Exercises

Each exercise has a difficulty rating in square brackets and a list of the chapter sections it depends on in angle brackets. See the Preface for a description of the difficulty scale. Solutions to the "starred" exercises appear in Appendix B.

1.1 [15/15/15/15] <1.3, 1.4, 7.2> Computer system designers must be alert to the rapid change of computer technology. To see one example of how radically change can affect design, consider the evolution of DRAM and magnetic disk technologies since publication of the first edition of this text in 1990. At that time DRAM density had been improving for 10 years at a rate of about 60% per year, giving rise every third year to a new generation of DRAM chips with four times more capacity than before. Magnetic disk data recording density had been improving for 30 years at nearly 30% per year, doubling every three years.

a. [15] <1.3> The first edition posed a question much like this. Assume that cost per megabyte for either type of storage is proportional to density, that 1990 is the start of the 4M bit DRAM generation, and that in 1990 DRAM costs 20 times more per megabyte than disk. Using the well-established historic density improvement rates, create a table showing projected relative cost of each DRAM generation and disk from 1990 for six generations. What conclusions can be drawn about the future of disk drives in computer designs and about the magnetic disk industry from this projection?

b. [15] <1.4, 7.2> The conclusion supported by the result from part (a) is far from today's reality. Shortly before 1990 the change from inductive heads to thin film, and then magnetoresistive heads, allowed magnetic disk recording density to begin a 60% annual improvement trend, matching DRAM. Since about 1997, giant magnetoresistive effect heads have upped the rate to 100% per year, and, available to the mass market in 2001, antiferromagnetically coupled recording media should support or improve that rate for several years. Using data from Figures 1.5 and 7.4, plot the actual ratio of DRAM to disk price per unit of storage for each DRAM generation (3-year intervals) starting in 1983. Compare your answer with part (a) by including those data points on the graph. Assume that DRAM storage is built from the then-available chip size with the lowest cost per bit and that disk cost is the median cost for that year. Note that 1 GB = 1000 MB. Ignore the cost of any packaging, support hardware, and control hardware needed to incorporate DRAM and disk into a computer system.

c. [15] <1.3> Not only price, but disk physical volume and mass improve with recording density. Today's standard laptop computer disk drive bay is 10 cm long and 7 cm wide. Assume that a 100 MB disk in 1990 occupied 500 cc (cubic centimeters) and massed 1000 g (grams). If disk volume and mass had improved only 30% per year since 1990, what would the height (neglect mechanical constraints on disk drive shape) and mass of a 30 GB laptop computer disk be today? For comparison, actual typical height and mass values for 2001 are 1.25 cm and 100 g.

d. [15] <1.3, 1.4> Increasing disk recording density expands the range of software applications possible at a given computer price point. High-quality desktop digital video editing capability is available in 2001 on a $1000 PC. Five minutes of digital video consumes about 1 GB of storage, so the 20 GB disk of the PC in Figure 1.9 provides reasonable capacity. If disk density had improved only at 30% per year since 1990, but other PC component costs shown in Figure 1.9 were unchanged and the ratio of retail price to component cost given in Figure 1.10 was unaffected, approximately how much more would a desktop video PC cost in 2001?

1.2 [20/10/10/10/15] <1.6> In this exercise, assume that we are considering enhancing a machine by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is 10 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode the *percentage of vectorization*.Vectors are discussed in Appendix G, but you don't need to know anything about how they work to answer this question!

a. [20] <1.6> Draw a graph that plots the speedup as a percentage of the computation performed in vector mode. Label the *y*-axis "Net speedup" and label the *x*-axis "Percent vectorization."

b. [10] <1.6> What percentage of vectorization is needed to achieve a speedup of 2?

c. [10] <1.6> What percentage of the computation run time is spent in vector mode if a speedup of 2 is achieved?

d. [10] <1.6> What percentage of vectorization is needed to achieve one-half the maximum speedup attainable from using vector mode?

e. [15] <1.6> Suppose you have measured the percentage of vectorization for programs to be 70%. The hardware design group says they can double the speed of the vector hardware with a significant additional engineering investment. You wonder whether the compiler crew could increase the use of vector mode as another approach to increasing performance. How much of an increase in the percentage of vectorization (relative to current usage) would you need to obtain the same performance gain as doubling vector hardware speed? Which investment would you recommend?

