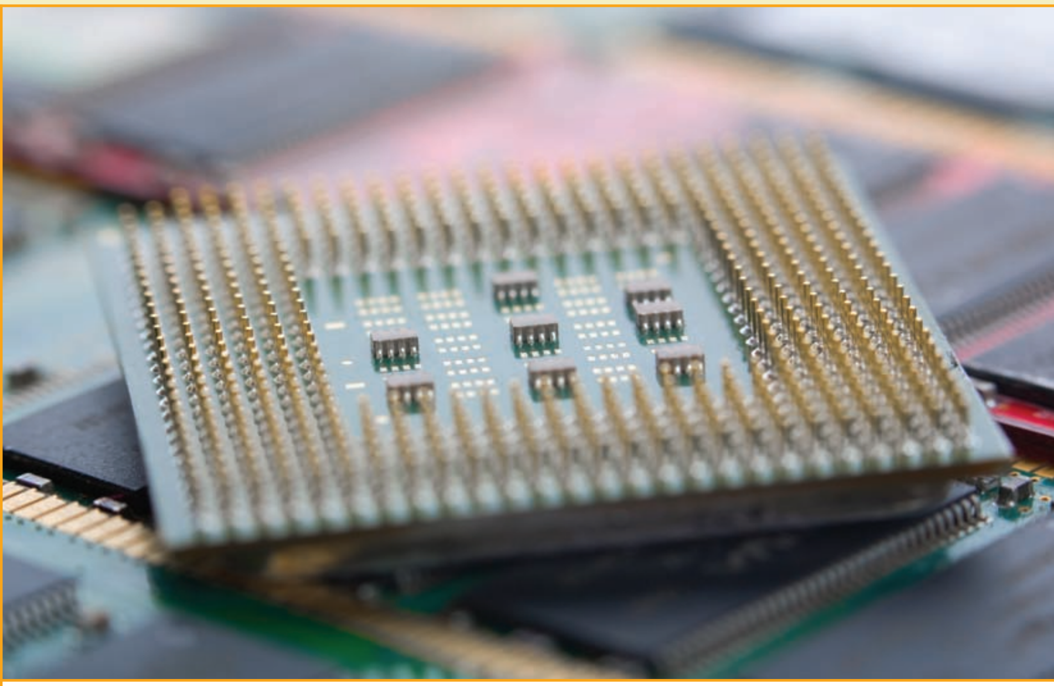
chapter 6

Microprocessors

"I'm going to give you a little advice. There's a force in the universe that makes things happen. And all you have to do is get in touch with it, stop thinking, let things happen, and be the ball."

—Ty Webb, Caddyshack



In this chapter, you will learn how to

- **Identify the core components of a CPU**
- **Describe the relationship of CPUs and memory**
- **Explain the varieties of modern CPUs**
- **Select and install a CPU**
- **Troubleshoot CPUs**

For all practical purposes, the terms **microprocessor** and **central processing unit (CPU)** mean the same thing: it's that big chip inside your computer that many people often describe as the brain of the system. You know from Chapter 3 that CPU makers name their microprocessors in a fashion similar to the automobile industry: CPUs get a make and a model, such as Intel Core i7 or AMD Phenom II X4. But what's happening inside the CPU to make it able to do the amazing things asked of it every time you step up to the keyboard?

This chapter delves into microprocessors in detail. We'll first discuss how processors work and the components that enable them to interact with the rest of the computer. The second section describes how CPUs work with memory. The third section takes you on a tour of modern CPUs. The final section gets into practical work: selecting, installing, and troubleshooting CPUs.

Historical/Conceptual

CPU Core Components

Although the computer might seem to act quite intelligently, comparing the CPU to a human brain hugely overstates its capabilities. A CPU functions more like a very powerful calculator than like a brain—but, oh, what a calculator! Today's CPUs add, subtract, multiply, divide, and move billions of numbers per second. Processing that much information so quickly makes any CPU look intelligent. It's simply the speed of the CPU, rather than actual intelligence, that enables computers to perform feats such as accessing the Internet, playing visually stunning games, or editing photos.

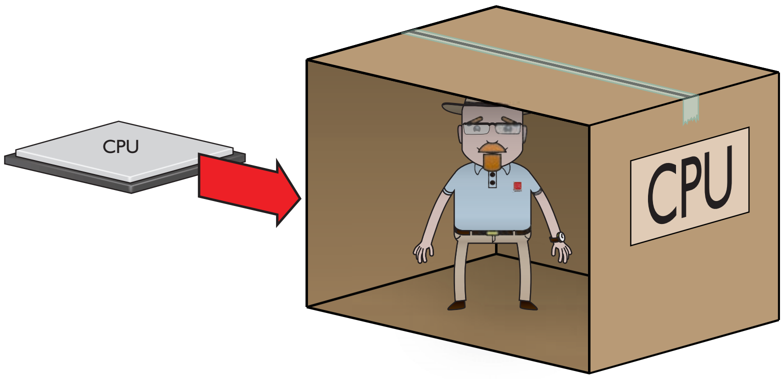
A good PC technician needs to understand some basic CPU functions to support PCs, so let's start with an analysis of how the CPU works. If you wanted to teach someone how an automobile engine works, you would use a relatively simple example engine, right? The same principle applies here. Let's begin our study of the CPU with the granddaddy of all PC CPUs: the famous Intel 8088, invented in the late 1970s. Although this CPU first appeared over 30 years ago, it defined the idea of the modern microprocessor and contains the same basic parts used in even the most advanced CPUs today. Prepare to enter that little bit of magic called the CPU.

The Man in the Box

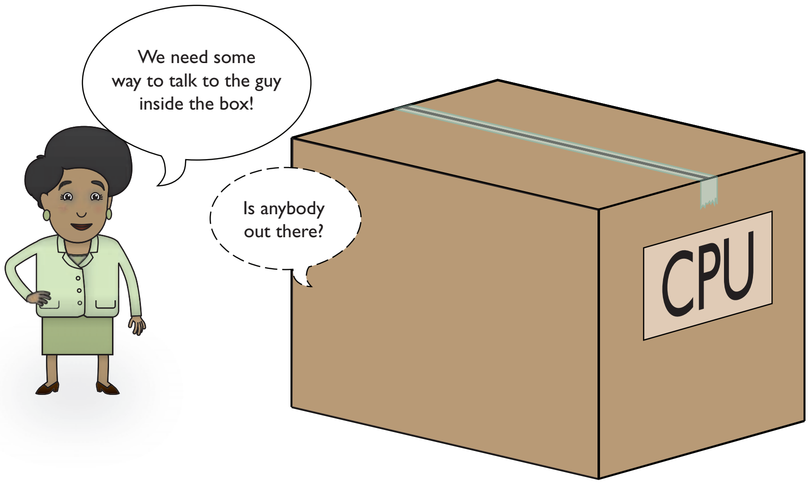
Let's begin by visualizing the CPU as a man in a box (see Figure 6.1). This is one clever guy. He can perform virtually any mathematical function, manipulate data, and give answers *very quickly*.

