Lab 1 Solution: Performance

CS 350: Computer Organization & Assembler Language Programming
Lab due Sat Feb 2

Problems

1. Exercise 1.2 a-c. (Eight great ideas.)
   (a) Performance via Pipelining,
   (b) Dependability via Redundancy,
   (c) Performance via Prediction

2. Exercise 1.4. (Frame buffer)
   2a. Bytes/frame = pixels/frame * colors/pixel * bytes/color
       = (1280 * 1024) * 3 * 1 = 3,932,160 bytes / frame
   2b. sec/frame = frame size/network speed
       = 3,932,160 bytes / frame / (100 Mb/s * 1 byte/8 bits)
       = 3,932,160 / (100 × 10^6/8) s
       = 0.315 s

3. Exercise 1.6. (Class and Global CPI)
   3a. For each processor, performance = instructions/s = clock rate/CPI
       P1: 3 × 10^9 / 1.5 = 2 × 10^9 instr/s
       P2: 2.5 × 10^9 / 1.0 = 2.5 × 10^9 [highest performance of P1, P2, P3]
       P3: 4 × 10^9 / 2.2 = 1.818 × 10^9
   3b. # instructions = instructions/s × runtime
       P1: 2 × 10^9 instr/s * 10 s = 20 × 10^9 instr
       P2: 2.5 × 10^9 * 10 = 25 × 10^9
       P3: 1.8181 × 10^9 * 10 = 18.18 × 10^9
   # clock cycles = clock rate × runtime
       P1: 3 × 10^9 cycles/s * 10 s = 30 × 10^9 cycles
       P2: 2.5 × 10^9 * 10 = 25 × 10^9
       P3: 4 × 10^9 * 1 = 40 × 10^9
   3c. new clock rate = new # cycles / new runtime.
       The new runtime is 30% less than 10 s, so 7 s. The new CPIs are 1.2 × the old CPIs.
       The new # clock cycles = new CPI × # instructions.
       P1: (1.2 * 1.5) CPI * 20 × 10^9 instr / 7s = 5.14 GHz clock
P2: \((1.2 \times 1.0) \times 25 \times 10^9 / 7 = 4.29 \text{ GHz}\)

P3: \((1.2 \times 2.2) \times 18.18 \times 10^9 / 7 = 6.86 \text{ GHz}\)

4. Exercise 1.7. (Compiler affects CPI)

4a. In general, \(CPI = C \times T / N\) where \(C\) is the clock cycle rate, \(T\) is the execution time, and \(N\) is the count of instructions executed.

Processor A: \((1/1 \text{ ns}) \times 1.1 \text{ s} / (1.0 \times 10^9) = 1.1 \text{ CPI}\)
Processor B: \((1/1 \text{ ns}) \times 1.5 \text{ s} / (1.2 \times 10^9) = 1.25 \text{ CPI}\)

4b. Since \(CPI = C \times T / N\), we have \(C = CPI \times N / T\).
Processor A: \(C_A = 1.1 \times 1.0 \times 10^9 / T\).
Processor B: \(C_B = 1.25 \times 1.2 \times 10^9 / T\).
Relative clock speed = \(C_B / C_A = (1.25 \times 1.2) / (1.1 \times 1.0) = 1.373\)

4c. Since \(CPI = C \times T / N\), we have \(T = CPI \times N / C\).
With the new compiler, \(T = 1.1 \times 6.0 \times 10^8 / C = 6.6 \times 10^8 / C\).
Ratio \(T_A / T_{new} = (1.1 \times 10^9 / C_A) / (6.6 \times 10^8 / C_A) = 11 / 6.6 = 1.677\)
Ratio \(T_B / T_{new} = (1.25 \times 1.2 \times 10^9 / C_B) / (6.6 \times 10^8 / C_A) = 2.272\)

5. Exercise 1.9.1. (# Processors affects Execution Time)

Here's a table with all the calculations (you didn't have to go into all this detail, but it would have helped for partial credit). We had a clock rate of 2 GHz.