1.3 [15/10] <1.6> Assume—as in the Amdahl's Law example on page 41—that we make an enhancement to a computer that improves some mode of execution by a factor of 10. Enhanced mode is used 50% of the time, measured as a percentage of the execution time *when the enhanced mode is in use*. Recall that Amdahl's Law depends on the fraction of the original, *unenhanced* execution time that could make use of enhanced mode. Thus, we cannot directly use this 50% measurement to compute speedup with Amdahl's Law.

a. [15] <1.6> What is the speedup we have obtained from fast mode?

b. [10] <1.6> What percentage of the original execution time has been converted to fast mode?

✪ 1.4 [12/10/Discussion] <1.6> Amdahl's Law implies that the ultimate goal of high-performance computer system design should be an enhancement that offers arbitrarily large speedup for all of the task time. Perhaps surprisingly, this goal can be approached quite closely with real computers and tasks. Section 3.5 describes how some branch instructions can, with high likelihood, be executed in zero time with a hardware enhancement called a branch-target buffer. Arbitrarily large speedup can be achieved for complex computational tasks when more efficient algorithms are developed. A classic example from the field of digital signal processing is the discrete Fourier transform (DFT) and the more efficient fast Fourier transform (FFT). How these two transforms work is not important here. All we need to know is that they compute the same result, and with an input of n floating-point data values, a DFT algorithm will execute approximately n^2 floating-point instructions, while the FFT algorithm will execute approximately $n \log_2 n$ floating-point instructions.

a. [12] <1.6> Ignore instructions other than floating point. What is the speedup gained by using the FFT instead of the DFT for an input of $n = 2^k$ floating-point values in the range $8 \leq n \leq 1024$ and also in the limit as $n \to \infty$?

b. [10] <1.6> When $n = 1024$, what is the percentage reduction in the number of executed floating-point instructions when using the FFT rather than the DFT?

c. [Discussion] <1.6> Despite the speedup achieved by processors with a branch-target buffer, not only do processors without such a buffer remain in production, new processor designs without this enhancement are still developed. Yet, once the FFT became known, the DFT was abandoned. Certainly speedup is desirable. What reasons can you think of to explain this asymmetry in use of a hardware and a software enhancement, and what does your answer say about the economics of hardware and algorithm technologies?

1.5 [15] <1.6> Show that the problem statements in the examples on pages 42 and 44 describe identical situations and equivalent design alternatives.

✪ 1.6 [15] <1.9> Dhrystone is a well-known integer benchmark. Computer A is measured to perform D_A executions of the Dhrystone benchmark per second, and to achieve a millions of instructions per second rate of $MIPS_A$ while doing Dhrystone. Computer B is measured to perform D_B executions of the Dhrystone benchmark per second. What is the fallacy in calculating the MIPS rating of computer B as $MIPS_B = MIPS_A \times (D_B / D_A)$?

1.7 [15/15/8] <1.9> A certain benchmark contains 195,578 floating-point operations, with the details shown in Figure 1.33.

The benchmark was run on an embedded processor after compilation with optimization turned on. The embedded processor is based on a current RISC processor that includes floating-point function units, but the embedded processor does not include floating point for reasons of cost, power consumption, and lack of need for floating point by the target applications. The compiler allows floating-point instructions to be calculated with the hardware units or using software routines, depending on compiler flags. The benchmark took 1.08 seconds on the

Operation	Count
Add	82,014
Subtract	8,229
Multiply	73,220
Divide	21,399
Convert integer to FP	6,006
Compare	4,710
Total	195,578

Figure 1.33 Occurrences of floating-point operations.

RISC processor and 13.6 seconds using software on its embedded version. Assume that the CPI using the RISC processor was measured to be 10, while the CPI of the embedded version of the processor was measured to be 6.

a. [15] <1.9> What is the total number of instructions executed for both runs?

b. [15] <1.9> What is the MIPS rating for both runs?

c. [8] <1.9> On the average, how many integer instructions does it take to perform a floating-point operation in software?