This guy is potentially very useful to us, but there's a catch—he lives closed up in a tiny box. Before he can work with us, we must come up with a way to exchange information with him (see Figure 6.2).

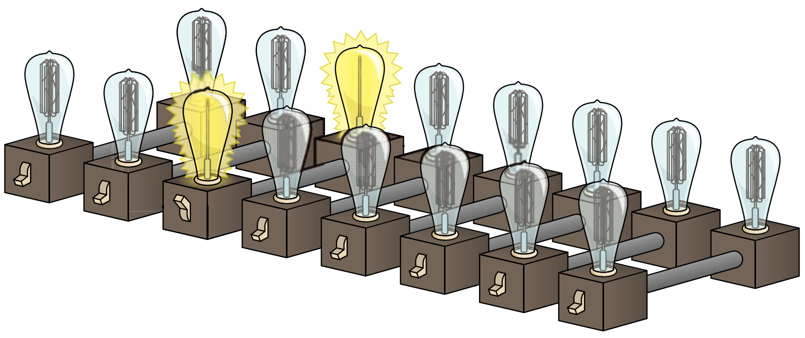
Imagine that we install a set of 16 light bulbs, 8 inside his box and 8 outside his box. Each of the 8 light bulbs inside the box connects to one of the 8 bulbs outside the box to form a pair. Each pair of light bulbs is always either on or off. You can control the 8 pairs of bulbs by using a set of 8 switches outside the box, and the Man in the Box can also



• **Figure 6.1** Imagine the CPU as a man in a box.



• **Figure 6.2** How do we talk to the Man in the Box?



• **Figure 6.3** Cutaway of the external data bus—note that one light bulb pair is on

control them by using an identical set of 8 switches inside the box. This light-bulb communication device is called the **external data bus (EDB)**.

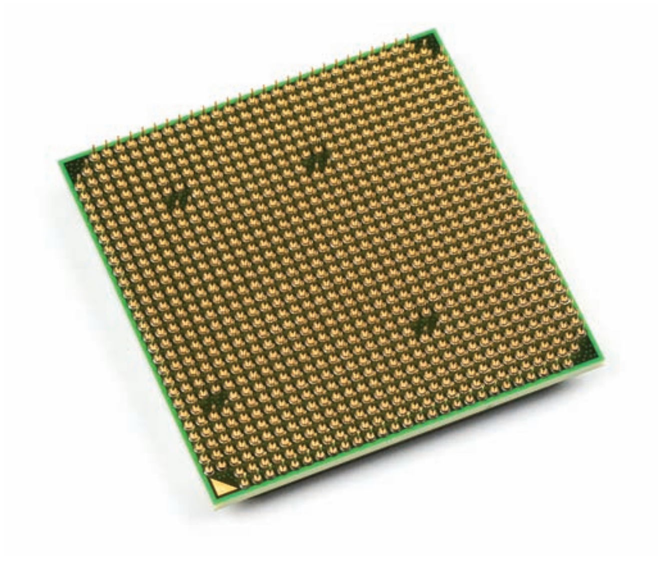
Figure 6.3 shows a cutaway view of the external data bus. When either you or the Man in the Box flips a switch on, *both* light bulbs go on, and the switch on the other side is also flipped to the on position. If you or the Man in the Box turns a switch off, the light bulbs on both sides are turned off, along with the other switch for that pair.

Can you see how this works? By creating on/off patterns with the light bulbs that represent different pieces of data or commands, you can send that information to the Man in the Box, and he can send information back in the same way—*assuming that you agree ahead of time on what the different patterns of lights mean*. To accomplish this, you need some sort of codebook that assigns meanings to the many patterns of lights that the EDB might display. Keep this thought in mind while we push the analogy a bit more.

Before going any further, make sure you're clear on the fact that this is an analogy, not reality. There really is an EDB, but you won't see any light bulbs or switches on the CPU. You can, however, see little wires sticking out of the CPU (see Figure 6.4). If you apply voltage to one of these wires, you in essence flip the switch. Get the idea? So if that wire had voltage, and if a tiny light bulb were attached to the wire, that light bulb would glow, would it not? By the same token, if the wire had no power, the light bulb would not glow. That is why the switch-and-light-bulb analogy may help you picture these little wires constantly flashing on and off.

Now that the EDB enables you to communicate with the Man in the Box, you need to see how it works by placing voltages on the wires. This brings up a naming problem. It's a hassle to say something like "on-off-on-off-on-on-off-off" when talking about which wires have voltage. Rather than saying that one of the EDB wires is on or off, use the number 1 to represent on and the number 0 to represent off (see Figure 6.5). That way, instead of describing the state of the lights as "on-off-on-off-on-on-off-off," I can instead describe them by writing "10101100."

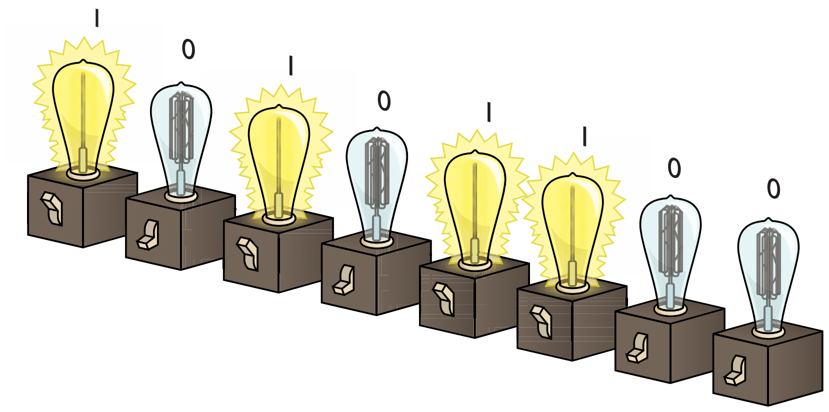
In the world of computers, we constantly turn wires on and off. As a result, we can use this "1 and 0" or **binary** system to describe the state of these wires at any given moment. (See, and you just thought computer geeks spoke in binary to confuse normal people. Ha!) There's much more to binary numbering in the world of computing, but this is a great place to start.



