CMSC 411 Computer Architecture

Metrics and Units Worksheet
Solutions

September 6, 2002
Dr. Hugue

Unitizing Things

1. MIPS and MFLOPS are not identical measures, even though they both measure millions of operations per second. Explain why not.

MIPS is \textit{million instructions per second} while MFLOPS is \textit{million floating-point operations}. Since their counting targets are different, they are not identical measures.

2. What is the clock cycle time of a system with a 1.2MHz clock?

\textbf{Answer:} Since clock cycle time = 1/clock rate, the clock cycle time is $1/(1.2 \times 10^9) = 0.83$ns. \textit{That's right, less than a nanosecond faster than light can move a meter. Whew!}

3. Explain why an increase in performance is good, while an increase in execution time is bad, and vice versa.

\textbf{Answer:} Execution time is defined as the time between the start and the completion of an event (textbook P.18). An increase in execution time is bad because this means it takes more time to finish a job.

Performance is the reciprocal of execution time (textbook P.18). Thus, an increase in performance is good because this means an decrease in execution time.

For example, increasing the clock rate (the number of clock cycles per second) is clearly good because you have more “pulses” or opportunities for the computer, a finite state machine, to change state each second. However, this increase in clock rate could be accomplished by \textit{decreasing} the clock cycle time (the number of seconds between consecutive clock cycles). Thus, you need to be careful with word problems that discuss “improving” certain features of an architecture.

4. Express CPU performance in terms of clocks per instruction, number of instructions, and clock rate.

\textbf{Answer:} From the CPU performance equation, we have

$$\text{CPU}_{\text{perf}} = \frac{\text{CR}}{\text{IC} \times \text{CPI}}$$

Speeding it Up

1. Machine A is modified to produce Machine B, where the clock cycle time of machine B is 15% more than that of machine A; the average number of clocks per instruction in machine B is 60% of the average number of clocks per instruction in machine A. The instruction counts do not change.

What is the speedup of the modification, if any? Is it reasonable to assume that the instruction counts don’t change? That is, under what conditions is it possible to get the type of modification presented in this problem.

\textbf{Answer:} Let CCT, CPI and IC denote clock cycle time, cycles per instruction and instruction count respectively. We have

$$\text{CCT}_B = 1.15 \times \text{CCT}_A$$
\[ CPI_B = 0.6 CPI_A \]
\[ IC_B = IC_A \]
\[ \text{Speedup} = \frac{\text{Execution Time of A}}{\text{Execution Time of B}} \]
\[ = \frac{(IC_A \times CPI_A \times CCT_A)}{(IC_B \times CPI_B \times CCT_B)} \]
\[ = \frac{1}{(0.6 \times 1.15)} = 1.45 \]

It is not reasonable to assume the instruction counts don’t change. After a machine is modified, the machine may use different instruction set with different instruction counts to finish the same job.

2. A program spends 75% of its time in iterative computations. A more robust iteration process is used, which speeds up the computations by a factor of 10. What is the effective speedup?

**Answer:** Put the values into the Amdahl’s law equation (textbook P.30):

\[ \text{Speedup} = \frac{1}{(1 - FE) + \frac{FE}{K}} \]

\[ \text{Speedup} = \frac{1}{((1-0.75)+(0.75/10))} = 3.077. \]

3. A process takes 10 seconds to execute, 8 seconds of which are spent in the square-root routine. Suppose a more efficient square-root program reduces the time spent performing the square-root by 90%.

What is the effective speedup? What percentage of the new execution time is spent using the new square-root process?

If I wanted to achieve an effective speedup of 4, how much faster would the new square-root process need to be.

**Answer:** Before enhancement, the program spends 8/10=80% of time in the square-root routine. After the enhancement, it takes 8*10% = 0.8s to do the square-root calculation. Thus, the enhanced speed up = 8/0.8 = 10. Put the values into the equation (textbook P.30). Speedup = 1/((1-0.8) + (0.8/10)) = 3.57.

In order to obtain an effective speedup of 4, 4 = 1/((1-0.8)+(0.8/Speedup\text{enhanced})). Speedup\text{enhanced} = 16.

4. A program spends 20% of the new execution time using an enhancement that speeds up the process by a factor of 10. What is the effective speedup? What percentage of time was spent in the enhanced process during the unenhanced execution.

**Answer:** Let \( T_{old} \) and \( T_{new} \) be the execution time of the program before and after enhancement respectively. We cannot apply the equation (textbook P.30) directly. But, by its principle, we have \( T_{old} = T_{new} \times ((1-0.2) + (0.2/0.1)) = 2.8T_{new} \). Thus, the effective speedup = \( T_{old}/T_{new} = 2.8 \).

During unenhanced execution, the program spends 0.2\( T_{new} \times 10 = 2T_{new} \) time for the enhanced process. The corresponding fraction = \( 2T_{new}/T_{old} = 2/2.8 = 71.43\% \).