1.8 [15/10/15/15/15] <1.3, 1.4> This exercise estimates the complete packaged cost of a microprocessor using the die cost equation and adding in packaging and testing costs. We begin with a short description of testing cost and follow with a discussion of packaging issues.

Testing is the second term of the chip cost equation:

$$\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging}}{\text{Final test yield}}$$

Testing costs are determined by three components:

$$\text{Cost of testing die} = \frac{\text{Cost of testing per hour} \times \text{Average die test time}}{\text{Die yield}}$$

Since bad dies are discarded, die yield is in the denominator in the equation—the good must shoulder the costs of testing those that fail. (In practice, a bad die may take less time to test, but this effect is small, since moving the probes on the die is a mechanical process that takes a large fraction of the time.) Testing costs about \$50 to \$500 per hour, depending on the tester needed. High-end designs with many high-speed pins require the more expensive testers. For higher-end microprocessors test time would run \$300 to \$500 per hour. Die tests take about 5 to 90 seconds on average, depending on the simplicity of the die and the provisions to reduce testing time included in the chip.

The cost of a package depends on the material used, the number of pins, and the die area. The cost of the material used in the package is in part determined by the

ability to dissipate heat generated by the die. For example, a *plastic quad flat pack* (PQFP) dissipating less than 1 W, with 208 or fewer pins, and containing a die up to 1 cm on a side costs $2 in 2001. A ceramic *pin grid array* (PGA) can handle 300 to 600 pins and a larger die with more power, but it costs $20 to $60. In addition to the cost of the package itself is the cost of the labor to place a die in the package and then bond the pads to the pins, which adds from a few cents to a dollar or two to the cost. Some good dies are typically lost in the assembly process, thereby further reducing yield. For simplicity we assume the final test yield is 1.0; in practice it is at least 0.95. We also ignore the cost of the final packaged test.

This exercise requires the information provided in Figure 1.34.

a. [15] <1.4> For each of the microprocessors in Figure 1.34, compute the number of good chips you would get per 20 cm wafer using the model on page 19. Assume a defect density of 0.5 defect per cm^2, a wafer yield of 95%, and $\alpha = 4$.

b. [10] <1.4> For each microprocessor in Figure 1.34, compute the cost per projected good die before packaging and testing. Use the number of good dies per wafer from part (a) of this exercise and the wafer cost from Figure 1.34.

c. [15] <1.3> Using the additional assumptions shown in Figure 1.35, compute the cost per good, tested, and packaged part using the costs per good die from part (b) of this exercise.

d. [15] <1.3> There are wide differences in defect densities between semiconductor manufacturers. Find the costs for the largest processor in Figure 1.34 (total cost including packaging), assuming defect densities are 0.3 per cm^2 and assuming that defect densities are 1.0 per cm^2.

e. [15] <1.3> The parameter α depends on the complexity of the process. Additional metal levels result in increased complexity. For example, α might be approximated by the number of interconnect levels. For the Digital 21064C with six levels of interconnect, estimate the cost of working, packaged, and tested die if $\alpha = 4$ and if $\alpha = 6$. Assume a defect density of 0.8 defects per cm^2.

Microprocessor	Die area (mm^2)	Pins	Technology	Estimated wafer cost ($)	Package
Alpha 21264C	115	524	CMOS, 0.18µ, 6M	4700	CLGA
Power3-II	163	1088	CMOS, 0.22µ, 6M	4000	SLC
Itanium	300	418	CMOS, 0.18µ, 6M	4900	PAC
MIPS R14000	204	527	CMOS, 0.25µ, 4M	3700	CPGA
UltraSPARC III	210	1368	CMOS, 0.15µ, 6M	5200	FC-LGA

Figure 1.34 Characteristics of microprocessors. About half of the pins are for power and ground connections. The technology entry is the process type, line width, and number of interconnect levels.

Package type	Pin count	Package cost ($)	Test time (secs)	Test cost per hour ($)
PAC	< 500	20	30	400
SLC	< 1100	20	20	420
Grid array (CLGA, CPGA, or FC-LGA)	< 500	20	20	400
Grid array (CLGA, CPGA, or FC-LGA)	< 1000	25	25	440
Grid array (CLGA, CPGA, or FC-LGA)	< 1500	30	30	480

Figure 1.35 Package and test characteristics.