• **Figure 6.4** Close-up of the underside of a CPU

Registers

The Man in the Box provides good insight into the workspace inside a CPU. The EDB gives you a way to



• **Figure 6.5** Here "1" means on, "0" means off.

communicate with the Man in the Box so you can give him work to do. But to do this work, he needs a worktable; in fact, he needs at least four worktables. Each of these four worktables has 16 light bulbs. These light bulbs are not in pairs; they're just 16 light bulbs lined up straight across the table. Each light bulb is controlled by a single switch, operated only by the Man in the Box. By creating on/off patterns like the ones on the EDB, the Man in the Box can use these four sets of light bulbs to work math problems. In a real computer, these worktables are called **registers** (see Figure 6.6).

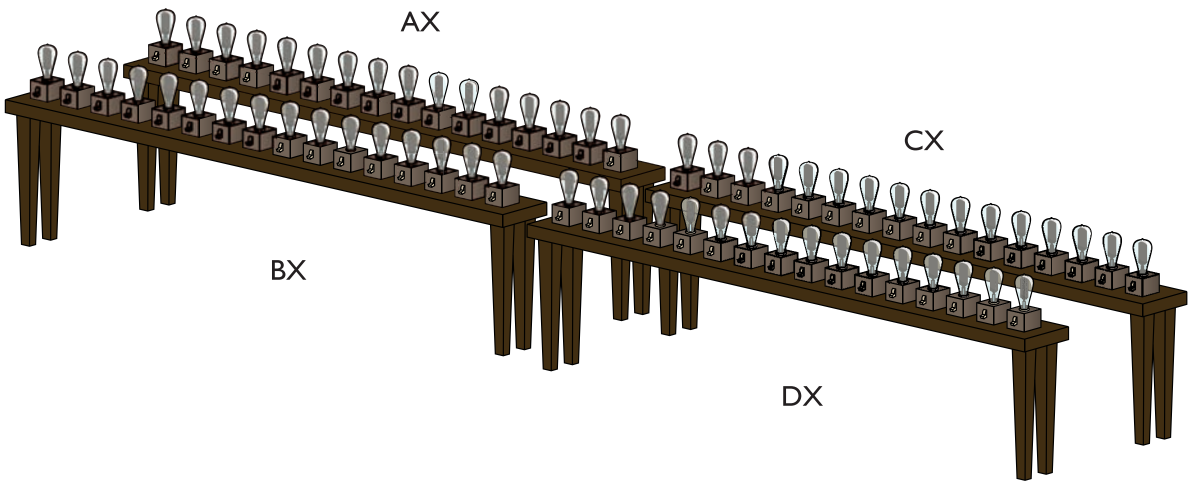
Registers provide the Man in the Box with a workplace for the problems you give him. All CPUs contain a large number of registers, but for the moment let's concentrate on the four most common ones: the *general-purpose registers*. Intel named them AX, BX, CX, and DX.

Great! We're just about ready to put the Man in the Box to work, but before you close the lid on the box, you must give the Man one more tool. Remember the codebook I mentioned earlier? Let's make one to enable us to communicate with him. Figure 6.7 shows the codebook we'll use. We'll give one copy to him and make a second for us.

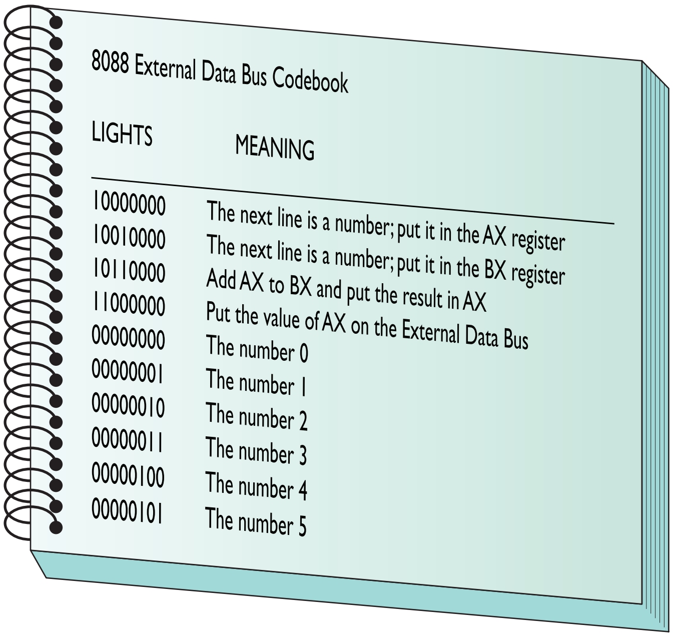
Tech Tip

General-Purpose Registers

The 8088 was the first CPU to use the four now famous AX–DX general-purpose registers, and they still exist in even the latest CPUs. (But they have a lot more light bulbs!)



• **Figure 6.6** The four general-purpose registers



8088 External Data Bus Codebook

LIGHTS	MEANING
10000000	The next line is a number; put it in the AX register
10010000	The next line is a number; put it in the BX register
10110000	Add AX to BX and put the result in AX
11000000	Put the value of AX on the External Data Bus
00000000	The number 0
00000001	The number 1
00000010	The number 2
00000011	The number 3
00000100	The number 4
00000101	The number 5

