Outline
- Project 4 Demo QA
- CPI - Performance
- Caches

Project 4
- Separation in sub-systems
  - Data path
  - Sequencer
  - Memory
  - Control & decoding

- Techniques
  - Hiding vs showing details
  - Probes
  - Decode implementation
    - Gates
    - Decoders
    - MUXes

Ideal
Basic Machine Org

Cycles Per Inst
WCI $t$ With Jmp/Br

\[
\text{cycle time} \ ?
\]
\[
P_C = P_C + X \quad ? \quad \text{IFetch}
\]

\[
\begin{array}{l}
\text{IR} \\
\text{Dec/} \quad \text{op/dec} \\
\text{EXEC} \\
\text{MEM} \\
\text{DMA} \\
\end{array}
\]

\[
\begin{array}{l}
\text{PC} \\
\text{I Mem} \\
\end{array}
\]
Performance

Time for Program (s) = \( \frac{\text{seconds}}{\text{Program}} = \frac{\text{seconds}}{\text{cycle}} \times \text{cycle time} \times \text{CPU} \times \text{# instructions} \)

Two ways to determine CPU:

Top down:
- measure \( \text{sec / min} \)
- simulate to count \# inst. (or counter)
- look up cycle time

Bottom up:
- simulate to determine inst mix

\[
\begin{align*}
50\% & \text{ Arith} \times \frac{1}{2} = 2.0 \\
30\% & \text{ Ld / S} \times \frac{5}{3} = 1.5 \\
20\% & \text{ Jmp Br} \times 2 = 0.4 \\
\hline
3.9
\end{align*}
\]

Are CPU = \( \sum f_i \cdot C_i \)

Memory:

Big memories are slow.

Real:
Address \( \Rightarrow \) data most recently written to that address
Write:
Address, Data \( \Rightarrow \) writes data to address

Program: series of Real (Addr), Write (Addr, Data)
Caches: small fast memories that hold the "most important" words of memory

**Hit:** Proc makes access to address/value "block" that is present in proc cache
  - read - return value
  - write - update value (mem?)

**Miss:** Not present in cache

<table>
<thead>
<tr>
<th>Hit</th>
<th>Miss Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit</td>
<td>Miss Cache</td>
</tr>
<tr>
<td>50%</td>
<td>42.0</td>
</tr>
<tr>
<td>30%</td>
<td>15.1</td>
</tr>
<tr>
<td>20%</td>
<td>2.4</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>

Stores value @ address

holds address/value pairs

**Proc** reads data from memory

**MAP**

**Cache**
Cache Operation

Read (addr)

- lookup block

\[
(Present, block) = \text{lookup}(addr)
\]

if (Present) return cache[block].data

else

\[
data = \text{read mem}(addr)
\]

\[
block = \text{place}(addr)
\]

\[
cache[block] = <addr, data>
\]

* what if

Write (addr, data)

\[
\text{mem}[addr] = block
\]

\[
(Present, block) = \text{lookup}(addr)
\]

if (Present)

\[
cache[block].data = data
\]

\[
\text{mem}[addr] = block
\]

write thr

no alloc

write-thru

Fully Associative Cache
Direct Map Cache

Cache Design Parameters:
- # blocks
- block size
- associativity
- write policy
- replacement policy

Hit: 1 cycle
Miss: $2 + \text{miss penalty}$ (say 100 cycles)

\[
\text{Miss Rate} = \frac{\text{# misses}}{\text{total mem refS}} = \frac{\text{misses}}{\text{hits + misses}}
\]

Wait on miss

\[
\text{CPI} = \text{ideal CPI} + \text{Acc Wait} + \text{Miss Rate} \times \text{Miss Penalty}
\]

ex: 2% @ 100 cycles $\Rightarrow$ 2 CPI

\[
3.4 \Rightarrow 5.9
\]

Table on Page 4