1.9 [20/20] <1.4> On page 20 the concluding discussion about the die cost model claims that, for realistic die sizes and defect densities, die cost is better modeled as a function of (roughly) the die area squared rather than to the fourth power.

a. [20] <1.4> Using the model and a spreadsheet, determine the cost of dies ranging in area from 0.5 to 4 cm^2 and assuming a defect density of 0.6 and $\alpha = 4$. Next, use a mathematical analysis tool for fitting polynomial curves to fit the (die area, die cost) data pairs you computed in the spreadsheet. What is the lowest degree polynomial that is a close fit to the data?

b. [20] <1.4> Suppose defect densities were much higher: say, 2 defects per cm^2. Now what is lowest degree polynomial that is a close fit?

✪ 1.10 [15/15/10] <1.5, 1.9> Assume the two programs in Figure 1.15 each execute 100 million floating-point operations during execution on each of the three machines. If performance is expressed as a rate, then the average that tracks total execution time is the *harmonic mean*,

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

where Rate_i is a function of $1/\text{Time}_i$, the execution time for the ith of n programs in the workload.

a. [15] <1.5, 1.9> Calculate the MFLOPS rating of each program.

b. [15] <1.5, 1.9> Calculate the arithmetic, geometric, and harmonic means of MFLOPS for each machine.

c. [10] <1.5, 1.9> Which of the three means matches the relative performance of total execution time?

1.11 [12] <1.5> One reason people may incorrectly summarize rate data using an arithmetic mean is that it always gives an answer greater than or equal to the geometric mean. Show that for any two positive integers, a and b, the arithmetic

mean is always greater than or equal to the geometric mean. When are the two equal?

1.12 [12] <1.5> For reasons similar to those in Exercise 1.11, some people use arithmetic mean instead of harmonic mean (see the definition of harmonic mean in Exercise 1.10). Show that for any two positive rates, r and s, the arithmetic mean is always greater than or equal to the harmonic mean. When are the two equal?

✪ 1.13 [10/10/10/10] <1.5> Sometimes we have a set of computer performance measurements that range from very slow to very fast execution. A single statistic, such as a mean, may not capture a useful sense of the data set as a whole. For example, the CPU pipeline and hard disk subsystem of a computer execute their respective basic processing steps at speeds that differ by a factor of typically 10^7. This is a speed difference in excess of that between a jet airliner in cruising flight (~1000 kilometers per hour) and a snail gliding on the long, thin leaf of an agapanthus (perhaps 1 meter per hour). Let's look at what happens when measurements with such a large range are summarized by a single number.

a. [10] <1.5> What are the arithmetic means of two sets of benchmark measurements, one with nine values of 10^7 and one value of 1 and the other set with nine values of 1 and one value of 10^7? How do these means compare with the data set medians? Which outlying data point affects the arithmetic mean more, a large or a small value?

b. [10] <1.5> What are the harmonic means (see Exercise 1.10 for the definition of harmonic mean) of the two sets of measurements specified in part (a)? How do these means compare with the data set medians? Which outlying data point affects the harmonic mean more, a large or a small value?

c. [10] <1.5> Which mean, arithmetic or harmonic, produces a statistic closest to the median?

d. [10] <1.5> Repeat parts (a) and (b) for two sets of 10 benchmark measurements with the outlying value only a factor of 2 larger or smaller. How representative of the entire set do the arithmetic and harmonic mean statistics seem for this narrow range of performance values?

1.14 [15/15] <1.5> A spreadsheet is useful for performing the computations of this exercise. Some of the results from the SPEC2000 Web site (*www.spec.org*) are shown in Figure 1.36. The *reference time* is the execution time for a particular computer system chosen by SPEC as a performance reference for all other tested systems. The *base ratio* is simply the run time for a benchmark divided into the reference time for that benchmark. The SPECfp_base2000 statistic is computed as the geometric mean of the base ratios. Let's see how a weighted arithmetic mean compares.

a. [15] <1.5> Calculate the weights for a workload so that running times on the reference computer will be equal for each of the 14 benchmarks in Figure 1.36.