• **Figure 6.7** CPU codebook

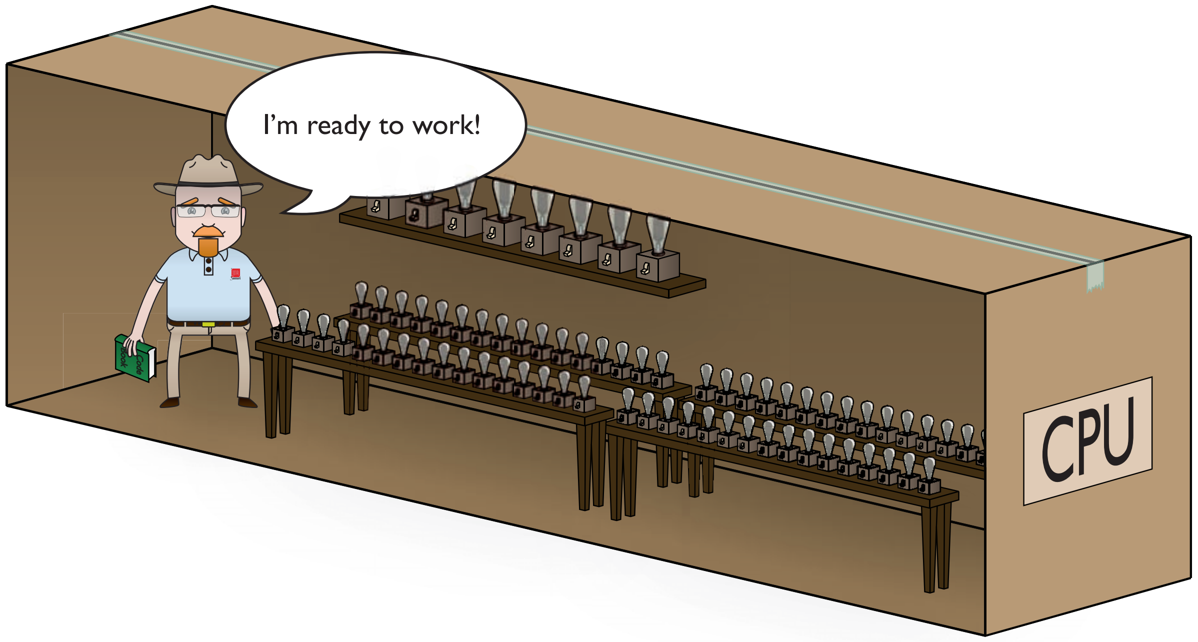
In this codebook, for example, 10000111 means *Move the number 7 into the AX register*. These commands are called the microprocessor's **machine language**. The commands listed in the figure are not actual commands; as you've probably guessed, I've simplified dramatically. The Intel 8088 CPU actually used commands very similar to these, plus a few hundred others.

Here are some examples of real machine language for the Intel 8088:

10111010	The next line of code is a number. Put that number into the DX register.
01000001	Add 1 to the number already in the CX register.
00111100	Compare the value in the AX register with the next line of code.

By placing machine language commands—called *lines of code*—onto the EDB one at a time, you can instruct the Man in the Box to do specific tasks. All of the machine language commands that the CPU understands make up the CPU's **instruction set**.

So here is the CPU so far: the Man in the Box can communicate with the outside world via the EDB; he has four registers he can use to work on the problems you give him; and he has a codebook—the instruction set—so he can understand the different patterns (machine language commands) on the EDB (see Figure 6.8).



• **Figure 6.8** The CPU so far

Clock

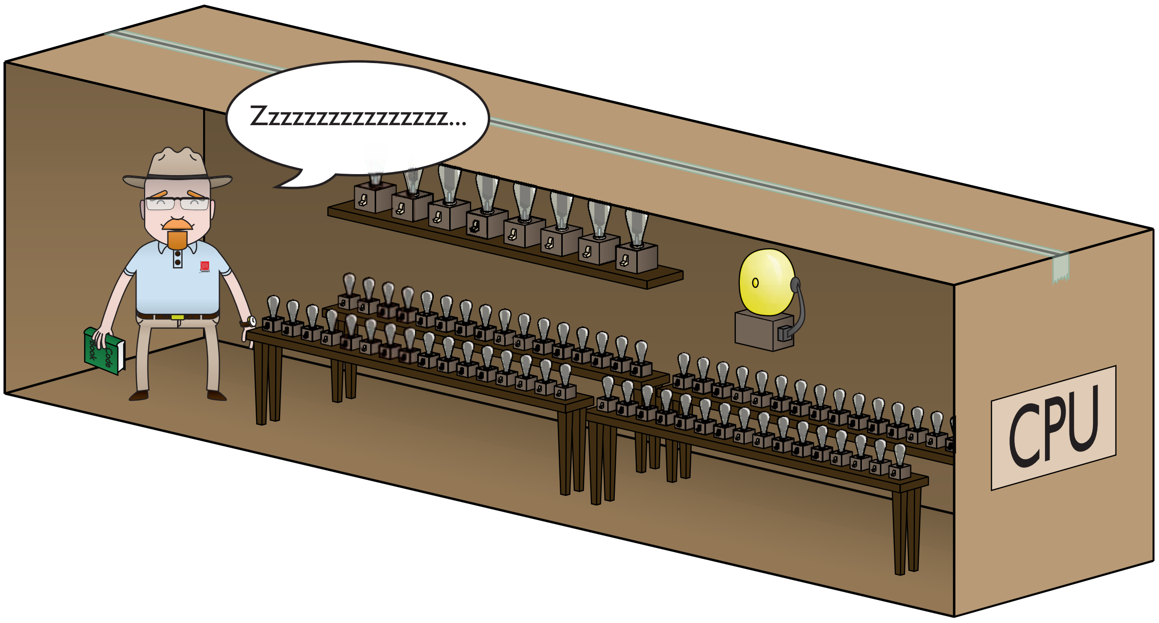
Okay, so you're ready to put the Man in the Box to work. You can send the first command by lighting up wires on the EDB. How does he know when you've finished setting up the wires and it's time to act?

Have you ever seen one of those old-time manual calculators with the big crank on one side? To add two numbers, you pressed a number key, the + key, and another number key, but then to make the calculator do the calculation and give you the answer, you had to pull down the crank. That was the signal that you had finished entering data and instructions and were ready for the calculator to give you an answer.

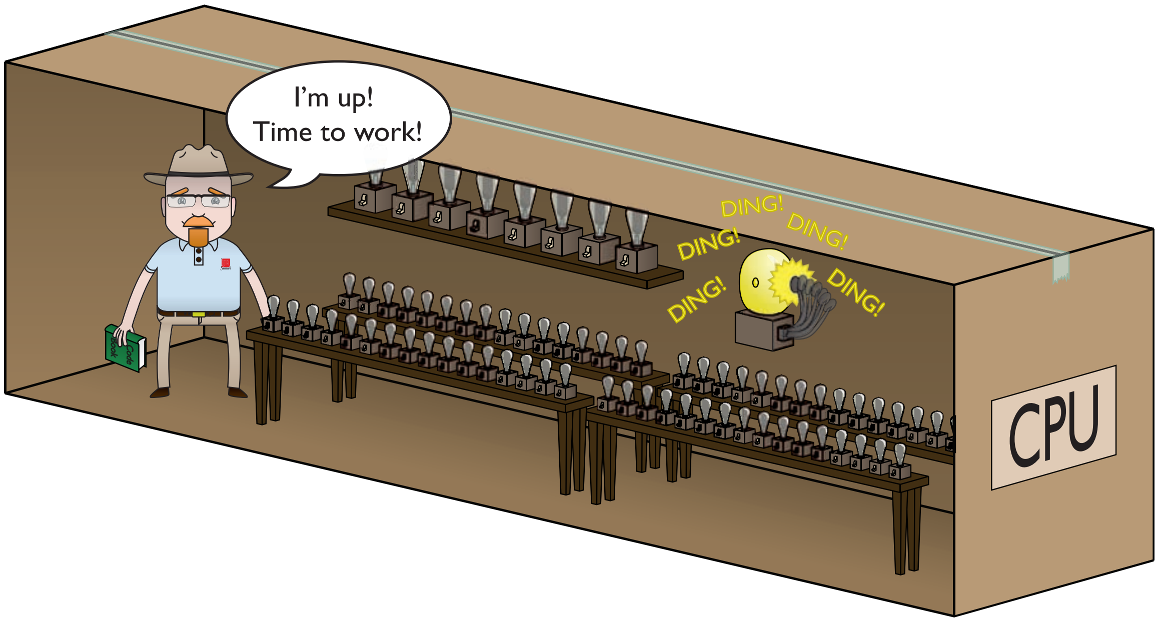
A CPU also has a type of crank. To return to the Man in the Box, imagine there's a bell inside the box activated by a button on the outside of the box. Each time you press the button to sound the bell, the Man in the Box reads the next set of lights on the EDB. Of course, a real computer doesn't use a bell. The bell on a real CPU is a special wire called the **clock wire** (most diagrams label the clock wire CLK). A charge on the CLK wire tells the CPU that another piece of information is waiting to be processed (see Figure 6.9).

