decrease in load capacitance and voltage, leading to an overall growth in power consumption. The first microprocessors consumed tenths of a watt, while a 2 GHz Pentium 4 consumes close to 100 watts. The fastest workstation and server microprocessors in 2001 consumed between 100 and 150 watts. Distributing the power, removing the heat, and preventing hot spots have become increasingly difficult challenges, and it is likely that power rather than raw transistor count will become the major limitation in the near future.

1.4 Cost, Price, and Their Trends

Although there are computer designs where costs tend to be less important—specifically supercomputers—cost-sensitive designs are of growing significance: More than half the PCs sold in 1999 were priced at less than $1000, and the average price of a 32-bit microprocessor for an embedded application is in the tens of dollars. Indeed, in the past 15 years, the use of technology improvements to achieve lower cost, as well as increased performance, has been a major theme in the computer industry.

Textbooks often ignore the cost half of cost-performance because costs change, thereby dating books, and because the issues are subtle and differ across industry segments. Yet an understanding of cost and its factors is essential for designers to be able to make intelligent decisions about whether or not a new feature should be included in designs where cost is an issue. (Imagine architects designing skyscrapers without any information on costs of steel beams and concrete!)

This section focuses on cost and price, specifically on the relationship between price and cost: price is what you sell a finished good for, and cost is the amount spent to produce it, including overhead. We also discuss the major trends and factors that affect cost and how it changes over time. The exercises and examples use specific cost data that will change over time, though the basic determinants of cost are less time sensitive. This section will introduce you to these topics by discussing some of the major factors that influence the cost of a computer design and how these factors are changing over time.

The Impact of Time, Volume, and Commodification

The cost of a manufactured computer component decreases over time even without major improvements in the basic implementation technology. The underlying principle that drives costs down is the *learning curve*—manufacturing costs decrease over time. The learning curve itself is best measured by change in *yield*—the percentage of manufactured devices that survives the testing procedure. Whether it is a chip, a board, or a system, designs that have twice the yield will have basically half the cost.

Understanding how the learning curve will improve yield is key to projecting costs over the life of the product. As an example of the learning curve in action, the price per megabyte of DRAM drops over the long term by 40% per year.

Since DRAMs tend to be priced in close relationship to cost—with the exception of periods when there is a shortage—price and cost of DRAM track closely. In fact, there are some periods (for example, early 2001) in which it appears that price is less than cost; of course, the manufacturers hope that such periods are both infrequent and short!

Figure 1.5 plots the price of a new DRAM chip over its lifetime. Between the start of a project and the shipping of a product, say, two years, the cost of a new DRAM drops by a factor of between 5 and 10 in constant dollars. Since not all component costs change at the same rate, designs based on projected costs result in different cost-performance trade-offs than those using current costs. The caption of Figure 1.5 discusses some of the long-term trends in DRAM price.

Microprocessor prices also drop over time, but because they are less standardized than DRAMs, the relationship between price and cost is more complex. In a

