

Chapter 2

Microprocessors and Memory

In This Chapter

- ▶ You will learn what a processor is and how it works.
 - ▶ You will learn the importance of cache to computer performance.
 - ▶ You will learn the differences among processors.
 - ▶ You will learn how different types of memory work.
 - ▶ You will receive recommendations on processors and memory.
-

Most people think of computers as single, monolithic entities, but the reality is, they're actually made up of many different components doing many different things. In this chapter, and the remaining five chapters in Part I, I dive into the details of the different components used in today's computer systems. My goal is to provide you with all the answers you need to make intelligent PC purchasing and upgrade decisions, as well to give you some background on how the various pieces work—both by themselves, and in conjunction with other components.

In each chapter, I focus on the main features of each of the different components, and explain what they are, what they do, and why they're important. As a result, by the end of Part I, you should be able to completely decipher just about any computer-related ad or spec sheet that you come across. And more importantly, you should be able to figure out which features or specifications are important and which ones aren't. The first components I cover, the microprocessor and memory, form the central core of any computer's operation. The processor, of course, is the most important part of any computer. It is the workhorse that drives a PC's operation. But memory also plays an important role, because memory provides a processor with the raw materials and the working area it needs to perform its computational magic. In fact, working together, these two pieces enable a computer to "compute."

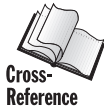
Practically speaking, the speed at which the processor and memory do their work has the most dramatic impact on how fast a computer operates. In addition, the type of processor you have can affect the type of software your computer is capable of running. So, knowing about different processors and

types of memory, as well as how they interact, gives you an excellent idea of how powerful a particular computer system really is.

The Processor

The heart and/or brain of the computer — depending on your perspective — is the *central processing unit* (CPU), also known as the microprocessor or simply processor for short. The processor performs the major number crunching that drives any computer's operation. Processors play such an important role that computers are often defined and described based solely on the type of processor they have; for example, "I've got a Pentium III computer" or "I've got a K7."

Processors used to be a single chip that connected to different-sized sockets on your computer's motherboard, which is the main circuit board inside your computer. In fact, some still are, but many newer processors come on a separate circuit board — sometimes called a daughterboard (because of its relationship to the motherboard) — that plugs into a special processor slot on the motherboard.



See the section on physical connections later in this chapter for more.

The most important features to look for in a new processor are:

- The type of processor it is (for example, Pentium III vs. K7)
- The speed at which it operates
- The amount and type of L2 cache (a special type of high-speed memory) it includes
- Any additional processor instructions it supports
- The type of physical connector it uses

How it works

Processors work by performing calculations based on specific instructions that software running on the computer provides. These instructions, which are loaded into the processor when an application runs, tell the processor how to manipulate chunks of data stored in the computer's memory (RAM). This, in turn, produces the desired result, whether it is adding boldface to text in a letter, changing the color of a background in a scanned digital photo, or what have you. In other words, processors are constantly churning through instructions and data that are loaded into it from the computer's memory.

In addition to working with the main memory, processors also work with a special type of high-speed memory referred to as *cache* (pronounced cash). In fact, most of the time processors work directly with various types of

cache memory and this cache memory, in turn, works with the main memory. Essentially, the cache memory acts as a high-speed buffer in between the processor and main memory, shuffling data into the processor as it needs it, or requests it. As a result, the processor takes advantage of the high-speed cache memory and therefore works faster, which, in turn, makes the computer that the processor drives, operate faster. Figure 2-1 illustrates the concept.

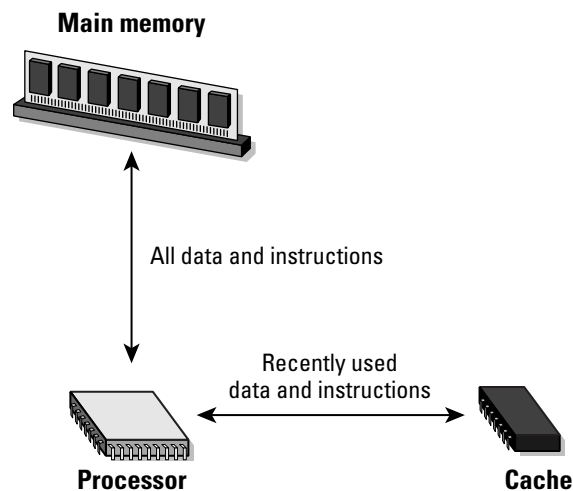


Figure 2-1: Cache helps make processors more efficient

Cache is a special form of high-speed memory that stores instructions and data the processor has recently used. Thanks to its proximity to the main computing engine inside the processor, and the fact that the processor often needs to reuse those instructions and data, cache keeps the processor busy and speeds up a computer's performance.

