CPI & Processor Tradeoffs

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Type Cycles</th>
<th>Type Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>1</td>
<td>50%</td>
</tr>
<tr>
<td>Load</td>
<td>5</td>
<td>20%</td>
</tr>
<tr>
<td>Store</td>
<td>3</td>
<td>10%</td>
</tr>
<tr>
<td>Branch</td>
<td>2</td>
<td>20%</td>
</tr>
</tbody>
</table>

How much faster would the machine be if:

1. A data cache reduced the average load time to 2 cycles?

$$\frac{2.2}{(0.5 + 2 \times 2 + 3 + 4)} = \frac{2.2}{1.6} = 1.375 \times$$

2. Branch prediction shaved a cycle off the branch time?

$$\frac{2.2}{2.2 - 1 \times 2} = \frac{2.2}{2.0} = 1.1 \times$$

3. Two ALU instructions could be executed at once?

$$\frac{2.2}{(0.25 \times 1.0 + 0.3 + 0.4)} = \frac{2.2}{1.95} = 1.13 \times$$
Warning 1: Amdahl’s Law

The impact of a performance improvement is limited by what is NOT improved:

\[
\frac{\text{Execution time after improvement}}{\text{Execution time of unaffected}} + \frac{\text{Execution time affected}}{\text{Amount of improvement}} = 1
\]

Example: Assume a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to speed up multiply to make the program run 4 times faster?

\[
\frac{100}{4} = 20 + 80 \times \frac{1}{\text{Improve}}
\]

\[
25 = 20 + 80 \times \frac{1}{\text{Improve}}
\]

\[
5 = \frac{80}{\text{Improve}} \quad \text{Improve} = \frac{80}{5} = 16
\]

5 times faster?

\[
\frac{100}{5} = 20 + 80 \times \frac{1}{\text{Improve}}
\]

\[
20 = 20 + 80 \times \frac{1}{\text{Improve}}
\]

\[
0 = \frac{80}{\text{Improve}} \quad \text{Improve} = \infty
\]
Warning 2: BIPs, GHz ≠ Performance

Higher MHz (clock rate) doesn’t always mean better CPU
Orange computer: 1000 MHz, CPI: 2.5, 1 billion instruction program

\[ \text{Exec} = 1 \times 10^9 \times 2.5 \times \frac{1}{1 \times 10^9} = 2.5 \text{ seconds} \]

Grape computer: 500MHz, CPI: 1.1, 1 billion instruction program

\[ \text{Exec} = 1 \times 10^9 \times 1.1 \times \frac{1}{0.5 \times 10^9} = \frac{1.1}{0.5} = 2.2 \text{ seconds} \]

Higher MIPS (million instructions per second) doesn’t always mean better CPU
1 GHz machine, with two different compilers
Compiler A on program X: 10 Billion ALU, 1 Billion Load
Compiler B on program X: 5 Billion ALU, 1 Billion Load

Execution Time: A 15  B 10

A: \( (10B \times 1 + 1B \times 5) \times \frac{1}{1B} = 15 \text{ sec} \)
B: \( (5B \times 1 + (B \times 5)) \times \frac{1}{1B} = 10 \text{ sec} \)

MIPS: A 733  B 600

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Type Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>1</td>
</tr>
<tr>
<td>Load</td>
<td>5</td>
</tr>
<tr>
<td>Store</td>
<td>3</td>
</tr>
<tr>
<td>Branch</td>
<td>2</td>
</tr>
</tbody>
</table>
Processor Performance Summary

Machine performance:

\[ \text{CPU execution time for a program} = \frac{\text{Instructions for a program} \cdot \text{CPI} \cdot \frac{1}{\text{Clock rate}}}{\text{CISC}} \quad \text{RISC} \]

Better performance:

- Number of instructions to implement computations: \(\downarrow\)
- CPI: \(\downarrow\) \(\uparrow\)
- Clock rate: \(\uparrow\) \(\downarrow\)

Improving performance must balance each constraint

Example: RISC vs. CISC

CISC: Complex Instruction Set Computers
RISC: Reduced Instruction Set Computers
Datapath & Control

Readings: 4.1-4.4

Datapath: System for performing operations on data, plus memory access.

Control: Control the datapath in response to instructions.
Simple CPU

Develop complete CPU for subset of instruction set

Memory: LDUR, STUR

Branch: B

Conditional Branch: CBZ

Arithmetic: ADD, SUB

Most other instructions similar
Execution Cycle

1. **Instruction Fetch**
   - Obtain instruction from program storage

2. **Instruction Decode**
   - Determine required actions and instruction size

3. **Operand Fetch**
   - Locate and obtain operand data

4. **Execute**
   - Compute result value or status

5. **Result Store**
   - Deposit results in storage for later use

6. **Next Instruction**
   - Determine successor instruction
Processor Overview

Overall Dataflow
PC fetches instructions
Instructions select operand registers, ALU immediate values
ALU computes values
Load/Store addresses computed in ALU
Result goes to register file or Data memory