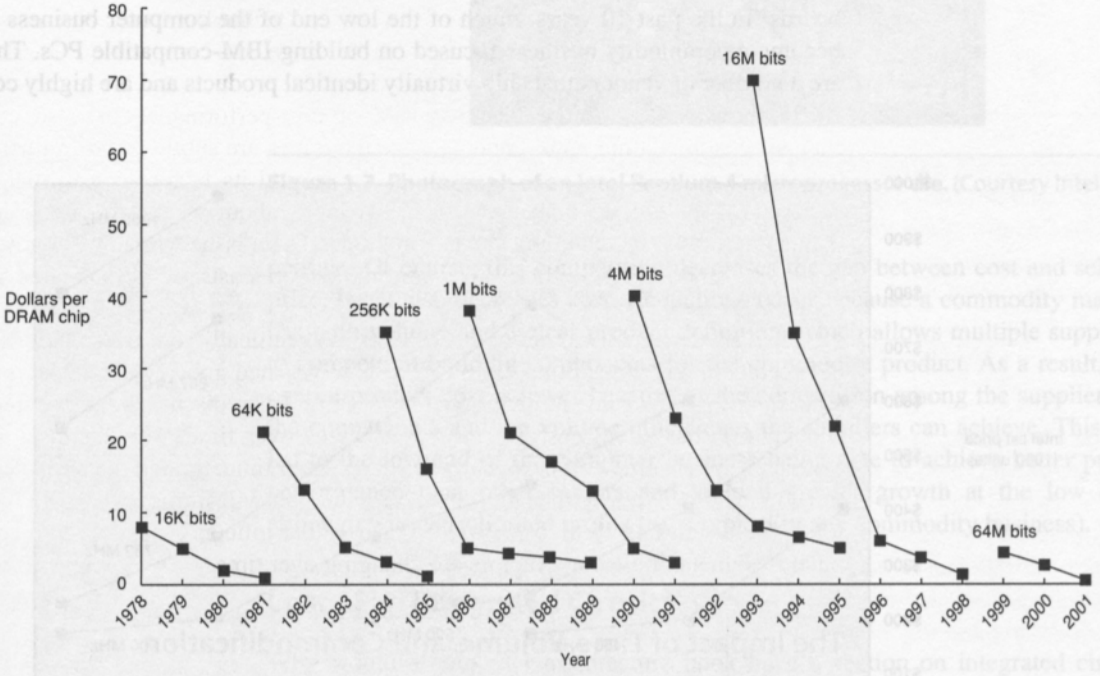

Figure 1.5 Prices of six generations of DRAMs (from 16K bits to 64M bits) over time in 1977 dollars, showing the learning curve at work. A 1977 dollar is worth about $2.95 in 2001; more than half of this inflation occurred in the five-year period of 1977–82, during which the value changed to $1.59. The cost of a megabyte of memory has dropped *incredibly* during this period, from over $5000 in 1977 to about $0.35 in 2000, and an amazing $0.08 in 2001 (in 1977 dollars)! Each generation drops in constant dollar price by a factor of 10 to 30 over its lifetime. Starting in about 1996, an explosion of manufacturers has dramatically reduced margins and increased the rate at which prices fall, as well as the eventual final price for a DRAM. Periods when demand exceeded supply, such as 1987–88 and 1992–93, have led to temporary higher pricing, which shows up as a slowing in the rate of price decrease; more dramatic short-term fluctuations have been smoothed out. In late 2000 and through 2001, there has been tremendous oversupply, leading to an accelerated price decrease, which is probably not sustainable.

period of significant competition, price tends to track cost closely, although microprocessor vendors probably rarely sell at a loss. Figure 1.6 shows processor price trends for the Pentium III.

Volume is a second key factor in determining cost. Increasing volumes affect cost in several ways. First, they decrease the time needed to get down the learning curve, which is partly proportional to the number of systems (or chips) manufactured. Second, volume decreases cost, since it increases purchasing and manufacturing efficiency. As a rule of thumb, some designers have estimated that cost decreases about 10% for each doubling of volume. Also, volume decreases the amount of development cost that must be amortized by each machine, thus allowing cost and selling price to be closer. We will return to the other factors influencing selling price shortly.

Commodities are products that are sold by multiple vendors in large volumes and are essentially identical. Virtually all the products sold on the shelves of grocery stores are commodities, as are standard DRAMs, disks, monitors, and keyboards. In the past 10 years, much of the low end of the computer business has become a commodity business focused on building IBM-compatible PCs. There are a number of vendors that ship virtually identical products and are highly com-