Processor types

Microprocessors come in different types and different speeds. The most popular processors are Intel's Pentium family, which includes the standard Pentium, the Pentium with MMX Technology, the Pentium Pro, the Celeron, the Pentium II, the Pentium II Xeon, and the Pentium III. (MMX, which is sometimes referred to as MultiMedia Extensions, but according to Intel doesn't really mean anything, is a set of additional processing instructions chips that support MMX can perform.) Other processors you may hear about are the AMD K5, K6, K6-2, K6-3, and K7; the Cyrix 5x86, 6x86, 6x86MX, MediaGX, MII, and MXi; Centaur/IDT's WinChip, WinChip 2, and WinChip 3; the Rise Technology MP6; the Motorola/IBM PowerPC; and the Digital Alpha.

Table 2-1 is a list of some of the more common processors.

Table 2-1 Common Microprocessors	
<i>Manufacturer</i>	<i>Models</i>
Intel	Pentium
	Pentium with MMX Technology
	Pentium Pro
	Celeron (and Celeron A)
	Pentium II
	Pentium II Xeon
AMD	Pentium III
	K5
	K6
	K6-2
	K6-3
Cyrix (A division of National Semiconductor)	K7
	5x86
	6x86
	6x86MX
	MediaGX
	MII
Centaur/IDT	MXi
	WinChip
	WinChip 2
Rise Technology	WinChip 3
	MP6
IBM/Motorola	PowerPC 603 and 603e
	PowerPC 604 and 604e
	PowerPC 750 (aka G3)
	G4
Digital/Compaq	Alpha

Speed

Many people think processors differ only in speed. They figure a 500 MHz processor from Company A is bound to be faster than a 400 MHz version from Company B, for example. But this is not always the case.



While a processor's rated speed is a critical factor in determining how fast it performs calculations (and hence, how fast the computer housing the processor runs), there are other important differences in how various processors work their magic internally. These differences impact how fast various microprocessors perform real-world tasks, such as checking a document for spelling mistakes, recalculating the numbers in a spreadsheet, or removing imperfections from a digital photograph.

For example, many processors can perform several calculations at once; the technology that supports this technique is called *pipelining*. In addition, some jump ahead to perform extra calculations they think the running program will ask for, before the program actually does. This is called *speculative execution*, and it is another one of several very complex operations occurring inside today's processors. In addition, different processors implement these techniques, along with other technology, in different ways, which accounts for many of the differences in overall chip performance, independent of the chip's rated speed (in MHz).

For example, future versions of the Pentium III (or perhaps IV) family of processors, which have the codenames Foster and Williamette (using product codenames is a common high-tech industry practice) are expected to include new internal refinements to make the chips faster than today's Pentium IIIs.

Another critical factor in overall chip performance is how efficient various processor designs are. Processors are capable of performing continuously, and cranking out results as quickly as they are given problems to work on. Ideally, therefore, you want to feed the processor a steady stream of data so it can keep crunching away at maximum speed. The reality, however, is that — for a number of different reasons — data is not fed into the CPU on a steady, consistent basis. In fact, most processing occurs in fits and starts. This, in turn, slows a computer down.

Figure 2-2 shows the principle in action.

One of the many ways processor designers attempt to compensate for this is by using the technologies, such as speculative execution, that I mention above. Another even more important way is by adding a special high-speed cache memory into the processor's or computer's overall design. In both cases, the goal is to make the processor work as much as possible because this translates directly into faster computer performance.