SPEC CFP2000 program name	Reference time	Base ratio		
		Compaq AlphaServer ES40 Model 6/667	IBM eServer pSeries 640	Intel VC820
168.wupwise	1600	458	307	393
171.swim	3100	1079	227	406
172.mgrid	1800	525	284	246
173.applu	2100	386	311	244
177.mesa	1400	502	273	535
178.galgel	2900	445	380	295
179.art	2600	1238	924	379
183.equake	1300	220	528	233
187.facerec	1900	677	215	296
188.ammp	2200	405	272	283
189.lucas	2000	639	261	312
191.fma3d	2100	472	305	282
200.sixtrack	1100	273	205	169
301.apsi	2600	445	292	345
SPECfp_base2000 (geometric mean)		500	313	304

Figure 1.36 SPEC2000 performance for SPEC CFP2000. *Reference time* for each program is for a particular Sun Microsystems Ultra 10 computer configuration. *Base ratio* is the measured execution time of an executable generated by conservative compiler optimization, which is required to be identical for each program, divided into the reference time and is expressed as a percentage. SPECfp_base2000 is the geometric mean of the 14 base ratio values; it would be 100 for the reference computer system. The Compaq AlphaServer ES40 6/667 uses a 667 MHz Alpha 21164A microprocessor and an 8 MB off-chip tertiary cache. The IBM eServer pSeries 640 uses a 375 MHz Power3-II CPU and a 4 MB off-chip secondary cache. The Intel VC820 uses a 1000 MHz Pentium III processor with a 256 KB on-chip secondary cache. Data are from the SPEC Web site (*www.spec.org*).

b. [15] <1.5> Using the weights computed in part (a) of this exercise, calculate the weighted arithmetic means of the execution times of the 14 programs in Figure 1.36.

1.15 [15/20/15] <1.5> "The only consistent and reliable measure of performance is the execution time of real programs" [page 25].

a. [15] <1.5> For the execution time of a real program on a given computer system to have a meaningful value, two conditions must be satisfied. One has to do with the conditions within the computer system at the time of measurement, and the other has to do with the measured program itself. What are the conditions?

b. [20] <1.5> Programs such as operating systems, Web servers, device drivers, and TCP/IP stacks are intended to either not terminate or terminate only upon

an exceptional condition. Is throughput (work per unit time) a consistent and reliable performance measure for these programs? Why, or why not?

c. [15] <1.5> The fundamental unit of work that is of interest for programs such as Web servers and database systems is the transaction. Many computer systems are able to pipeline the processing of transactions, thus overlapping transaction execution times. What performance measurement error does the use of throughput rather than transaction execution time avoid?

✪ 1.16 [15/15/15] <1.6> Three enhancements with the following speedups are proposed for a new architecture:

$\text{Speedup}_1 = 30$

$\text{Speedup}_2 = 20$

$\text{Speedup}_3 = 15$

Only one enhancement is usable at a time.

a. [15] <1.6> If enhancements 1 and 2 are each usable for 25% of the time, what fraction of the time must enhancement 3 be used to achieve an overall speedup of 10?

b. [15] <1.6> Assume the enhancements can be used 25%, 35%, and 10% of the time for enhancements 1, 2, and 3, respectively. For what fraction of the reduced execution time is no enhancement in use?

c. [15] <1.6> Assume, for some benchmark, the possible fraction of use is 15% for each of enhancements 1 and 2 and 70% for enhancement 3. We want to maximize performance. If only one enhancement can be implemented, which should it be? If two enhancements can be implemented, which should be chosen?

1.17 [10/10/10/15/10] <1.6, 1.9> Your company has a benchmark that is considered representative of your typical applications. An embedded processor under consideration to support your task does not have a floating-point unit and must emulate each floating-point instruction by a sequence of integer instructions. This processor is rated at 120 MIPS on the benchmark. A third-party vendor offers a compatible coprocessor to boost performance. That coprocessor executes each floating-point instruction in hardware (i.e., no emulation is necessary). The processor/coprocessor combination rates 80 MIPS on the same benchmark. The following symbols are used to answer parts (a)–(e) of this exercise:

I—Number of integer instructions executed on the benchmark.

