Solutions for Chapter 4 Exercises

4.1 For P1, M2 is 4/3 (2 sec/1.5 sec) times as fast as M1. For P2, M1 is 2 times as fast (10 sec/5 sec) as M2.

4.2 We know the number of instructions and the total time to execute the program. The execution rate for each machine is simply the ratio of the two values. Thus, the instructions per second for P1 on M1 is (5×10^9 instructions/2 seconds) = 2.5×10^9 IPS, and the instructions for P1 on M2 is (6×10^9 instructions/1.5 seconds) = 4×10^9 IPS.

4.3 M2 runs 4/3 as fast as M1, but it costs 8/5 as much. As 8/5 is more than 4/3, M1 has the better value.

4.6 Running P1 1600 times on M1 and M2 requires 3200 and 2400 seconds respectively. This leaves 400 seconds left for M1 and 1200 seconds left for M2. In that time M1 can run (400 seconds/(5 seconds/iteration)) = 80 iterations of P2. M2 can run (1200 seconds/(10 seconds/iteration)) = 120 iterations. Thus M2 performs better on this workload.

Looking at cost-effectiveness, we see it costs ($500/(80 iterations/hour)) = $6.25 per (iteration/hour) for M1, while it costs ($800/(120 iterations/hour)) = $6.67 per (iteration/hour) for M2. Thus M1 is most cost-effective.

4.7

a. Time = (Seconds/cycle) * (Cycles/instruction) * (Number of instructions)

   Therefore the expected CPU time is (1 second/5×10^9 cycles) * (0.8 cycles/instruction) * (7.5×10^9 instructions) = 1.2 seconds

b. P received 1.2 seconds/3 seconds or 40% of the total CPU time.

4.8 The ideal instruction sequence for P1 is one composed entirely of instructions from class A (which have CPI of 1). So M1's peak performance is (4×10^9 cycles/second)/(1 cycle/instruction) = 4000 MIPS.

Similarly, the ideal sequence for M2 contains only instructions from A, B, and C (which all have a CPI of 2). So M2's peak performance is (6×10^9 cycles/second)/(2 cycles/instruction) = 3000 MIPS.

4.9 The average CPI of P1 is $(1 \times 2 + 2 + 3 + 4 + 3)/6 = 7/3$. The average CPI of P2 is $(2 \times 2 + 2 + 2 + 4 + 4)/6 = 8/3$. P2 then is (($6 \times 10^9$ cycles/second)/(8/3 cycles/instruction))/((4×10^9 cycles/second)/(7/3 cycles/instruction)) = 21/16 times faster than P1.

4.10 Using C1, the average CPI for I1 is (.4 * 2 + .4 * 3 + .2 * 5) = 3, and the average CPI for I2 is (.4 * 1 + .2 * 2 + .4 * 2) = 1.6. Thus, with C1, I1 is ((6×10^9 cycles/second)/(3 cycles/instruction))/((3×10^9 cycles/second)/(1.6 cycles/instruction)) = 16/15 times as fast as I2.

Using C2, the average CPI for I2 is (.4 * 2 + .2 * 3 + .4 * 5) = 3.4, and the average CPI for I2 is (.4 * 1 + .4 * 2 + .2 * 2) = 1.6. So with C2, I2 is faster than I1 by factor of $((3 \times 10^9$ cycles/second)/(1.6 cycles/instruction))/((6×10^9 cycles/second)/(3.4 cycles/instruction)) = 17/16.

For the rest of the questions, it will be necessary to have the CPIs of I1 and I2 on programs compiled by C3. For I1, C3 produces programs with CPI (.6 * 2 + .15 * 3 + .25 * 5) = 2.9. I2 has CPI (.6 * 1 + .15 * 2 + .25 * 2) = 1.4.

The best compiler for each machine is the one which produces programs with the lowest average CPI. Thus, if you purchased either I1 or I2, you would use C3.

Then performance of I1 in comparison to I2 using their optimal compiler (C3) is ((6×10^9 cycles/second)/(2.9 cycles/instruction))/((3×10^9 cycles/second)/(1.4 cycles/instruction)) = 28/29. Thus, I2 has better performance and is the one you should purchase.

4.11 Program P running on machine M takes (10^9 cycles/seconds) * 10 seconds = 10^{10} cycles. P′ takes (10^9 cycles/seconds) * 9 seconds = 9×10^9 cycles. This leaves 10^9 less cycles after the optimization.

Everytime we replace a mult with two adds, it takes 4 – 2 * 1 = 2 cycles less per replacement.

Thus, there must have been 10^9 cycles /(2 cycles/replacement) = 5×10^8 replacements to make P into P′.

4.12 The first option reduces the number of instructions to 80%, but increases the time to 120%. Thus it will take: 0.8 * 1.2 = 0.96 as much time as the initial case.

The second option removes 20%/2 = 10% of the instructions and increases the time taken to 110%. Therefore it will take 0.9 * 1.1 = 0.99 times as much time as the initial case.

Therefore, the first option is the faster of the two, and it is faster than the orginial, so you should have hardware automatically do garbage collection.