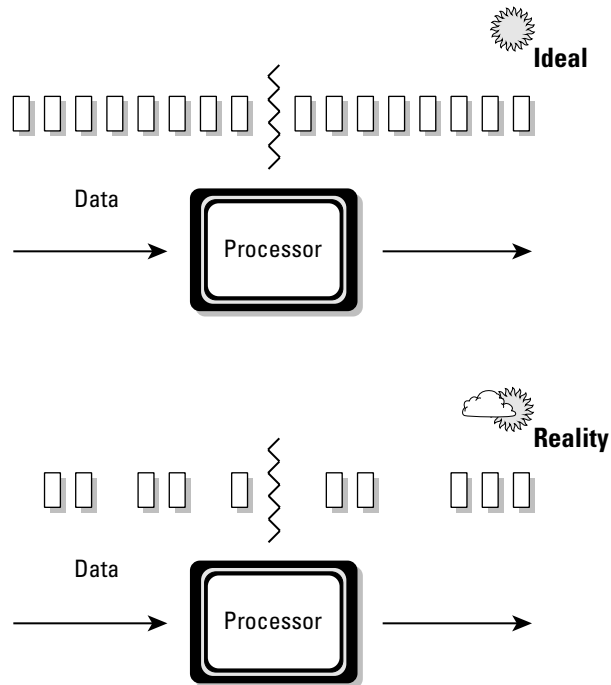


Figure 2-2: Feeding the CPU

In an ideal environment, a computer's processor runs at its maximum rate because it is fed a steady stream of data. In reality, however, various delays that occur throughout a computer system often force a processor to sit idle for brief periods of time, waiting for the next chunk of data to arrive.

L1 and L2 cache

Cache memory, in its various forms, plays a particularly important role in a processor's performance. Cache can dramatically improve a processor's efficiency by offering it access to the data it needs more quickly than regular memory would. Not only are cache memory chips (typically Static Random Access Memory, or SRAM) faster than regular memory chips, but they also have a faster connection to the processor, as I explain in just a bit.

The way cache works

Because of the way most software works, processors tend to spend a lot of their time either performing the same operations over and over or performing several different operations on the same set of data. Well, one day, somebody realized that if a processor could access used

instructions and data more quickly, it could run much more efficiently. So, a clever designer came up with the idea to create a special "work area" right alongside the processor, called a cache, that temporarily stores the data and instructions the processor used most recently.

The idea was (and is) that once the processor finishes what it is working on, it can “fetch” what it needs next from this nearby area instead of getting it from regular memory, which is further away and takes longer to get to.

Processors aren’t the only computer-related component to use a cache. Many software programs, such as Web browsers, also use a cache. While a processor’s cache and a browser’s cache are not the same thing, they are conceptually similar. For both components, a cache speeds up access to recently used information.

Web browsers, for example, set up memory and/or disk caches (which use RAM or space on the hard disk, respectively) to store recently used files. The thinking is that you probably will want to use the files you’ve accessed recently again. If they’re stored either in RAM, or on disk, the browser will be able to get to them much more quickly than if it had to go out to the Internet again. For example, when you hit the Back button on your browser, the Web page loads almost instantly because the files the browser needs are nearby. If there wasn’t a cache (or if you empty or clear the files in the cache), going back to the previous page would take just as long as when you called it up in the first place. This is similar to what happens with processors and their specialized cache; if the information the processor needs is close by in the cache, the

processor operates quickly without waiting, but if the information isn’t close by, the processor has to request it from main memory. (The main memory isn’t as slow as the Internet, of course, but it is a lot slower than getting it from the cache.)

The process of going out to main memory to get more data and instructions also forces the cache to become “flushed,” or emptied out. During the process of running programs, the processor regularly flushes the cache. This is important to know because it explains why it’s possible to have too much of a good thing; that is, too much cache. Beyond a certain amount, extra cache doesn’t help the computer’s performance very much because the cache’s contents are often flushed before it’s full. So, if you have too much cache, a certain portion will consistently go to waste. And a cache is a terrible thing to waste . . .

When the processor finds what it needs in the nearby cache, it’s called a *cache hit* and if it doesn’t, well, it’s a *cache miss*. Some utility programs, such as Symantec’s popular Norton Utilities, can track the percentage of cache hits and misses that your processor has while it’s running various applications. If you really want to impress your technical friends, you might want to start talking about your computer’s cache hit percentages while running certain applications. Then again, maybe you shouldn’t . . .

The two most common types of cache are referred to as L1, or Level 1, and L2, or Level 2 cache. (It is possible to have Level 3 caches, but they are not very common.) Although technically speaking caches are a type of memory, in most cases the L1 and L2 cache are actually built into the processor chip or processor card itself. Thus, they’re really more a feature of the processor than of memory.

