Where we are headed

- Single Cycle Problems:
  - what if we had a more complicated instruction like floating point?
  - wasteful of area
- One Solution:
  - use a “smaller” cycle time
  - have different instructions take different numbers of cycles
  - a “multicycle” datapath:
Multicycle Approach

- We will be reusing functional units
  - ALU used to compute address and to increment PC
  - Memory used for instruction and data
- Our control signals will not be determined solely by instruction
  - e.g., what should the ALU do for a “subtract” instruction?
- We’ll use a finite state machine for control
Multicycle Approach

• Break up the instructions into steps, each step takes a cycle
  – balance the amount of work to be done
  – restrict each cycle to use only one major functional unit
• At the end of a cycle
  – store values for use in later cycles (easiest thing to do)
  – introduce additional “internal” registers
Multicycle Datapath
Five Execution Steps

- Instruction Fetch
- Instruction Decode and Register Fetch
- Execution, Memory Address Computation, or Branch Completion
- Memory Access or R-type instruction completion
- Write-back step

*INSTRUCTIONS TAKE FROM 3 - 5 CYCLES!*
Step 1: Instruction Fetch

- Use PC to get instruction and put it in the Instruction Register.
- Increment the PC by 4 and put the result back in the PC.
- Can be described succinctly using RTL "Register-Transfer Language"

\[
\begin{align*}
IR &= \text{Memory}[PC]; \\
PC &= PC + 4;
\end{align*}
\]

Can we figure out the values of the control signals?

What is the advantage of updating the PC now?
Step 2: Instruction Decode and Register Fetch

• Read registers rs and rt in case we need them
• Compute the branch address in case the instruction is a branch
• RTL:

\[
\begin{align*}
A &= \text{Reg}[\text{IR}[25-21]]; \\
B &= \text{Reg}[\text{IR}[20-16]]; \\
\text{ALUOut} &= \text{PC} + (\text{sign-extend}(\text{IR}[15-0]) \ll 2);
\end{align*}
\]

• We aren’t setting any control lines based on the instruction type (we are busy "decoding" it in our control logic)
Step 3 (instruction dependent)

- ALU is performing one of three functions, based on instruction type

- Memory Reference:

  \[ ALUOut = A + \text{sign-extend}(\text{IR}[15-0]) \];

- R-type:

  \[ ALUOut = A \text{ op } B \];

- Branch:

  \[ \text{if } (A==B) \text{ PC } = \text{ALUOut} \];
Step 4 (R-type or memory-access)

• Loads and stores access memory
  
  \[
  \text{MDR} = \text{Memory}[\text{ALUOut}];
  \]
  
  or
  
  \[
  \text{Memory}[\text{ALUOut}] = B;
  \]

• R-type instructions finish
  
  \[
  \text{Reg}[\text{IR}[15-11]] = \text{ALUOut};
  \]

  The write actually takes place at the end of the cycle on the edge
Write-back step

- \( \text{Reg}[\text{IR}[20-16]] = \text{MDR}; \)

*What about all the other instructions?*
### Summary:

<table>
<thead>
<tr>
<th>Step name</th>
<th>Action for R-type instructions</th>
<th>Action for memory-reference instructions</th>
<th>Action for branches</th>
<th>Action for jumps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction fetch</td>
<td></td>
<td>IR = Memory[PC]</td>
<td>PC = PC + 4</td>
<td></td>
</tr>
<tr>
<td>Instruction decode/register fetch</td>
<td></td>
<td>A = Reg [IR[25-21]]</td>
<td>B = Reg [IR[20-16]]</td>
<td></td>
</tr>
<tr>
<td>Execution, address computation, branch/ jump completion</td>
<td>ALUOut = A op B</td>
<td>ALUOut = A + sign-extend (IR[15-0])</td>
<td>if (A ==B) then PC = ALUOut</td>
<td>PC = PC [31-28] II (IR[25-0]&lt;2)</td>
</tr>
<tr>
<td>Memory access or R-type completion</td>
<td>Reg [IR[15-11]] = ALUOut</td>
<td>Load: MDR = Memory[ALUOut] or Store: Memory [ALUOut] = B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory read completion</td>
<td></td>
<td>Load: Reg[IR[20-16]] = MDR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simple Questions

- How many cycles will it take to execute this code?

```
lw $t2, 0($t3)
lw $t3, 4($t3)
beq $t2, $t3, Label  #assume not
add $t5, $t2, $t3
sw $t5, 8($t3)
Label: ...```

- What is going on during the 8th cycle of execution?
- In what cycle does the actual addition of $t2$ and $t3$ takes place?
Implementing the Control

• Value of control signals is dependent upon:
  – what instruction is being executed
  – which step is being performed

• Use the information we’ve accumulated to specify a finite state machine
  – specify the finite state machine graphically, or
  – use microprogramming

• Implementation can be derived from specification
Graphical Specification of FSM

- How many state bits will we need?
Finite State Machine for Control

- Implementation: