### Outline

- Last time:
  - Introduction to Computer Organization
  - Control
  - Datapath
  - I/O Interface
  - Bussing Strategies

- This lecture:
  - Deriving the State Diagram & Datapath (Cont.)
  - Mapping the Datapath onto Control

### Finite State Machines for Simple CPUs

**State Diagram and Datapath Derivation**

**Processor Specification:**

- Instruction Format:
  - Operation: 00 = LD, 01 = ST, 10 = ADD, 11 = BRN

- Memory Interface:
  - Load from memory: Mem[XXX] → AC
  - Store to memory: AC → Mem[XXX]
  - Add from memory: AC + Mem[XXX] → AC
  - Branch if accumulator is negative: AC < 0 → XXX → PC

**Deriving the State Diagram and Datapath**

**Assume Synchronous Mealy Machine:**

- Transitions associated with arcs rather than states

**Reset State (State 0) and Instruction Fetch Sequence**

- On Reset:
  - zero the PC
  - Mem Request unasserted
  - Mem asserts Wait signal

- Instruction Fetch:
  - issue read request
  - 4 cycle handshake on Wait signal

**Note:** No explicit mention of the busses being used to implement register transfers!
Deriving the State Diagram and Datapath

Operation Decode State

Four Way Next State Branch based on opcode bits

Load Sequence
like IFetch, except that operand address comes from IR and data should be loaded into AC

Store Execution Sequence
Memory write sequence
Deriving the State Diagram and Datapath

Store Execution Sequence
Memory write sequence

Add Execution Sequence
Similar to Load sequence
Add MBR, AC rather than simply transfer MBR to AC.

Similar to Load sequence
Add MBR, AC rather than simply transfer MBR to AC.
Deriving the State Diagram and Datapath

Add Execution Sequence
- Similar to Load sequence
- Add MBR, AC rather than simply transfer MBR to AC

Branch Execution Sequence
- IR<15:14> = 10/
- IR<13:0> ® MAR
- Wait/
- MAR ® Memory
- 1 ® Read/Write
- 1 ® Request
- Wait/Mem ® MBR
- Wait/
- MBR + AC ® AC
- Wait/
- OD
- AD0
- AD1
- AD2
- RES

Branch Execution Sequence
- Replace PC withOperand Address if AC < 0
- Otherwise, do nothing

Revised/Complete State Diagram
- Simplify Wait Looping
- Eliminate some Wait states

At this point, Wait must be asserted, so why loop on Wait?
- Why loop on Wait when resync will take place at state IF0?

Processor Signal Flow
Specification so far is independent of bussing strategy.

Implied transfers:

This is the point-to-point connection scheme.

Observe that instruction fetch and operand fetch take place at different times.

This implies that IR, PC, and MAR transfers can be implemented by single bus (Address Bus).

Combine MBR, IR, ALU B, and AC connections (Memory Bus).

Combine ALU, AC, and MBR connections (Result Bus).

Three bus architecture:

AC + MBR → AC implemented in single state.

AC has two inputs, RBUS and MBUS.

(Other registers except MBR have single input and output)

Dual ported configuration is more complex.

Better idea: reuse existing paths were possible.

MBR → AC transfer implemented by PASS B ALU operation.

Detailed implementation of register transfer operations.

More detailed control operations are called microoperations.

One register transfer operation = several microoperations.

Some operations directly implemented by functional units:

- E.g., ADD, Pass B, 0 → PC, PC + 1 → PC

Some operations require multiple control operations:

- E.g., PC → MAR implemented as PC → ABUS and ABUS → MAR

Timing of State Changes and Microoperations.

Tri-state Control.

Load Input

PC implemented by counter with COUNT and CLEAR inputs.
Mapping onto Datapath Control
Relationship between register transfer and microoperations:

<table>
<thead>
<tr>
<th>Register Transfer</th>
<th>Microoperations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 → PC</td>
<td>0 → PC (delayed);</td>
</tr>
<tr>
<td>PC + 1 → PC</td>
<td>PC + 1 → PC (delayed);</td>
</tr>
<tr>
<td>PC → MAR</td>
<td>MAR → MAR (delayed);</td>
</tr>
<tr>
<td>Data Bus → MAR</td>
<td>Data Bus → MAR (delayed);</td>
</tr>
<tr>
<td>MBR → Data Bus</td>
<td>MBR → Data Bus (immediate);</td>
</tr>
<tr>
<td>MBR → IR</td>
<td>MBR → ARUS (immediate),</td>
</tr>
<tr>
<td>MBR → AC</td>
<td>MBR → ARUS (immediate),</td>
</tr>
<tr>
<td>IR&lt;13:0&gt; → MAR</td>
<td>MAR → ARUS (immediate),</td>
</tr>
<tr>
<td>IR&lt;13:0&gt; → PC</td>
<td>IR → MAR (immediate),</td>
</tr>
<tr>
<td>1 → Read/Write</td>
<td>Read (immediate),</td>
</tr>
<tr>
<td>0 → Read/Write</td>
<td>Write (immediate),</td>
</tr>
<tr>
<td>1 → Request</td>
<td>Request (immediate),</td>
</tr>
</tbody>
</table>

5 inputs
make sure that Reset and Wait are synchronized
16 datapath control lines
2 memory control lines

Special microoperations for AC → ALU and ALU Result → RBUS not strictly necessary since these connections can be hardwired.