For the CPU to process a command placed on the EDB, a certain minimum voltage must be applied to the CLK wire. A single charge to the CLK wire is called a **clock cycle**. Actually, the CPU requires at least two clock cycles to act on a command, and usually more. Using the manual calculator analogy, you need to pull the crank at least twice before anything happens. In fact, a CPU may require hundreds of clock cycles to process some commands (see Figure 6.10).

The maximum number of clock cycles that a CPU can handle in a given period of time is referred to as its **clock speed**. The clock speed is the fastest speed at which a CPU can operate, determined by the CPU manufacturer. The Intel 8088 processor had a clock speed of 4.77 MHz (4.77 million cycles



• **Figure 6.9** The CPU does nothing until activated by the clock.



• **Figure 6.10** The CPU often needs more than one clock cycle to get a result.

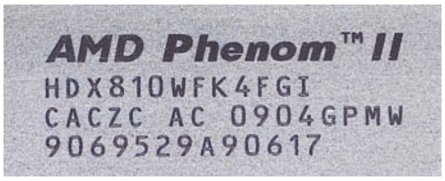
per second), extremely slow by modern standards, but still a pretty big number compared to using a pencil and paper. CPUs today run at speeds in excess of 3 GHz (3 billion cycles per second). You'll see these "hertz" terms a lot in this chapter, so here's what they mean:

- 1 hertz (1 Hz) = 1 cycle per second
- 1 megahertz (1 MHz) = 1 million cycles per second
- 1 gigahertz (1 GHz) = 1 billion cycles per second

A CPU's clock speed is its *maximum* speed, not the speed at which it *must* run. A CPU can run at any speed, as long as that speed does not exceed its clock speed. Manufacturers used to print the CPU's clock speed directly onto the CPU, but for the past few years they've used cryptic codes (see Figure 6.11). As the chapter progresses you'll see why they do this.

The **system crystal** determines the speed at which a CPU and the rest of the PC operate. The system crystal is usually a quartz oscillator, very similar to the one in a wristwatch, soldered to the motherboard (see Figure 6.12).

The quartz oscillator sends out an electric pulse at a certain speed, many millions of times per second. This signal goes first to a clock chip that adjusts the pulse, usually increasing the pulse sent by the crystal by some large multiple. (The folks who make motherboards could connect the crystal directly to the CPU's clock wire, but then if you wanted to replace your CPU with a CPU with a different clock speed, you'd need to replace the crystal too.) As long as the PC is turned on, the quartz oscillator, through the clock chip,



AMD Phenom™ II
HDX810WFK4FGI
CACZC AC 0904GPMW
9069529A90617

• **Figure 6.11** Where is the clock speed?

Tech Tip

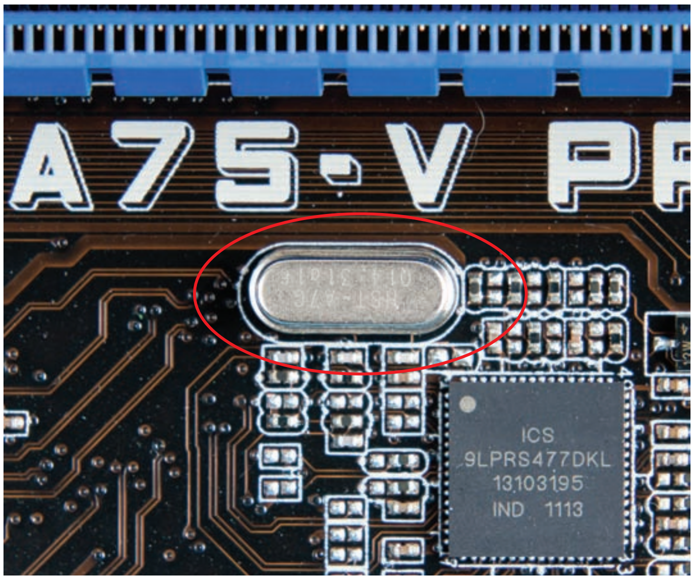
Processor Variations

CPU makers sell the exact make and model of CPU at a number of different speeds. All of these CPUs come off of the same assembly lines, so why do they have different speeds? Every CPU comes with subtle differences—flaws, really—in the silicon that makes one CPU run faster than another. The speed difference comes from testing each CPU to see what speed it can handle.

Aggressive users sometimes intentionally overclock CPUs by telling the clock chip to multiply the pulse faster than the CPU's designed speed. They do this to make slower (cheaper) CPUs run faster and to get more performance in demanding programs. See the "Overclocking" section later in this chapter.

fires a charge on the CLK wire, in essence pushing the system along.

Visualize the system crystal as a metronome for the CPU. The quartz oscillator repeatedly fires a charge on the CLK wire, setting the beat, if you will, for the CPU's activities. If the system crystal sets a beat slower than the CPU's clock speed, the CPU will work just fine, though at the slower speed of the system crystal. If the system crystal forces the CPU to run faster than its clock speed, it can overheat and stop working. Before you install a CPU into a system, you must make sure that the crystal and clock chip send out the correct clock pulse for that particular CPU. In the old days, this required very careful adjustments. With today's systems, the motherboard talks to the CPU. The CPU tells the motherboard the clock speed it needs, and the clock chip automatically adjusts for the CPU, making this process now invisible.