4.13 Let I = number of instructions in program and C = number of cycles in program. The six subsets are {clock rate, C} {cycle time, C} {MIPS, I} {CPI, C, MIPS} {CPI, I, clock rate} {CPI, I, cycle time}. Note that in every case each subset has to have at least one rate {CPI, clock rate, cycle time, MIPS} and one absolute {C, I}.

4.14 The total execution time for the machines are as follows:

Computer A = 2 + 20 + 200 seconds = 222 seconds

Computer B = 5 + 20 + 50 seconds = 75 seconds

Computer C = 10 + 20 + 15 seconds = 45 seconds

Thus computer C is faster. It is 75/45 = 5/3 times faster than computer B and 222/45 = 74/15 times faster than computer A.

4.15 With the new weighting the total execution time for the group of programs is:

Computer A = 8 * 2 + 2 * 20 + 1 * 200 seconds = 256 seconds

Computer B = 8 * 5 + 2 * 20 + 1 * 50 seconds = 130 seconds

Computer C = 8 * 10 + 2 * 20 + 1 * 15 seconds = 135 seconds

So with this workload, computer B is faster by a factor of 135/130 = 1.04 with respect to computer C and a factor of 256/130 = 1.97 with respect to computer A. This new weighting reflects a bias from the previous results by a bias toward program 1 and program 2, which resulted in computer A and computer B looking comparitively better than before.

4.16 Comparing the times of the program executions on computer A, we see that to get an equal amount of execution time, we will have to run program 1 100 times, program 2 10 times, and Program 3 1 time. This results in the following execution times:

Computer A = 100 * 2 + 10 * 20 + 1 * 200 seconds = 600 seconds

Computer B = 100 * 5 + 10 * 20 + 1 * 50 seconds = 750 seconds

Computer C = 100 * 10 + 10 * 20 + 1 * 15 seconds = 1215 seconds

So computer A is fastest with this workload.

Using computer B's program execution times to determine a weighting, we get a ratio of 20:5:2 for program 1, program 2, and program 3, respectively. This results in the following execution times:

Computer A = 20 * 2 + 5 * 20 + 2 * 200 seconds = 540 seconds

Computer B = 20 * 5 + 5 * 20 + 2 * 50 seconds = 300 seconds

Computer C = 20 * 10 + 5 * 20 + 2 * 15 seconds = 330 seconds

So in this case, computer B is fastest.

Using the execution times on computer C , we get a 6:3:4 ratio, resulting in the following total execution times:

Computer A = 6 * 2 + 3 * 20 + 4 * 200 seconds = 872 seconds

Computer B = 6 * 5 + 3 * 20 + 4 * 50 seconds = 290 seconds

Computer C = 6 * 10 + 3 * 20 + 4 * 15 seconds = 180 seconds

So in this case computer C is fastest.

As we did the weighting to equalize the times on one particular machine, we ended up running programs that that computer could do the fastest most often and the programs that it was slower on less often. This will tend to be a comparative improvement in the execution time for the machine we are normalizing time to (as the weightings are not guaranteed to bias towards the programs that the other machines are better at). In this example, this type of weighting was enough to make that computer appear the fastest.

4.17 We know CPI is equal to (Cycles/second)/(Instructions/second). So the CPI of P1 on M1 is (4×10^9 cycles/second)/(2.5×10^9 instructions/second) = 1.6 CPI, and the CPI of P1 on M2 is (6×10^9 cycles/second)/(4×10^9 instructions/second) = 1.5 CPI.

4.18 We have the CPI, the clock rate, and the total execution time, and we're trying to find the total number of instructions. Using the following equation:

(Cycles/instruction)/(Cycles/second) * Instructions = (Execution time)

We find that there are (5 seconds) * (4×10^9 cycles/second)/(0.8 cycles/instruction) = 12.5×10^9 instructions in P2 on M1, and (10 seconds) * (6×10^9 cycles/second)/(1.5 CPI) = 40×10^9 instructions in P2 on M2.

4.19 No solution provided.

4.20 No solution provided.

4.21 No solution provided.

4.22 Using Amdahl's law (or just common sense), we can determine the following:

- Speedup if we improve only multiplication = 100/(30 + 50 + 20/4) = 100/85 = 1.18.
- Speedup if we only improve memory access = 100/(30 + 50/2 + 20)) = 100/75 = 1.33.
- Speedup if both improvements are made = 100/(30 + 50/2 + 20/4) = 100/60 = 1.67.

4.23 The problem is solved algebraically and results in the equation

$$100/(Y + (100 - X - Y) + X/4) = 100/(X + (100 - X - Y) + Y/2)$$

where X = multiplication percentage and Y = memory percentage. Solving, we get memory percentage = 1.5 × multiplication percentage. Many examples thus exist: for example, multiplication = 20%, memory = 30%, other = 50%, or multiplication = 30%, memory = 45%, other = 25%, and so on.