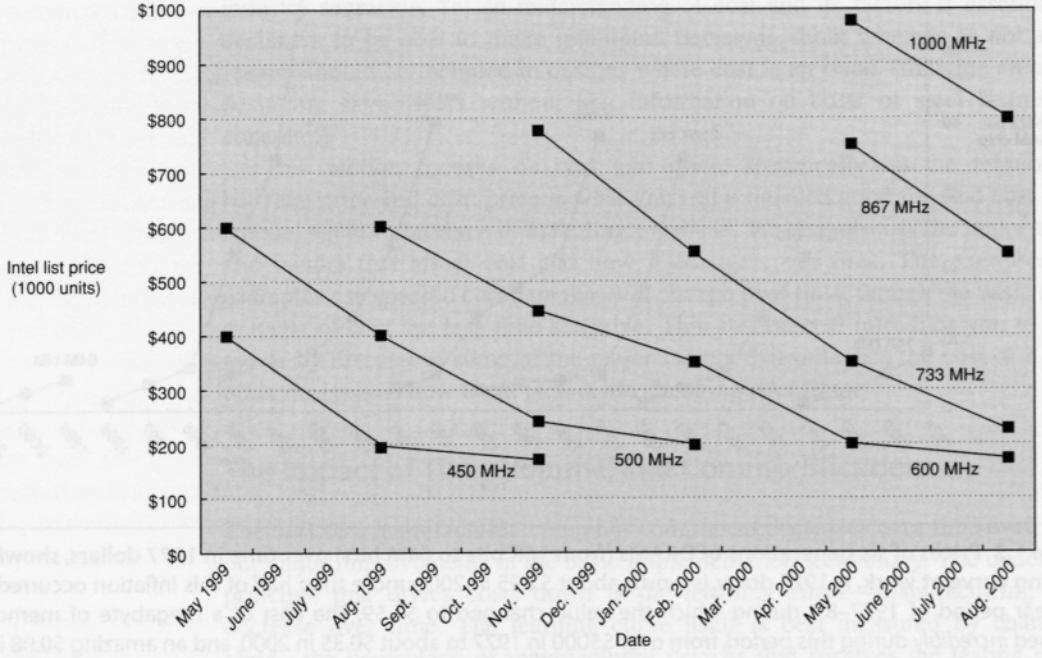

Figure 1.6 The price of an Intel Pentium III at a given frequency decreases over time as yield enhancements decrease the cost of a good die and competition forces price reductions. Data courtesy of *Microprocessor Report,* May 2000 issue. The most recent introductions will continue to decrease until they reach similar prices to the lowest-cost parts available today ($100–$200). Such price decreases assume a competitive environment where price decreases track cost decreases closely.

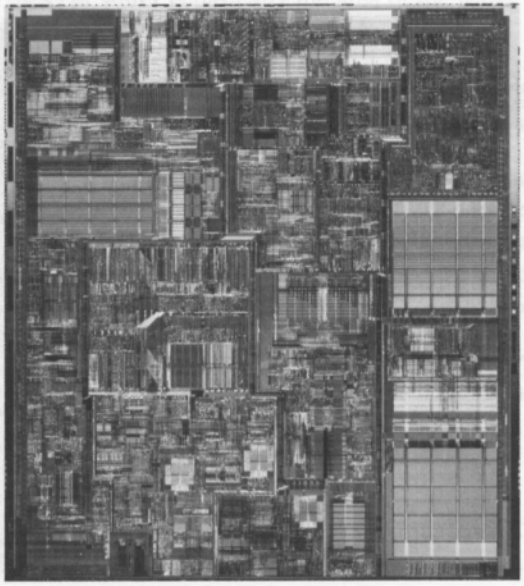

Figure 1.7 Photograph of an Intel Pentium 4 microprocessor die. (Courtesy Intel.)

petitive. Of course, this competition decreases the gap between cost and selling price, but it also decreases cost. Reductions occur because a commodity market has both volume and a clear product definition, which allows multiple suppliers to compete in building components for the commodity product. As a result, the overall product cost is lower because of the competition among the suppliers of the components and the volume efficiencies the suppliers can achieve. This has led to the low end of the computer business being able to achieve better price-performance than other sectors and yielded greater growth at the low end, although with very limited profits (as is typical in any commodity business).

Cost of an Integrated Circuit

Why would a computer architecture book have a section on integrated circuit costs? In an increasingly competitive computer marketplace where standard parts—disks, DRAMs, and so on—are becoming a significant portion of any system's cost, integrated circuit costs are becoming a greater portion of the cost that varies between machines, especially in the high-volume, cost-sensitive portion of the market. Thus computer designers must understand the costs of chips to understand the costs of current computers.