Each level of cache is a separate chunk of memory and is treated independently by the processor. The two levels refer to how close the cache is physically located to the main number-crunching section of the processor. Figure 2-3 shows how the different caches work together with main memory.

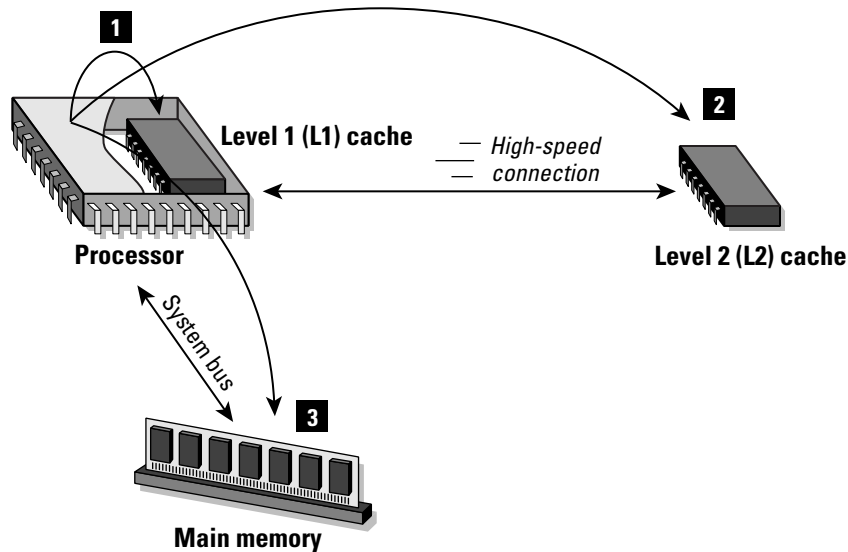


Figure 2-3: Multiple caches

They way a processor works with a system that has multiple caches is that the processor checks the L1 cache first, then the L2 cache, and then, finally, the main memory.

Traditionally, L1 cache, which is usually the smaller of the two, has been located on the processor itself and L2 cache has been located outside, but near the processor. (The physical location on a computer's motherboard does make a difference because when you shuttle data back and forth to different places, the further away something is, the longer it takes to get there and back. And when it comes to computer processing, nanoseconds — billionths of a second — really do count.)

Recent processor designs have begun to integrate L2 cache onto the processor card or into the CPU chip itself, much like L1 cache. This speeds up access to the larger L2 cache which, in turn, speeds up the computer's performance. Figure 2-4 demonstrates the differences.

Another traditional difference between L1 and L2 caches has been the speed at which the processor can access the different types of memory. Because L1 cache is integrated into the core of the microprocessor, it typically runs at the same speed as the CPU; so on a 500MHz processor, the connection speed to L1 cache is usually 500MHz. On older systems, the L2 cache often connected to the processor at the same speed as main memory. This speed is determined by a connecting route, called the computer's *system bus*, and typically runs at 66, 100, or 133MHz (although faster speeds are possible).



For more on system buses, see the “Logical connections” section later in this chapter.

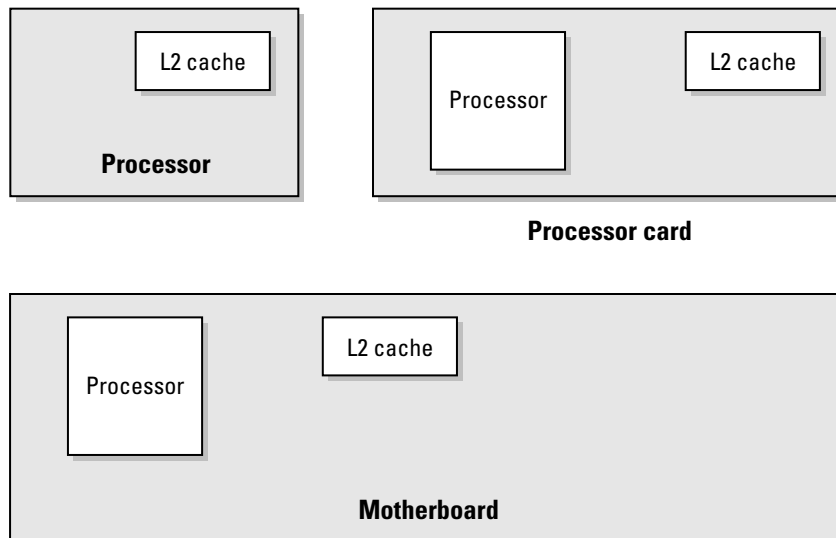


Figure 2-4: Cache locations

The L2 cache is located in different places on different processors. Some processors have the L2 cache integrated into the main chip itself, others have L2 cache on the circuit board that holds the processor, and still others work with L2 cache that's separate from the processor on the computer's motherboard.

