CONTROL UNIT DESIGN

Introduction

* To design a control unit we need to design an FSM
* When you design an FSM, this is what you need to specify:

```
Inputs → Next-state logic → FSM state → Output logic → Outputs
```

* We will see two approaches to design the combinational logic (next-state and output)
An FSM can be represented in two ways:

1) State transition table

<table>
<thead>
<tr>
<th>Current state</th>
<th>Inputs</th>
<th>Next state</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2) State transition diagram:

```
 MAR ← PC  
 PC ← PC + 1

 MDR ← M[MAR]  
 R = 0

 R = 1
...
```

© 2017-2018 Juan Jose Jaramillo. All rights reserved.
Hardwired control unit

* Using the state transition table, we can design the combinational logic using the techniques learnt so far.
* For the state representation we can use a counter and the IR

Advantage: it can work at high speed
Disadvantage: rigid design
Microprogrammed control unit

Key insight: rewrite FSM diagram as flowchart

### FSM diagram

1. **MAR** ← **PC**
2. **PC** ← **PC + 1**
3. **MDR** ← **M[MAR]**  
   - **R** = 0
   - **R** = 1

### Flowchart

1. **MAR** ← **PC**
2. **PC** ← **PC + 1**
3. **MDR** ← **M[MAR]**
   - **R**?  
     - **R** = 0
     - **R** = 1
FSM becomes microprogram that executes microinstructions.

LC-3 state machine was designed with microprogrammed control approach in mind.
First approach: use a Read-Only Memory (ROM)

Question: how do we handle conditional constructs? (E.g., wait for memory read, branch instruction)
Question: what about the DECODE phase, where the opcode indicates the next state?
Advantage: flexible, easy to alter instruction set
Disadvantage: slow compared to hardwired design