EE 4720 **Homework 1 Solution** **Assigned: Spring 1997**

Problem 1: (2 pts) Just plug the run times into these equations

$$\text{AM} = \frac{1}{n}\sum_{i=1}^{n} t_i \qquad \text{HM} = \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{t_i}\right)^{-1} \qquad \text{GM} = \sqrt[n]{\prod_{i=1}^{n} t_i}$$

to obtain 42.6, 13.8, and 27.7 for the arithmetic, harmonic, and geometric mean (respectively) of the program run times on the base machine and 18.6, 4.8, 11.7 for the arithmetic, harmonic, and geometric mean (respectively) of the program run times on the test machine.

Problem 2: (3 pts) The key phrase in the problem is, "each type of program is of equal importance." This means that, say, if machine A runs the compilers, A1, A2, and A3, 10% faster, and machine B runs the databases 10% faster, both would have the same TigerMark rating (assuming they ran the other programs as fast as the base machine). A common, and incorrect, solution was taking the geometric mean of the speedups of each program. That is,

$$\left(\frac{t_{\text{A1}}(\text{Base})}{t_{\text{A1}}(\text{Test})} \times \frac{t_{\text{A2}}(\text{Base})}{t_{\text{A2}}(\text{Test})} \times \frac{t_{\text{A3}}(\text{Base})}{t_{\text{A3}}(\text{Test})} \times \frac{t_{\text{B1}}(\text{Base})}{t_{\text{B1}}(\text{Test})} \times \frac{t_{\text{B2}}(\text{Base})}{t_{\text{B2}}(\text{Test})} \times \frac{t_{\text{C1}}(\text{Base})}{t_{\text{C1}}(\text{Test})} \times \frac{t_{\text{C2}}(\text{Base})}{t_{\text{C2}}(\text{Test})} \times \frac{t_{\text{C3}}(\text{Base})}{t_{\text{C3}}(\text{Test})} \times \frac{t_{\text{C4}}(\text{Base})}{t_{\text{C4}}(\text{Test})}\right)^{1/9}.$$

Because the number of programs of each type is different a 10% change in each program of one type will have a different impact than a 10% change in each program of another type, which is not acceptable in this case.

In one correct solution, the average speedup of programs of each type is computed, yielding three speedups. These three speedups are averaged to get the TigerMark. Symbolically,

$$\text{TM}(\text{Test}) = \frac{1}{3}\left(\frac{1}{3}\left(\frac{t_{\text{A1}}(\text{Base})}{t_{\text{A1}}(\text{Test})} + \frac{t_{\text{A2}}(\text{Base})}{t_{\text{A2}}(\text{Test})} + \frac{t_{\text{A3}}(\text{Base})}{t_{\text{A3}}(\text{Test})}\right) + \frac{1}{2}\left(\frac{t_{\text{B1}}(\text{Base})}{t_{\text{B1}}(\text{Test})} + \frac{t_{\text{B2}}(\text{Base})}{t_{\text{B2}}(\text{Test})}\right) + \frac{1}{4}\left(\frac{t_{\text{C1}}(\text{Base})}{t_{\text{C1}}(\text{Test})} + \frac{t_{\text{C2}}(\text{Base})}{t_{\text{C2}}(\text{Test})} + \frac{t_{\text{C3}}(\text{Base})}{t_{\text{C3}}(\text{Test})} + \frac{t_{\text{C4}}(\text{Base})}{t_{\text{C4}}(\text{Test})}\right)\right).$$

Problem 3: Another possible benefit is code density. That is, it's quite likely that the space needed for the single new instruction is less than the five instructions it replaces, so less space is needed to store the program.

One drawback is that the benefit does not justify the cost. The new instruction may only be rarely used while the hardware cost might be substantial.

Another drawback is that it might be difficult to quickly execute the new instruction on future implementations. The instruction might do 100% of what the programmer wants and 10% more. The 10% more might preclude a faster future implementation. A sequence of simpler instructions might do exactly what the programmer wants and could be executed quickly.

Wrong answers:

(Benefit) Lower instruction count means lower execution time. This is wrong because it implies that fewer instructions will always lead to improved performance. (In this case it does.)

(Drawback) Lower instruction execution rate (MIPs). This is wrong because lower or higher MIPs does not mean lower or higher performance in general, and in this case.

(Drawback) Higher CPI. This is wrong because lower or higher CPI does not mean lower or higher performance in general, and in this case. Note that MIPs $= 10^6/\text{CPI}$.

Problem 4: (3 pts) For implementation A and compiler I, average instruction execution time is $2.625\,\text{CPI} = 2.625\,\mu\text{s}$ (either answer is acceptable). Total execution time is 21.0 ms. For implementation A and compiler II, average instruction execution time is $2.595\,\text{CPI} = 2.595\,\mu\text{s}$. Total execution time is 21.8 ms. In these cases compiler II had the lower CPI (good) but the higher execution time (bad), and so CPI is not a good predictor.

For implementation B and compiler I, average instruction execution time is $2.625\,\text{CPI} = 2.625\,\mu\text{s}$ (either answer is acceptable). Total execution time is 21.0 ms. For implementation B and compiler II, average instruction execution time is $2.405\,\text{CPI} = 2.405\,\mu\text{s}$. Total execution time is 20.2 ms. In these cases compiler II had the lower CPI *and* the lower execution time. CPI does agree with execution time here.

Wrong answer explained: $\text{CPI} = (2+2+3)/3 = (3+1+3)/3 = 7/3$ is wrong because it gives equal weight to all instruction categories. Average instruction execution time (CPI) is based on the mix of instructions actually executed, so frequently executed instructions should be counted more than infrequent ones.