Designing a Pipelined CPU

CSE 141 Dean Tullsen

Review -- Single Cycle CPU

CSE 141 Dean Tullsen

Review -- Multiple Cycle CPU

CSE 141 Dean Tullsen

Review -- Instruction Latencies

CSE 141 Dean Tullsen

• Single-Cycle CPU

Load Ifetch Reg/Dec Exec Mem Wr

• Multiple Cycle CPU

Cycle 1 Cycle 2 Cycle 3 Cycle 4 Cycle 5

Load Ifetch Reg/Dec Exec Mem Wr

Add Ifetch Reg/Dec Exec Wr
Instruction Latencies and Throughput

- **Single-Cycle CPU**
  - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr

- **Multiple Cycle CPU**
  - Cycle 1
    - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr
  - Cycle 2
    - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr
  - Cycle 3
    - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr
  - Cycle 4
    - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr
  - Cycle 5
    - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr

- **Pipelined CPU**
  - Cycle 1
    - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr
  - Cycle 2
    - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr
  - Cycle 3
    - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr
  - Cycle 4
    - Load
    - Ifetch
    - Reg/Dec
    - Exec
    - Mem
    - Wr

Pipelining Advantages

- Higher maximum
- Higher of CPU resources
- But, more complicated, more complex (?)

Pipelining in Modern CPUs

- CPU Datapath
- Arithmetic Units
- System Buses
- Software (at multiple levels)
- etc...

<table>
<thead>
<tr>
<th>CPU Design Technology</th>
<th>Control Logic</th>
<th>Peak Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Cycle CPU</td>
<td>Combinational Logic</td>
<td>1</td>
</tr>
<tr>
<td>Multiple-Cycle CPU</td>
<td>FSM or Microprogram</td>
<td>1</td>
</tr>
<tr>
<td>Pipelined CPU</td>
<td>Combinational Logic</td>
<td>1</td>
</tr>
</tbody>
</table>
A Pipelined Datapath

IF: Instruction fetch
ID: Instruction decode and register fetch
EX: Execution and effective address calculation
MEM: Memory access
WB: Write back
Pipeline Principles

- All instructions that share a pipeline must have the same in the same
  - therefore, \textit{add} does nothing during Mem stage
  - \textit{sw} does nothing during WB stage
- All values must be latched each cycle.
- There is no .

Pipelined Datapath

The Pipeline in Execution

\texttt{add} $10, S1, S2$

The Pipeline in Execution

\texttt{lw} $12, 1000(S4)$ \texttt{add} $10, S1, S2$
The Pipeline in Execution, with controls

Pipelined Control

- can’t use
- not really appropriate
  - signals generated once, but follow instruction through the pipeline

Pipelined Control Signals

The Pipeline with Control Logic
Is it really that easy?

- What happens when...
  - add $3, $10, $11
  - lw $8, 1000($3)
  - sub $11, $8, $7
Data Hazards

• When a result is needed in the pipeline before it is available, a “data hazard” occurs.

sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)