• **Figure 6.12** One of many types of system crystals

Back to the External Data Bus

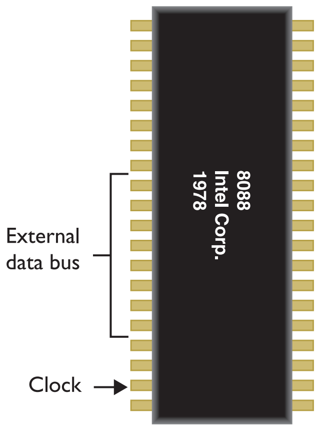
One more reality check. We've been talking about tables with racks of light bulbs, but of course real CPU registers don't use light bulbs to represent on/1 and off/0. Registers are tiny storage areas on the CPU made up of microscopic semiconductor circuits that hold charges. It's just easier to imagine a light bulb lit up to represent a circuit holding a charge; when the light bulb is off, there is no charge.

Figure 6.13 is a diagram of a real 8088 CPU, showing the wires that comprise the external data bus and the single clock wire. Because the registers are inside the CPU, you can't see them in this figure.

Now that you have learned what components are involved in the process, try the following simple exercise to see how the process works. In this example, you tell the CPU to add 2 + 3. To do this, you must send a series of commands to the CPU; the CPU will act on each command, eventually giving you an answer. Refer to the codebook in Figure 6.7 to translate the instructions you're giving the Man in the Box into binary commands.

Did you try it? Here's how it works:

1. Place 10000000 on the external data bus (EDB).
2. Place 00000010 on the EDB.
3. Place 10010000 on the EDB.
4. Place 00000011 on the EDB.
5. Place 10110000 on the EDB.
6. Place 11000000 on the EDB.



• **Figure 6.13** Diagram of an Intel 8088 showing the external data bus and clock wires

When you finish step 6, the value on the EDB will be 00000101, the decimal number 5 written in binary.

Congrats! You just added 2+3 by using individual commands from the codebook. This set of commands is known as a **program**, which is a series of commands sent to a CPU in a specific order for the CPU to perform work. Each discrete setting of the EDB is a line of code. This program, therefore, has six lines of code.

Memory

Now that you've seen how the CPU executes program code, let's work backward in the process for a moment and think about how the program code gets to the external data bus. The program itself is stored on the hard drive. In theory, you could build a computer that sends data from the hard drive directly to the CPU, but there's a problem—the hard drive is too slow. Even the ancient 8088, with its clock speed of 4.77 MHz, could conceivably process several million lines of code every second. Modern CPUs crank out billions of lines every second. Hard drives simply can't give the data to the CPU at a fast enough speed.

Computers need some other device that takes copies of programs from the hard drive and then sends them, one line at a time, to the CPU quickly enough to keep up with its demands. Because each line of code is nothing more than a pattern of eight ones and zeros, any device that can store ones and zeros eight-across will do. Devices that in any way hold ones and zeros that the CPU accesses are known generically as **memory**.

Many types of devices store ones and zeros perfectly well—technically even a piece of paper counts as memory—but computers need memory that does more than just store groups of eight ones and zeros. Consider this pretend program:

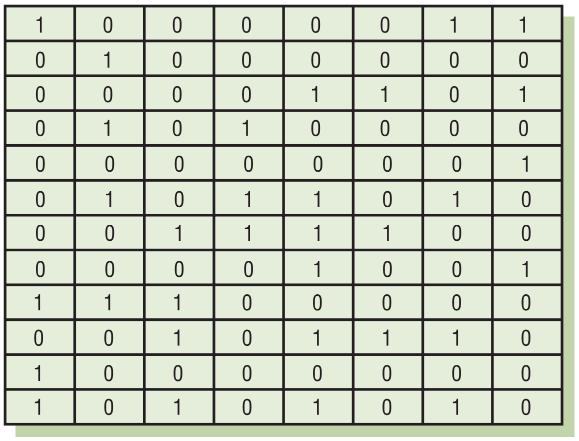
1. Put 2 in the AX register.
2. Put 5 in the BX register.
3. If AX is greater than BX, run line 4; otherwise, go to line 6.
4. Add 1 to the value in AX.
5. Go back to line 1.
6. Put the value of AX on the EDB.

This program has an IF statement, also called a *branch* by CPU makers. The CPU needs a way to address each line of this memory—a way for the CPU to say to the memory, "Give me the next line of code" or "Give me line 6." Addressing memory takes care of another problem: the memory must not only store programs but also store the result of the programs. If the CPU adds 2 + 3 and gets 5, the memory needs to store that 5 in such a way that other programs may later read that 5, or possibly even store that 5 on a hard drive. By addressing each line of memory, other programs will know where to find the data.

Memory and RAM

Memory must store not only programs, but also data. The CPU needs to be able to read and write to this storage medium. Additionally, this system must enable the CPU to jump to *any* line of stored code as easily as to any other line of code. All of this must be done at or at least near the clock speed of the CPU. Fortunately, this magical device has existed for many years: **random access memory (RAM)**.

Chapter 7 develops the concept of RAM in detail, so for now let's look at RAM as an electronic spreadsheet, like one you can generate in Microsoft Excel (see Figure 6.14). Each cell in this spreadsheet can store only a one or a zero. Each cell is called a **bit**. Each row in the spreadsheet is eight bits across to match the EDB of the 8088. Each row of eight bits is called a **byte**. In the PC world, RAM transfers and stores data to and from the CPU in byte-sized chunks. RAM is therefore arranged in byte-sized rows. Here are the terms used to talk about quantities of bits:



1	0	0	0	0	0	1	1
0	1	0	0	0	0	0	0
0	0	0	0	1	1	0	1
0	1	0	1	0	0	0	0
0	0	0	0	0	0	0	1
0	1	0	1	1	0	1	0
0	0	1	1	1	1	0	0
0	0	0	0	1	0	0	1
1	1	1	0	0	0	0	0
0	0	1	0	1	1	1	0
1	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0

• **Figure 6.14** RAM as a spreadsheet

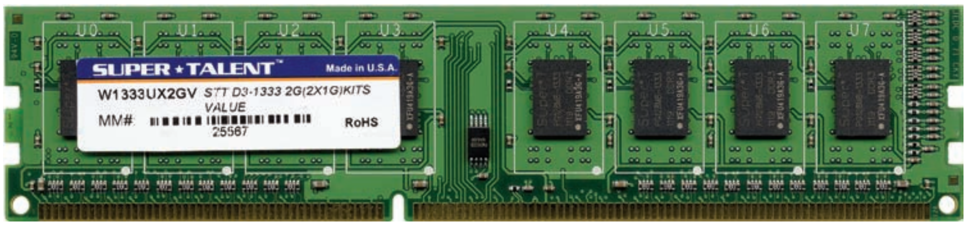
- Any individual 1 or 0 = a bit
- 4 bits = a nibble
- 8 bits = a byte
- 16 bits = a word
- 32 bits = a double word
- 64 bits = a paragraph or quad word

The number of bytes of RAM varies from PC to PC. In earlier PCs, from around 1980 to 1990, the typical system would have only a few hundred thousand bytes of RAM. Today's systems often have billions of bytes of RAM.