4.24 $\text{Speedup} = \dfrac{\text{Execution time before improvement}}{\text{Execution time after improvement}}$

Rewrite the execution time equation:

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

$$= \frac{\text{Execution time affected} + \text{Amount of improvement} \times \text{Execution time unaffected}}{\text{Amount of improvement}}$$

Rewrite execution time affected by improvement as execution time before improvement $\times f$, where f is the fraction affected. Similarly execution time unaffected.

$$= \frac{\text{Execution time before improvement} \times f}{\text{Amount of improvement}} + \text{Execution time before improvement} \times (1 - f)$$

$$= \left(\frac{f}{\text{Amount of improvement}} + (1 - f)\right) \times \text{Execution time before improvement}$$

$$\text{Speedup} = \frac{\text{Execution time before improvement}}{\left(\frac{f}{\text{Amount of improvement}} + (1 - f)\right) \times \text{Execution time before improvement}}$$

$$\text{Speedup} = \frac{1}{\left(\frac{f}{\text{Amount of improvement}} + (1 - f)\right)}$$

The denominator has two terms: the fraction improved (f) divided by the amount of the improvement and the fraction unimproved ($1 - f$).

4.25 We can just take the GM of the execution times and use the inverse.

$$\text{GM(A)} = \sqrt{1000} = 32\text{, GM(B)} = \sqrt{1000} = 32\text{, and GM(C)} = \sqrt{400} = 20,$$

so C is fastest.

4.26 A, B: B has the same performance as A. If we run program 2 once, how many times should we run program 1? $x + 1000 = 10x + 100$, or x = 100. So the mix is 99% program 1, 1% program 2.

B, C: C is faster by the ratio of $\frac{32}{20} = 1.6$.

Program 2 is run once, so we have $10x + 100 = 1.6 \times (20x + 20)$, $x = 3.1$ times. So the mix is 76% program 1 and 24% program 2.

A, C: C is also faster by 1.6 here. We use the same equation, but with the proper times: $x + 1000 = 1.6 \times (20x + 20)$, $x = 31.2$. So the mix is 97% program 1 and 3% program 2. Note that the mix is very different in each case!

4.27 No solution provided.

4.28 No solution provided.

4.29 No solution provided.

4.30

Program	Computer A	Computer B	Computer C
1	10	1	0.5
2	0.1	1	5

4.31 The harmonic mean of a set of rates,

$$\text{HM} = \frac{n}{\sum_{i=1}^{n} \frac{1}{\text{Rate}_i}} = \frac{n}{\sum_{i=1}^{n} \text{Time}_i} = \frac{1}{\frac{\sum_{i=1}^{n} \text{Time}_i}{n}} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} \text{Time}_i} = \frac{1}{\text{AM}}$$

where AM is the arithmetic mean of the corresponding execution times.

4.32 No solution provided.

4.33 The time of execution is (Number of instructions) * (CPI) * (Clock period). So the ratio of the times (the performance increase) is:

$$10.1 = \frac{(\text{Number of instructions})*(\text{CPI})*(\text{Clock period})}{(\text{Number of instructions w/opt.})*(\text{CPI w/opt.})*(\text{Clock period})}$$

$$= 1/(\text{Reduction in instruction count}) * (2.5 \text{ improvement in CPI})$$

Reduction in instruction count = .2475.

Thus the instruction count must have been reduced to 24.75% of the original.

4.34 We know that

(Number of instructions on V) * (CPI on V) * (Clock period)

$$5 = \frac{(\text{Time on V})}{(\text{Time on P})} = \frac{(\text{Number of instructions on V}) * (\text{CPI on V}) * (\text{Clock period})}{(\text{Number of instructions on P}) * (\text{CPI on P}) * (\text{Clock period})}$$

5 = (1/1.5) * (CPI of V)/(1.5 CPI)

CPI of V = 11.25.

4.45 The average CPI is .15 * 12 cycles/instruction + .85 * 4 cycles/instruction = 5.2 cycles/instructions, of which .15 * 12 = 1.8 cycles/instructions of that is due to multiplication instructions. This means that multiplications take up 1.8/5.2 = 34.6% of the CPU time.

4.46 Reducing the CPI of multiplication instructions results in a new average CPI of .15 * 8 + .85 * 4 = 4.6. The clock rate will reduce by a factor of 5/6 . So the new performance is (5.2/4.6) * (5/6) = 26/27.6 times as good as the original. So the modification is detrimental and should not be made.

4.47 No solution provided.

4.48 Benchmarking suites are only useful as long as they provide a good indicator of performance on a typical workload of a certain type. This can be made untrue if the typical workload changes. Additionally, it is possible that, given enough time, ways to optimize for benchmarks in the hardware or compiler may be found, which would reduce the meaningfulness of the benchmark results. In those cases changing the benchmarks is in order.

4.49 Let *T* be the number of seconds that the benchmark suite takes to run on Computer A. Then the benchmark takes 10 * *T* seconds to run on computer B. The new speed of A is (4/5 * *T* + 1/5 * (*T*/50)) = 0.804 *T* seconds. Then the performance improvement of the optimized benchmark suite on A over the benchmark suite on B is 10 * T/(0.804 *T*) = 12.4.

4.50 No solution provided.

4.51 No solution provided.

4.52 No solution provided.