<table>
<thead>
<tr>
<th>(p)</th>
<th>(N_A)</th>
<th>(CPI_A)</th>
<th>(C_A)</th>
<th>(N_L)</th>
<th>(CPI_L)</th>
<th>(C_L)</th>
<th>(N_B)</th>
<th>(CPI_B)</th>
<th>(C_B)</th>
<th>(C)</th>
<th>(Exec)</th>
<th>(Speedup)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.56</td>
<td>1</td>
<td>2.56</td>
<td>12</td>
<td>15.36</td>
<td>0.256</td>
<td>5</td>
<td>1.28</td>
<td>19.20</td>
<td>9.6</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.83</td>
<td>1</td>
<td>1.83</td>
<td>12</td>
<td>10.97</td>
<td>0.256</td>
<td>5</td>
<td>1.28</td>
<td>14.08</td>
<td>7.04</td>
<td>1.36</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.91</td>
<td>1</td>
<td>0.91</td>
<td>12</td>
<td>5.49</td>
<td>0.256</td>
<td>5</td>
<td>1.28</td>
<td>7.68</td>
<td>3.84</td>
<td>2.50</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.46</td>
<td>1</td>
<td>0.46</td>
<td>12</td>
<td>2.74</td>
<td>0.256</td>
<td>5</td>
<td>1.28</td>
<td>4.48</td>
<td>2.24</td>
<td>4.29</td>
<td></td>
</tr>
</tbody>
</table>

**Key:** (The \(N\)... and \(C\)... columns are in units of \(10^9\))

- \(p\) = number of processors
- \(A, L, B\) are the instruction categories (Arith, Load/Store, and Branch)
- \(N_A, N_L, N_B\) are the number of instructions by category
  - We're given \(N_A = 2.56 / (0.7 \ p), N_L = 1.28 / (0.7 \ p),\) and \(N_B = 0.256.\)
- \(CPI_A, CPI_L, CPI_B\) are the CPI rates, by category (given as 1, 12, and 5)
- \(C_A, C_L, C_B\) are the number of clock cycles by category. \((C_A = N_A \times CPI_A, \text{etc.})\)
- \(C = C_A + C_L + C_B\) is the total number of clock cycles.
- \(Exec\) is execution time in seconds = \(C / 2 \text{ GHz}\) clock rate
- \(Speedup\) is execution time relative to 1 processor
6. Exercise 1.9.2. (CPI affects Execution Time)
Doubling the arithmetic CPI will double number of cycles for arithmetic instructions, so \(\text{new } C_A = 2 \text{ old } C_A\). The extra cycles will increase the total number of clock cycles: \(\text{new } C = \text{old } C + (\text{new } C_A - \text{old } C_A)\), which in turn changes the execution time and relative speedup:

<table>
<thead>
<tr>
<th></th>
<th>Old C_A</th>
<th>Old C</th>
<th>Old Exec</th>
<th>Old Speedup</th>
<th>New C_A</th>
<th>New C</th>
<th>New Exec</th>
<th>New Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.56</td>
<td>19.20</td>
<td>9.6</td>
<td>1.00</td>
<td>5.12</td>
<td>24.32</td>
<td>12.16</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>1.83</td>
<td>14.08</td>
<td>7.04</td>
<td>1.36</td>
<td>3.66</td>
<td>17.74</td>
<td>8.87</td>
<td>1.37</td>
</tr>
<tr>
<td>4</td>
<td>0.91</td>
<td>7.68</td>
<td>3.84</td>
<td>2.50</td>
<td>1.82</td>
<td>9.5</td>
<td>4.75</td>
<td>2.56</td>
</tr>
<tr>
<td>8</td>
<td>0.46</td>
<td>4.48</td>
<td>2.24</td>
<td>4.29</td>
<td>0.92</td>
<td>5.4</td>
<td>2.7</td>
<td>4.50</td>
</tr>
</tbody>
</table>

7. Exercise 1.9.3. (Change CPI of Load/Store to reduce Execution Time)
To decrease the execution time for 1 processor \((C = 19.20 \text{ G cycles})\) to match the execution time for 4 processors \((C = 7.68)\), we need to lower \(C_L\) from 15.36 to \((15.36 - (19.20 - 7.68)) = 15.36 - 11.52 = 3.84\). So new \(CPI_L = (\text{new } C_L) / N_L = 3.84 / 1.28 = 3.0\).

8. Exercise 1.14.2. (Change CPI to effect Execution Time)
We're given a program that uses (in units of \(10^6\))

- 50 FP instructions \(\times \text{CPI} = 1\) = 50 cycles
- 110 INT instructions \(\times \text{CPI} = 1\) = 110 cycles
- 16 BR instructions \(\times \text{CPI} = 2\) = 32 cycles

\(= \) total 192 cycles for non-L/S instructions

plus 80 L/S instructions \(\times \text{CPI} = 4\) = 320 cycles
\(=\) 512 cycles total, if L/S CPI = 4

To run the program twice as fast, we need half the number of cycles: \(512 / 2 = 256\). So the new L/S cycles = 256 – the 192 from non-L/S instructions = 64, and the new L/S CPI = 64 M cycles / 80 M L/S instructions = 0.80. (They have to be 100 times as fast!)