Let's stop here for a quick reality check. Electronically, RAM looks like a spreadsheet, but real RAM is made of groups of semiconductor chips soldered onto small cards that snap into your computer (see Figure 6.15). In Chapter 7, you'll see how these groups of chips actually make themselves look like a spreadsheet. For now, don't worry about real RAM and just stick with the spreadsheet idea.

The CPU accesses any one row of RAM as easily and as fast as any other row, which explains the "random access" part of RAM. Not only is RAM randomly accessible, it's also fast. By storing programs on RAM, the CPU can access and run them very quickly. RAM also stores any data that the CPU actively uses.

Computers use **dynamic RAM (DRAM)** for the main system memory. DRAM needs both a constant electrical charge and a periodic refresh of the circuits; otherwise, it loses data—that's what makes it dynamic rather than static in content.



• **Figure 6.15** Typical RAM

The refresh can cause some delays, because the CPU has to wait for the refresh to happen, but modern CPU manufacturers have clever ways to get by this issue, as you'll see when you read about modern processor technology later in this chapter.

Don't confuse RAM with mass storage devices such as hard drives and flash drives. You use hard drives and flash drives to store programs and data permanently. Chapters 11–13 discuss permanent storage in intimate detail.

Cross Check

RAM

You learned a few essentials about RAM in Chapter 3, so check back in that chapter and see if you can answer these questions. How is RAM packaged? Where do you install the PC's primary RAM, called the *system RAM*, in the system unit?

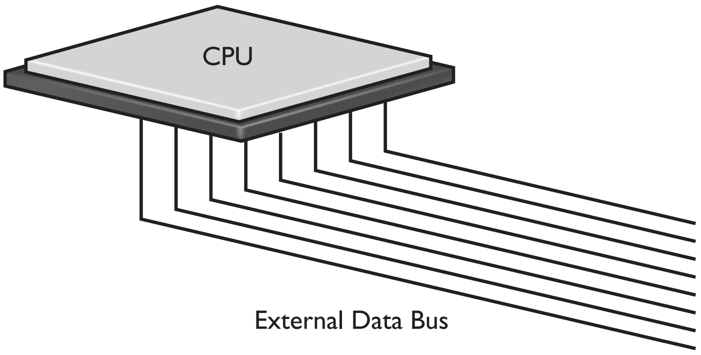
Address Bus

So far, the entire PC consists of only a CPU and RAM. But the CPU and the RAM need some connection so they can talk to each other. To do so, extend the external data bus from the CPU so it can talk to the RAM (see Figure 6.16).

Wait a minute. This is not a matter of just plugging the RAM into the EDB wires! RAM is a spreadsheet with thousands and thousands of discrete rows, and you need to look at the contents of only one row of the spreadsheet at a time, right? So how do you connect the RAM to the EDB in such a way that the CPU can see any one given row but still give the CPU the capability to look at *any* row in RAM? We need some type of chip between the RAM and the CPU to make the connection. The CPU needs to be able to say which row of RAM it wants, and the chip should handle the mechanics of retrieving that row of data from the RAM and putting it on the EDB. Wouldn't you know I just happen to have such a chip? This chip comes with many names, but for right now just call it the **memory controller chip (MCC)**.

The MCC contains special circuitry so it can grab the contents of any single line of RAM and place that data or command on the EDB. This in turn enables the CPU to act on that code (see Figure 6.17).

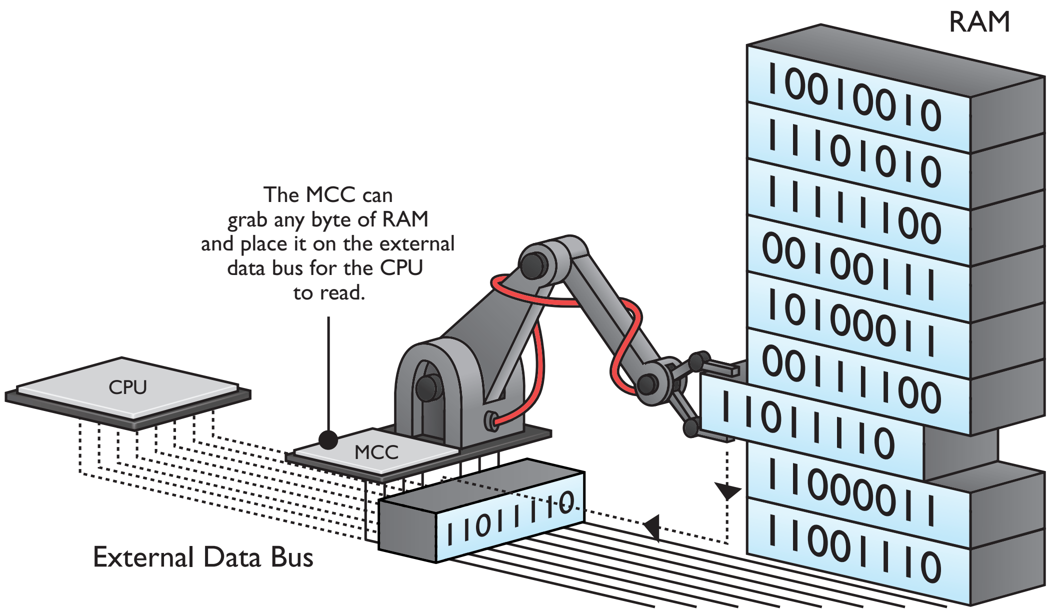
Once the MCC is in place to grab any discrete byte of RAM, the CPU needs to be able to tell the MCC which line of code it needs. The CPU therefore gains a second set of wires, called the **address bus**, with which it can