Although the costs of integrated circuits have dropped exponentially, the basic procedure of silicon manufacture is unchanged: A *wafer* is still tested and chopped into *dies* that are packaged (see Figures 1.7 and 1.8). Thus the cost of a packaged integrated circuit is

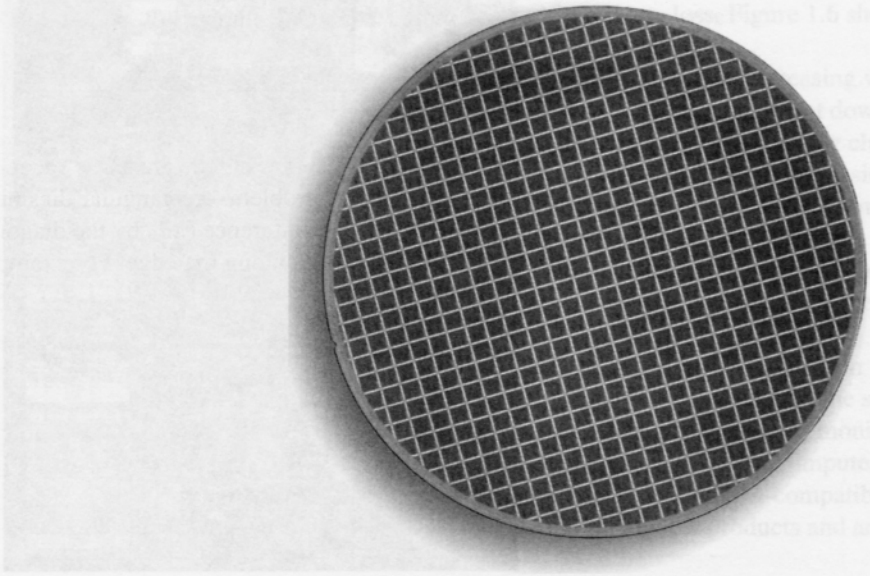

Figure 1.8 This 8-inch wafer contains 564 MIPS64 R20K processors implemented in a 0.18µ process. The R20K is an implementation of the MIPS64 architecture with instruction set extensions, called MIPS-3D, for use in three-dimensional graphics computations. The R20K is available at speeds from 500 to 750 MHz and is capable of executing two integer operations every clock cycle. Using the MIPS-3D instructions, the R20K can perform up to 3 billion floating-point operations per second. (Courtesy MIPS Technologies, Inc.)

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

In this section, we focus on the cost of dies, summarizing the key issues in testing and packaging at the end. A longer discussion of the testing costs and packaging costs appears in the exercises.

Learning how to predict the number of good chips per wafer requires first learning how many dies fit on a wafer and then learning how to predict the percentage of those that will work. From there it is simple to predict cost:

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$$

The most interesting feature of this first term of the chip cost equation is its sensitivity to die size, shown below.

The number of dies per wafer is basically the area of the wafer divided by the area of the die. It can be more accurately estimated by

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}}$$

The first term is the ratio of wafer area (πr^2) to die area. The second compensates for the "square peg in a round hole" problem—rectangular dies near the periphery of round wafers. Dividing the circumference (πd) by the diagonal of a square die is approximately the number of dies along the edge. For example, a wafer 30 cm (≈ 12 inches) in diameter produces $\pi \times 225 - (\pi \times 30/1.41) = 640$ 1-cm dies.

Example Find the number of dies per 30 cm wafer for a die that is 0.7 cm on a side.

Answer The total die area is 0.49 cm^2. Thus

$$\text{Dies per wafer} = \frac{\pi \times (30/2)^2}{0.49} - \frac{\pi \times 30}{\sqrt{2 \times 0.49}} = \frac{706.5}{0.49} - \frac{94.2}{0.99} = 1347$$

But this only gives the maximum number of dies per wafer. The critical question is, What is the fraction or percentage of good dies on a wafer number, or the *die yield*? A simple empirical model of integrated circuit yield, which assumes that defects are randomly distributed over the wafer and that yield is inversely proportional to the complexity of the fabrication process, leads to the following:

$$\text{Die yield} = \text{Wafer yield} \times \left(1 + \frac{\text{Defects per unit area} \times \text{Die area}}{\alpha}\right)^{-\alpha}$$

where *wafer yield* accounts for wafers that are completely bad and so need not be tested. For simplicity, we'll just assume the wafer yield is 100%. Defects per unit area is a measure of the random manufacturing defects that occur. In 2001, these values typically range between 0.4 and 0.8 per square centimeter, depending on the maturity of the process (recall the learning curve, mentioned earlier). Lastly, α is a parameter that corresponds inversely to the number of masking levels, a measure of manufacturing complexity, critical to die yield. For today's multilevel metal CMOS processes, a good estimate is $\alpha = 4.0$.

Example Find the die yield for dies that are 1 cm on a side and 0.7 cm on a side, assuming a defect density of 0.6 per cm^2.