F—Number of floating-point instructions executed on the benchmark.

Y—Number of integer instructions to emulate one floating-point instruction.

W—Time to execute the benchmark on the processor alone.

B—Time to execute the benchmark on the processor/coprocessor combination.

a. [10] <1.6, 1.9> Write an equation for the MIPS rating of each configuration using the symbols above.

b. [10] <1.6> For the configuration without the coprocessor, we measure that $F = 8 \times 10^6$, $Y = 50$, and $W = 4$ seconds. Find I.

c. [10] <1.6> What is the value of B?

d. [15] <1.6, 1.9> What is the MFLOPS rating of the system with the coprocessor?

e. [10] <1.6, 1.9> Your colleague wants to purchase the coprocessor even though the MIPS rating for the configuration using the coprocessor is less than that of the processor alone. Is your colleague's evaluation correct? Defend your answer.

✪ 1.18 [10/12] <1.6, 1.9> One problem cited with MFLOPS as a measure is that not all FLOPS are created equal. To overcome this problem, normalized or weighted MFLOPS measures were developed. Figure 1.37 shows how the authors of the "Livermore Loops" benchmark calculate the number of normalized floating-point operations per program according to the operations actually found in the source code. Thus, the *native MFLOPS* rating is not the same as the *normalized MFLOPS* rating reported in the supercomputer literature, which has come as a surprise to a few computer designers.

Let's examine the effects of this weighted MFLOPS measure. The SPEC CFP2000 171.swim program runs on the Compaq AlphaServer ES40 in 287 seconds. The number of floating-point operations executed in that program are listed in Figure 1.38.

a. [10] <1.6, 1.9> What is the native MFLOPS for 171.swim on a Compaq AlphaServer ES40?

b. [12] <1.6, 1.9> Using the conversions in Figure 1.37, what is the normalized MFLOPS?

1.19 [30] <1.5, 1.9> Devise a program in C that gets the peak MIPS rating for a computer. Run it on two machines to calculate the peak MIPS. Now run SPEC CINT2000 176.gcc on both machines. How well do peak MIPS predict performance of 176.gcc?

Real FP operations	Normalized FP operations
Add, Subtract, Compare, Multiply	1
Divide, Square root	4
Functions (Exponentiation, Sin, . . .)	8

Figure 1.37 Real versus normalized floating-point operations. The number of normalized floating-point operations per real operation in a program used by the authors of the Livermore FORTRAN kernels, or "Livermore Loops," to calculate MFLOPS. A kernel with one Add, one Divide, and one Sin would be credited with 13 normalized floating-point operations. Native MFLOPS won't give the results reported for other machines on that benchmark.

Floating-point operation	Times executed
load	77,033,084,546
store	22,823,523,329
copy	4,274,605,803
add	41,324,938,303
sub	21,443,753,876
mul	31,487,066,317
div	1,428,275,916
convert	11,760,563
Total	199,827,008,653

Figure 1.38 Floating-point operations in SPEC CFP2000 171.swim.

1.20 [30] <1.5, 1.9> Devise a program in C or FORTRAN that gets the peak MFLOPS rating for a computer. Run it on two machines to calculate the peak MFLOPS. Now run the SPEC CFP2000 171.swim benchmark on both machines. How well do peak MFLOPS predict performance of 171.swim?

1.21 [20/20/25] <1.7> Vendors often sell several models of a computer that have identical hardware with the sole exception of processor clock speed. The following questions explore the influence of clock speed on performance.

a. [20] <1.7> From the collection of computers with reported SPEC CFP2000 benchmark results at *www.spec.org/osg/cpu2000/results/*, choose a set of three computer models that are identical in tested configurations (both hardware and software) except for clock speed. For each pair of models, compare the clock speedup to the SPECint_base2000 benchmark speedup. How closely does benchmark performance track clock speed? Is this consistent with the description of the SPEC benchmarks on pages 28–30?