On newer systems, however, where the L2 cache is located on a daughtercard, such as most Pentium II and Pentium IIIs, or in the processor itself, as with the Celeron A, K6-3, and some mobile Pentium IIs (sometimes called Pentium II PEs, for performance enhanced) and Pentium IIIs (those designed for notebooks), communication between the processor and the L2 cache occurs much more rapidly. On the Pentium II and III, for example, the processor-to-L2 cache connection is often via a *backside bus*; it runs faster than the system bus, but at half the speed of the processor. (This is sometimes referred to as a 1:2 ratio.) Again, with a 500MHz Pentium III processor, the processor-to-L2 cache connection speed is 250MHz. Additionally, systems that incorporate L2 cache on the chip itself feature a 1:1 ratio between the speed of the processor and the speed of the processor-to-L2 cache connection. So with a 500MHz processor, the connection to the L2 cache also runs at 500MHz.

The faster your processor is, the more important it is to have a reasonable amount of L2 cache. In fact, without the proper amount of L2 cache, a processor often sits idle, "wasting cycles" as they say, which means your computer is not running as fast it can.

This lack of L2 cache explains why some of the early Celeron-based computers had relatively poor performance. The original Celeron essentially wasted a great deal of its processing power. The upgraded Celeron A chip and all current Celerons (including the mobile versions), however, incorporate some

L2 cache on the processor itself, and dramatically improve the performance of computers using the “A” version of the Celeron.



Note

You can tell whether or not a system uses the Celeron or Celeron A because all Celerons faster than 300MHz are Celeron As and all processors slower than 300MHz are original Celerons. (The only exception is that all Celerons designed for notebooks have the integrated L2 cache, regardless of speed.) Unfortunately, desktop PC-oriented 300MHz chips were available in both Celeron and Celeron A formats, so the only way to tell 300MHz Celerons apart is to look at the computer’s documentation (or use a diagnostic program that lists the processor’s type and speed).

Because most processors incorporate L2 cache into their basic design, you often don’t have the option to choose more or less cache in the system you’d like to purchase or put together. (Older processors with standalone L2 cache are the one exception.) Instead, you get the amount of L2 cache a particular model of microprocessor includes. Therefore, when you decide which type of processor to get, make sure you find out how much L2 cache it includes.

Remember, though, you can have too much of a good thing—in other words, it’s possible to have too much cache (believe it or not . . .). Depending on the type of applications you plan to use, you can reach a point of diminishing return, and additional cache won’t improve your performance very much. Also, cache memory tends to be very expensive, so you need to balance the price vs. performance.

Computers that operate as *servers*, machines that sit at the center point of computer networks, typically need more cache than normal desktop machines because of the type of work they do. Due to this fact, several versions of the Pentium II and Pentium III Xeon, which is designed for servers, include 2MB (or more) of expensive L2 cache, as opposed to most desktop-oriented Pentium IIs and IIIs, which include only 512KB.

Architecture

In addition to their rated MegaHertz (MHz) speed, and the amount and type of cache, processors can be categorized by their internal structure, or *architecture*; it basically determines the language software programs must use to work with them. The vast majority of IBM-compatible PCs use what are called x86 processors because they are derived from Intel’s 8086 processor—the same processor found in some of the earliest IBM PCs. The very first IBM PC, however, used Intel’s 8088 processor, a predecessor to the 8086.



Note

The 8086 and 8088 are often confused because the later chip used a lower number as its product name. The reason for this is that the 8088 was an 8-bit processor, while the 8086 is a 16-bit processor (a distinction I explain below), and hence the difference in the last digit.

The x86 family of processors share a common set of instructions that software programs use to run on the chip. These instructions are the basic “language” of the processor and determine what types of calculations the processor is capable of doing.

Faking it with software

When is an x86-compatible processor not really an x86-compatible processor? When it's a Motorola processor (or one of several other types) pretending to be an x86-compatible via trickery, commonly called software emulation. Software emulation basically fools an application into thinking it's running on the type of chip that the application was originally written, even though it really isn't. The processor does this by translating one set of instructions to another.