Answer The total die areas are 1 cm^2 and 0.49 cm^2. For the larger die the yield is

$$\text{Die yield} = \left(1 + \frac{0.6 \times 1}{4.0}\right)^{-4} = 0.57$$

For the smaller die, it is

$$\text{Die yield} = \left(1 + \frac{0.6 \times 0.49}{4.0}\right)^{-4} = 0.75$$

The bottom line is the number of good dies per wafer, which comes from multiplying dies per wafer by die yield (which incorporates the effects of defects). The examples above predict 366 good 1 cm^2 dies from the 30 cm wafer and 1014 good 0.49 cm^2 dies. Most 32-bit and 64-bit microprocessors in a modern 0.25μ technology fall between these two sizes, with some processors being as large as 2 cm^2 in the prototype process before a shrink. Low-end embedded 32-bit processors are sometimes as small as 0.25 cm^2, while processors used for embedded control (in printers, automobiles, etc.) are often less than 0.1 cm^2. Figure 1.34 for Exercise 1.8 shows the die size and technology for several current microprocessors.

Given the tremendous price pressures on commodity products such as DRAM and SRAM, designers have included redundancy as a way to raise yield. For a number of years, DRAMs have regularly included some redundant memory cells, so that a certain number of flaws can be accommodated. Designers have used similar techniques in both standard SRAMs and in large SRAM arrays used for caches within microprocessors. Obviously, the presence of redundant entries can be used to significantly boost the yield.

Processing a 30 cm diameter wafer in a leading-edge technology with four to six metal layers costs between $5000 and $6000 in 2001. Assuming a processed wafer cost of $5500, the cost of the 0.49 cm^2 die would be around $5.42, while the cost per die of the 1 cm^2 die would be about $15.03, or almost three times the cost for a die that is two times larger.

What should a computer designer remember about chip costs? The manufacturing process dictates the wafer cost, wafer yield, and defects per unit area, so the sole control of the designer is die area. Since α is around 4 for the advanced processes in use today, it would appear that the cost of a die would grow with the fourth power of the die size. In practice, however, because the number of defects per unit area is small, the number of good dies per wafer, and hence the cost per die, grows roughly as the square of the die area. The computer designer affects die size, and hence cost, both by what functions are included on or excluded from the die and by the number of I/O pins.

Before we have a part that is ready for use in a computer, the die must be tested (to separate the good dies from the bad), packaged, and tested again after packaging. These steps all add significant costs. These processes and their contribution to cost are discussed and evaluated in Exercise 1.8.

The above analysis has focused on the variable costs of producing a functional die, which is appropriate for high-volume integrated circuits. There is, however, one very important part of the fixed cost that can significantly impact the cost of an integrated circuit for low volumes (less than 1 million parts), namely, the cost of a mask set. Each step in the integrated circuit process requires

a separate mask. Thus, for modern high-density fabrication processes with four to six metal layers, mask costs often exceed $1 million. Obviously, this large fixed cost affects the cost of prototyping and debugging runs and, for small-volume production, can be a significant part of the production cost. Since mask costs are likely to continue to increase, designers may incorporate reconfigurable logic to enhance the flexibility of a part, or choose to use gate arrays (which have fewer custom mask levels) and thus reduce the cost implications of masks.

Distribution of Cost in a System: An Example

To put the costs of silicon in perspective, Figure 1.9 shows the approximate cost breakdown for a $1000 PC in 2001. Although the costs of some parts of this machine can be expected to drop over time, other components, such as the packaging and power supply, have little room for improvement. Furthermore, we can expect that future machines will have larger memories and disks, meaning that prices drop more slowly than the technology improvement.

System	Subsystem	Fraction of total
Cabinet	Sheet metal, plastic	2%
	Power supply, fans	2%
	Cables, nuts, bolts	1%
	Shipping box, manuals	1%
	Subtotal	**6%**
Processor board	Processor	22%
	DRAM (128 MB)	5%
	Video card	5%
	Motherboard with basic I/O support, networking	5%
	Subtotal	**37%**
I/O devices	Keyboard and mouse	3%
	Monitor	19%
	Hard disk (20 GB)	9%
	DVD drive	6%
	Subtotal	**37%**
Software	OS + Basic Office Suite	20%

Figure 1.9 Estimated distribution of costs of the components in a $1000 PC in 2001. Notice that the largest single item is the CPU, closely followed by the monitor. (Interestingly, in 1995, the DRAM memory at about 1/3 of the total cost was the most expensive component! Since then, cost per MB has dropped by about a factor of 15!) Touma [1993] discusses computer system costs and pricing in more detail. These numbers are based on estimates of volume pricing for the various components.