b. [20] <1.7> Now the workload for the computers in part (a) is as follows: a user launches a word-processing program, opens the file of an existing five-page text document, checks spelling, finds no errors, and finally prints the document to an inkjet printer. Suppose the execution time for this benchmark on the slowest clock rate model is 1 minute and 30 seconds, apportioned in this way: 5 seconds to load the word-processing program and the chosen document file from disk to memory, 5 seconds for the user to invoke spell checking, 1 second for spell checking to complete, 2 seconds for the user to absorb the information that there are no spelling errors, 5 seconds for the user to initiate the printing command, 2 seconds for the printing dialog box to appear, 2 seconds for the user to accept the default printing options and command that printing proceed, 8 seconds for the printer to start, and 1 minute to print the five pages.

User think time—the time it takes for a human to respond after waiting for a computer reply in interactive use—improves significantly when the computer can respond to a command quickly because the user maintains better mental focus. Assume that for computer response times less than 2 seconds, any computer response time improvement is matched by double that amount of improvement in the human response time, bounded by a 0.5 second minimum human response time.

What is the clock speedup and word-processing benchmark speedup for each pair of computer models? Discuss the importance of a faster processor for this workload.

c. [25] <1.7> Choose a desktop computer vendor that has a Web-based store and find the price for three systems that are configured identically except for processor clock rate. What is the relative price performance for each system if the workload execution time is determined only by processor clock speed ($ per MHz)? What is the relative price performance ($ per second) for each system if, during a workload execution time total of 100 seconds on the slowest system, the processor is busy 5% of the time and other system components and/or the user are busy the other 95% of the time?

1.22 [30] <1.5, 1.7> Find results from different benchmark sets, for example, PC versus SPEC benchmarks, and compare their performance measurements for two related processors, such as the Pentium III and Pentium 4. Discuss reasons for the differences in performance.

1.23 [20] <1.5, 1.8> Assume that typical power consumption for the 667 MHz Alpha 21164A, 375 MHz Power3-II, and 1000 MHz Pentium III processors is 50, 27, and 35 W, respectively. Using data from Figure 1.36 and scaling to the performance of the Pentium III, create a graph showing the relative performance and the relative performance per watt of these three processors for 171.swim, 183.equake, 301.apsi, and SPECfp_base2000.

1.24 [25] <1.4, 1.8> Design goals for a desktop computer system besides price and performance might include reducing size and noise. Assume that room air is available for cooling. Develop a simple model, similar to the cost model of Figure 1.10, that identifies the sources of additional system demands for power caused by a watt of processor power and includes the transition from passive, convective airflow to forced airflow cooling. Develop an analogous model showing the effect of processor power on system volume. Describe the effect that processor power consumption has on system noise and size.

1.25 [Discussion] <1.5> What is an interpretation of the geometric mean of execution times? What do you think are the advantages and disadvantages of using (a) total execution times versus (b) weighted arithmetic means of execution times using equal running time on the SPARC versus (c) geometric means of ratios of speed to the SPARC (used as the reference machine by SPEC2000)?

1.26 [30] <1.5> SPEC2000 programs are often compiled at levels of optimization that are almost never used by software that is sold commercially—and sometimes using compilers that no one would use in a real product. Rerun SPEC2000 programs on machines for which you can find official ratings, but this time run binaries of the programs compiled with simple optimization and no optimization. Does relative performance change? What do you conclude about the machines? About SPEC2000?

1.27 [Discussion] <1.5> PC benchmark suites use scripts to run programs as fast as possible, that is, with no user think time, the time a real user would spend understanding the current program output before providing the next user input. Also, to be sure to exercise new features of the latest version of the benchmark program, apparently they exercise every option once. What are the disadvantages of this approach? Can you think of compiler or architecture techniques that improve performance for real users but are penalized by this style of benchmarking?

1.28 [Discussion] <1.6> Amdahl's Law makes it clear that to deliver substantial performance improvement, a design enhancement must be usable a large fraction of the time. With this principle in mind, examine the table of contents for this text, determine the major themes of computer design that are covered and the ranking of specific techniques within the major topics, and discuss the extent to which Amdahl's Law is a useful dimension on which to organize the study of computer design.