So, for example, Insignia Solution's SoftWindows 98 or Connectix Corp.'s Virtual PC, which are programs for the Apple Macintosh, enable Mac users to run Windows programs on the Motorola processor inside the Macintosh, even though those Windows programs were written for Intel

processors. Basically, the instructions the Windows program (and even Windows itself) calls for are translated into a form that the Motorola processor understands through the software emulation program. Then the Motorola processor performs the necessary calculations, and finally, the software emulator takes those results and feeds them back to the Windows application. As far as the application is concerned, it's running under Windows on an Intel processor, as normal. Performing these translations back and forth takes time and processing power, however, and software emulators are much slower than running the same programs on PC hardware.

Other processors use different instructions. This is why programs written for the Macintosh, for example, won't work on PC-compatible machines; Mac programs use instructions that are specific to the PowerPC family of processors. If you try to run a Mac program on a computer with an x86 chip, the x86 will think you are speaking to it in a foreign language. (Not to confuse matters, but it is possible to run applications written for one type of chip architecture on another via a technology called *software emulation*.)

Each generation of x86 chips has added to the original set of core instructions that previous generations could understand, thereby expanding the capabilities of the processor. This explains why some newer applications, written to work with the latest generation of processors, won't run on older computers, even though they use the same type of chip. In other words, some applications require a Pentium or Pentium-class processor to work and won't run, for example, on a 486.

MMX, 3DNow, and Streaming SIMD Extensions

One of the most well-known extensions to these core instructions is MMX technology, which chip leader Intel introduced several years ago as the Pentium with MMX Technology (the "official" name for the Pentium MMX processor). MMX consists of 57 new instructions that processors that support the technology understand and execute. The new instructions were primarily designed to improve the performance of multimedia applications, such as computer games and entertainment titles.

In 1998, AMD developed a different set of instructions called 3DNow. It was designed not only to improve 2D games, but also to improve 3D gaming

performance. The 3DNow instructions are found in AMD's K6-2 and later processors, as well as some processors from other third-party manufacturers, such as Cyrix and Centaur (makers of the IDT WinChip).

Most recently, Intel has added Streaming SIMD (Single Instruction, Multiple Data) Extensions, or SSE, technology to its newest Pentium III processors. Streaming SIMD was designed to improve the performance of 3D games as well as improve speech recognition and other advanced applications, and it brings an additional 80 new instructions to the core language of processors that support it.



SIMD refers to a technique for performing the same operation on multiple bits of data at the same time. This can be helpful for things like making all the pixels, or individual dots, in a digital photograph all a bit darker. The original MMX instructions perform SIMD on integer data and the Streaming SIMD instructions in the Pentium III extend this to floating point data. (Integer refers to whole numbers and floating point refers to decimals where the decimal point floats.)

As great as these new additions can be, they also raise important and potentially problematic issues. On the positive side, if software programmers take advantage of these new instructions, they can make certain features in their programs run faster. But if the programmers don't provide an alternative method to perform the same operation using a non-MMX, non-3DNow, or non-Streaming SIMD set of instructions, the program won't work on machines that don't support the special extensions. Given that many of these extensions are only supported in relatively new computers, this limits their potential audience. Additionally, now that there are three different sets of extensions, figuring out which chips support which extensions and which applications work most efficiently with which processor can be very confusing. For the record, though, virtually every CPU now supports MMX. Brand new Intel chips support MMX and Streaming SIMD Extensions (SSE). AMD K6's and K7's, as well as the Cyrix' MXi and later IDT/Centaur WinChips, support MMX and 3DNow.

Thankfully, many applications that support these technologies (programs that are said to be MMX- or 3DNow- or SSE-enabled) will run on older machines because programmers provided an alternative "backdoor" capability. Programs using older instructions on a machine that doesn't support new extensions won't run as fast as they would on a computer with a chip that supports MMX, 3DNow, or SSE, but at least they'll run. The only time you run into a problem is if the program actually *requires* MMX, 3DNow or SSE. If this is the case, then this program will only run on computers with processors that support the appropriate extensions. (Whew!)

Word size

Processors also differ in the size of data chunks they can work on at any given time. The 8086 processor and the subsequent 80286 processor (often shortened to 286) are 16-bit processors, which means they can munch on 16