files in the Wintel world
Greatly simplified view of modern CPUs.

Program (in binary) stored in memory

Memory Registers

Arithmetic and Logic Unit (ALU)

Control FSM

Instruction Pointer

Lots of Custom Hardware

RAM
### Examples of Machine Language Instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Op</th>
<th>R1</th>
<th>R2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD</strong></td>
<td>3</td>
<td>7</td>
<td>12</td>
<td>Add contents of Register 3 and Register 7 and store in Register 12</td>
</tr>
<tr>
<td><strong>LOAD</strong></td>
<td>3</td>
<td>67432</td>
<td></td>
<td>Read Location 67432 from memory and load into Register 3</td>
</tr>
<tr>
<td><strong>JUMP</strong></td>
<td>4</td>
<td>35876</td>
<td></td>
<td>If register 4 has a number &gt; 0 set IP to 35876</td>
</tr>
</tbody>
</table>

Stored in binary (recall Davis’s binary encoding of T-P programs)
Different CPUs have different machine languages

- Intel Pentium, Core, Xeon, etc. (PC, recent Mac)
- Power PC (old Mac)
- ARM (cellphones, mobile devices, etc.)

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

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

Computer = FSM controlling a larger (or infinite) memory.
Meet the little green man..

The Fetch – Decode – Execute FSM
Fetch – Decode – Execute FSM

“Fetch”

Decode

Execute

ADD Instruction

JUMP Instruction

Go to next instruction

IP ← IP + 1
CPU as a conductor of a symphony

Network Card

CPU

Sound Card

CD-ROM

Video Card

“BUS”
e.g., PCI

Bus: “Everybody hears everybody else”
How an FSM does “reasoning”

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

```
L = 0
L = 1
T = 1
T = 0
```
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.