Cost versus Price—Why They Differ and By How Much

Costs of components may confine a designer's desires, but they are still far from representing what the customer must pay. But why should a computer architecture book contain pricing information? Cost goes through a number of changes before it becomes price, and the computer designer should understand how a design decision will affect the potential selling price. For example, changing cost by $1000 may change price by $3000 to $4000. Without understanding the relationship of cost to price the computer designer may not understand the impact on price of adding, deleting, or replacing components.

The relationship between price and volume can increase the impact of changes in cost, especially at the low end of the market. Typically, fewer computers are sold as the price increases. Furthermore, as volume decreases, costs rise, leading to further increases in price. Thus, small changes in cost can have a larger than obvious impact. The relationship between cost and price is a complex one, and entire books have been written on the subject. The purpose of this section is to give you a simple introduction to what factors determine price, and to typical ranges for these factors.

The categories that make up price can be shown either as a tax on cost or as a percentage of the price. We will look at the information both ways. These differences between price and cost also depend on where in the computer marketplace a company is selling. To show these differences, Figure 1.10 shows how the difference between cost of materials and list price is decomposed, with the price increasing from left to right as we add each type of overhead.

Direct costs refer to the costs directly related to making a product. These include labor costs, purchasing components, scrap (the leftover from yield), and warranty, which covers the costs of systems that fail at the customer's site during the warranty period. Direct cost typically adds 10% to 30% to component cost. Service or maintenance costs are not included because the customer typically pays those costs, although a warranty allowance may be included here or in gross margin, discussed next.

The next addition is called the *gross margin*, the company's overhead that cannot be billed directly to one product. This can be thought of as indirect cost. It includes the company's research and development (R&D), marketing, sales, manufacturing equipment maintenance, building rental, cost of financing, pretax profits, and taxes. When the component costs are added to the direct cost and gross margin, we reach the *average selling price*—ASP in the language of MBAs—the money that comes directly to the company for each product sold. The gross margin is typically 10% to 45% of the average selling price, depending on the uniqueness of the product. Manufacturers of low-end PCs have lower gross margins for several reasons. First, their R&D expenses are lower. Second, their cost of sales is lower, since they use indirect distribution (by mail, the Internet, phone order, or retail store) rather than salespeople. Third, because their products are less distinctive, competition is more intense, thus forcing lower prices and often lower profits, which in turn lead to a lower gross margin.