The Performing CPU

1. Suppose that the instruction frequencies and clock cycle counts for a an ISA are as shown in the table below:

<table>
<thead>
<tr>
<th>Ins Type</th>
<th>Freq</th>
<th>CCt</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>.44</td>
<td>1</td>
</tr>
<tr>
<td>Load</td>
<td>.21</td>
<td>2</td>
</tr>
<tr>
<td>Store</td>
<td>.12</td>
<td>2</td>
</tr>
<tr>
<td>Branch</td>
<td>.23</td>
<td>2</td>
</tr>
</tbody>
</table>
A proposed modification to the ISA would replace one fourth of the ALU instructions, that are paired with a loaded operand that is never used again, with a new ALU2 instruction that incorporates the previously loaded operand in the instruction word. While the addition of this ALU2 instruction decreases the number of load instructions, the number of clock cycles for a branch is increased to three. The ALU2 instruction takes one clock cycle. So, the designers plan to use an optimizing compiler that will discard a third of the remaining ALU instructions.

Will this modified protocol improve the average execution time? Justify your answer using a quantitative approach, with the following table, to help you.

<table>
<thead>
<tr>
<th>Ins Type</th>
<th>Freq</th>
<th>CCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>.22</td>
<td>1</td>
</tr>
<tr>
<td>ALU2</td>
<td>.11</td>
<td>1</td>
</tr>
<tr>
<td>Load</td>
<td>.10</td>
<td>2</td>
</tr>
<tr>
<td>Store</td>
<td>.12</td>
<td>2</td>
</tr>
<tr>
<td>Branch</td>
<td>.23</td>
<td>3</td>
</tr>
</tbody>
</table>

Incorrect Answer:

Let CCT be the clock cycle time. Assume that it remains the same after the modification. Let IC be the instruction counts of the original system.

\[
CPU_{\text{old}} = (1 \times 0.44 + 2 \times 0.21 + 2 \times 0.12 + 2 \times 0.23) \times IC \times CCT = 1.56 \times IC \times CCT \\
CPU_{\text{new}} = (1 \times 0.22 + 1 \times 0.11 + 2 \times 0.10 + 2 \times 0.12 + 3 \times 0.23) \times IC \times CCT = 1.46 \times IC \times CCT
\]

Speedup = \(CPU_{\text{old}}/CPU_{\text{new}} = 1.56/1.46 = 1.07\)

**Note**: This answer is numerically correct, but textually incorrect. Why? What's wrong with it.\(^1\)

2. Two machines use the same number of instructions to execute a given set of programs, and have the same clock cycle times. Must their performance be the same, or can the workload execution time differ on the two machines. Give a reasonable explanation to justify your conclusions. (This is a thought question. There are lots of right answers here. Think of at least two.)

**Answer**: No, it is possible that their performance is not the same.

- Although they have the same total number of instructions, they may use different number of instructions for different kind of instructions. Then, the total number of clock cycles may be different and so the performance.

- The performance of a machine is also affected by many other factors such as cache size and I/O speed.

3. A modification to a computer system and its operations decreases the number of stores by 50%, increases the number of ALU operations by 20%, and increases the number of loads by 20%. The original system took 1 clock cycle per ALU instruction, and 2 clock cycles for all other instructions. In the new system, stores take 3, branches now take 2 clock cycles and loads and ALUs only take 1 clock cycle.

**Answer**: Evaluate the effect of the modification quantitatively.

Let CCT be the clock cycle time. Assume that it remains the same after the modification. Let \(IC_{\text{STORE}}, IC_{\text{ALU}}, IC_{\text{LOAD}}\) and \(IC_{\text{BR}}\) be the instruction counts of the STORE, ALU, LOAD and BRANCH operations in the original system.

Speed up = \((CCT \times (IC_{\text{STORE}} \times 2 + IC_{\text{ALU}} \times 1 + IC_{\text{LOAD}} \times 2 + IC_{\text{BR}} \times 2))/(CCT \times (0.5IC_{\text{STORE}} \times 3 + 1.2IC_{\text{ALU}} \times 1 + 1.2IC_{\text{LOAD}} \times 1 + IC_{\text{BR}} \times 2)) = (2IC_{\text{STORE}} + IC_{\text{ALU}} + 2IC_{\text{LOAD}} + 2IC_{\text{BR}})/(1.5IC_{\text{STORE}} + 1.2IC_{\text{ALU}} + 1.2IC_{\text{LOAD}} + 2IC_{\text{BR}})\)

\(^1\)The proper answer is \(CPU_{\text{new}} = [(1 \times 0.22 + 1 \times 0.11 + 2 \times 0.10 + 2 \times 0.12 + 3 \times 0.23)/(0.78IC) \times CCT = 1.46 \times IC \times CCT\), because the adjusted “frequencies” no longer add up to 1.00, and because the new IC is only 0.78 of the original.