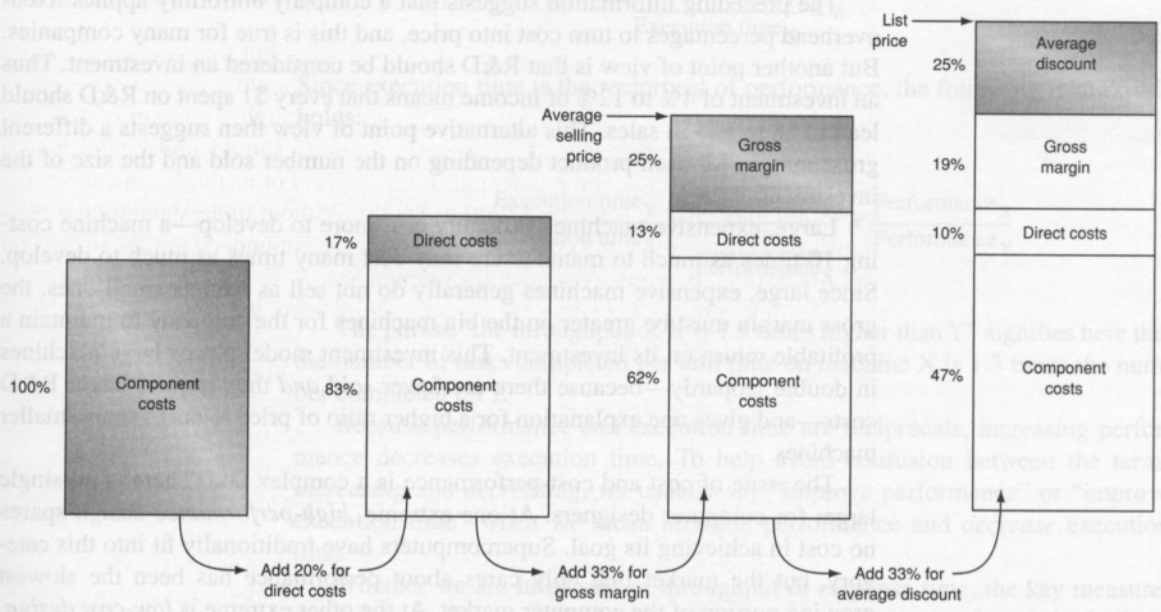

Figure 1.10 The components of price for a $1000 PC. Each increase is shown along the bottom as a tax on the prior price. The percentages of the new price for all elements are shown on the left of each column.

List price and average selling price are not the same, since companies typically offer volume discounts, lowering the average selling price. As personal computers became commodity products, the retail markups have dropped significantly, so list price and average selling price have closed.

As we said, pricing is sensitive to competition: A company may not be able to sell its product at a price that includes the desired gross margin. In the worst case, the price must be significantly reduced, lowering gross margin until profit becomes negative! A company striving for market share can reduce price and profit to increase the attractiveness of its products. If the volume grows sufficiently, costs can be reduced. Remember that these relationships are extremely complex and to understand them in depth would require an entire book, as opposed to one section in one chapter. For example, if a company cuts prices, but does not obtain a sufficient growth in product volume, the chief impact would be lower profits.

Many engineers are surprised to find that most companies spend only 4% (in the commodity PC business) to 12% (in the high-end server business) of their income on R&D, which includes all engineering (except for manufacturing and field engineering). This well-established percentage is reported in companies' annual reports and tabulated in national magazines, so this percentage is unlikely to change over time. In fact, experience has shown that computer companies with R&D percentages of 15–20% rarely prosper over the long term.

The preceding information suggests that a company uniformly applies fixed-overhead percentages to turn cost into price, and this is true for many companies. But another point of view is that R&D should be considered an investment. Thus an investment of 4% to 12% of income means that every $1 spent on R&D should lead to $8 to $25 in sales. This alternative point of view then suggests a different gross margin for each product depending on the number sold and the size of the investment.

Large, expensive machines generally cost more to develop—a machine costing 10 times as much to manufacture may cost many times as much to develop. Since large, expensive machines generally do not sell as well as small ones, the gross margin must be greater on the big machines for the company to maintain a profitable return on its investment. This investment model places large machines in double jeopardy—because there are fewer sold *and* they require larger R&D costs—and gives one explanation for a higher ratio of price to cost versus smaller machines.

The issue of cost and cost-performance is a complex one. There is no single target for computer designers. At one extreme, *high-performance design* spares no cost in achieving its goal. Supercomputers have traditionally fit into this category, but the market that only cares about performance has been the slowest growing portion of the computer market. At the other extreme is *low-cost design,* where performance is sacrificed to achieve lowest cost; some portions of the embedded market—for example, the market for cell phone microprocessors—behave exactly like this. Between these extremes is *cost-performance design,* where the designer balances cost versus performance. Most of the PC market, the workstation market, and most of the server market (at least including both low-end and midrange servers) operate in this region. In the past 10 years, as computers have downsized, both low-cost design and cost-performance design have become increasingly important. This section has introduced some of the most important factors in determining cost; the next section deals with performance.

1.5 Measuring and Reporting Performance

When we say one computer is faster than another, what do we mean? The user of a desktop machine may say a computer is faster when a program runs in less time, while the computer center manager running a large server system may say a computer is faster when it completes more jobs in an hour. The computer user is interested in reducing *response time*—the time between the start and the completion of an event—also referred to as *execution time*. The manager of a large data processing center may be interested in increasing *throughput*—the total amount of work done in a given time.

In comparing design alternatives, we often want to relate the performance of two different machines, say, X and Y. The phrase "X is faster than Y" is used here to mean that the response time or execution time is lower on X than on Y for the given task. In particular, "X is n times faster than Y" will mean