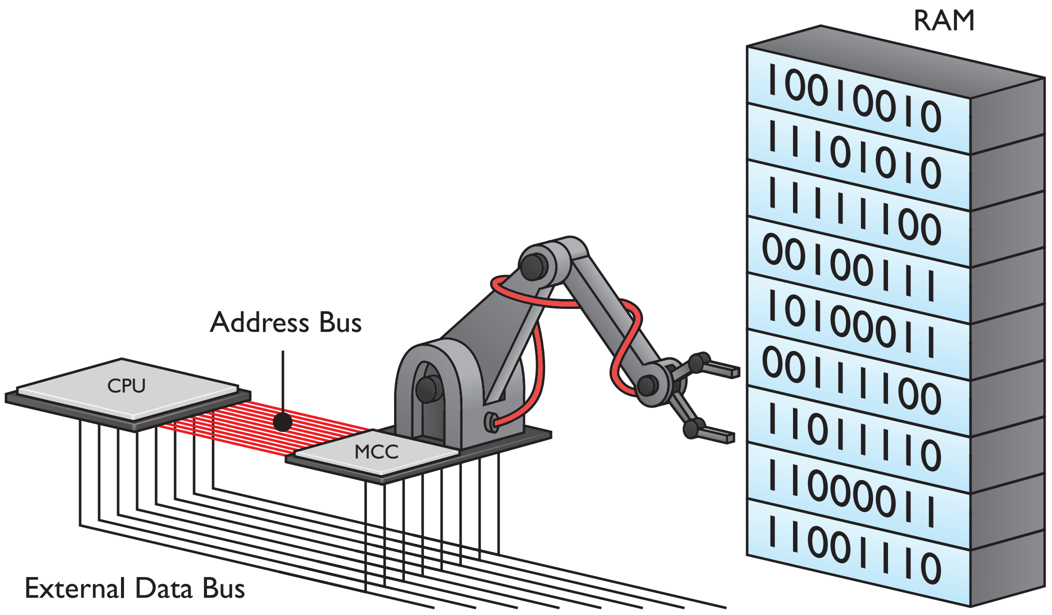
• **Figure 6.16** Extending the EDB

communicate with the MCC. Different CPUs have different numbers of wires (which, you will soon see, is very significant). The 8088 had 20 wires in its address bus (see Figure 6.18).

By turning the address bus wires on and off in different patterns, the CPU tells the MCC which line of RAM it wants at any given moment. Every different pattern of ones and zeros on these 20 wires points to one byte of RAM. There are two big questions here. First, how many different patterns of on-and-off wires can exist with 20 wires? And second, which pattern goes to which row of RAM?



• **Figure 6.17** The MCC grabs a byte of RAM.



• **Figure 6.18** Address bus

How Many Patterns?

Mathematics can answer the first question. Each wire in the address bus exists in only one of two states: on or off. If the address bus consisted of only one wire, that wire would at any given moment be either on or off. Mathematically, that gives you (pull out your old pre-algebra books) $2^1 = 2$ different combinations. If you have two address bus wires, the address bus wires create $2^2 = 4$ different combinations. If you have 20 wires, you would have 2^{20} (or 1,048,576) combinations. Because each pattern points to one line of code and each line of RAM is one byte, if you know the number of wires in the CPU's address bus, you know the maximum amount of RAM that a particular CPU can handle.

Because the 8088 had a 20-wire address bus, the most RAM it could handle was 2^{20} or 1,048,576 bytes. The 8088, therefore, had an *address space* of 1,048,576 bytes. This is not to say that every computer with an 8088 CPU had 1,048,576 bytes of RAM. Far from it! The original IBM PC only had a measly 64 kilobytes—but that was considered plenty back in the Dark Ages of Computing in the early 1980s.

Okay, so you know that the 8088 had 20 address wires and a total address space of 1,048,576 bytes. Although this is accurate, no one uses such an exact term to discuss the address space of the 8088. Instead you say that the 8088 had one *megabyte* (1 MB) of address space.

What's a "mega"? Well, let's get some terminology down. Dealing with computers means constantly dealing with the number of patterns a set of wires can handle. Certain powers of 2 have names used a lot in the computing world. The following list explains.

Tech Tip

Kilos

Of course, 1 kilo is equal to 1000 when you talk in terms of the metric system. It also means 1000 when you talk about the clock speed of a chip, so 1 KHz is equal to 1000 Hz. When you talk storage capacity, though, the binary numbers kick in, making 1 KB = 1024 bytes. Got it?

This same bizarre dual meaning applies all the way up the food chain, so 1 MHz is 1,000,000 Hz, but 1 MB is 1,048,576 bytes; 1 GHz is 1 billion Hz, but 1 GB is 1,073,741,824 bytes; and so on.

1 kilo = 2^{10} = 1024 (abbreviated as "K")

1 kilobyte = 1024 bytes (abbreviated as "KB")

1 mega = 2^{20} = 1,048,576 (abbreviated as "M")

1 megabyte = 1,048,576 bytes (abbreviated as "MB")

1 giga = 2^{30} = 1,073,741,824 (abbreviated as "G")

1 gigabyte = 1,073,741,824 bytes (abbreviated as "GB")

1 tera = 2^{40} = 1,099,511,627,776 (abbreviated as "T")

1 terabyte = 1,099,511,627,776 bytes (abbreviated as "TB")

1 kilo is *not* equal to 1000 (one thousand)

1 mega is *not* equal to 1,000,000 (one million)

1 giga is *not* equal to 1,000,000,000 (one billion)

1 tera is *not* equal to 1,000,000,000,000 (one trillion)

(But they are pretty close!)

Which Pattern Goes to Which Row?

The second question is a little harder: "Which pattern goes to which row of RAM?" To understand this, let's take a moment to discuss binary counting. In binary, only two numbers exist, 0 and 1, which makes binary a handy way to work with wires that turn on and off. Let's try to count in binary: 0, 1…what's next? It's not 2—you can only use zeros and ones. The next number after 1 is 10! Now let's count in binary to 1000: 0, 1, 10, 11, 100, 101, 110, 111, 1000. Try counting to 10000. Don't worry; it hardly takes any time at all.

Super; you now count in binary as well as any math professor. Let's add to the concept. Stop thinking about binary for just a moment and think about good old base 10 (regular numbers). If you have the number 365, can you put zeros in front of the 365, like this: 000365? Sure you can—it doesn't change the value at all. The same thing is true in binary. Putting zeros in front of a value doesn't change a thing! Let's count again to 1000 in binary. In this case, add enough zeros to make 20 places:

```
00000000000000000000
00000000000000000001
00000000000000000010
00000000000000000011
00000000000000000100
00000000000000000101
00000000000000000110
00000000000000000111
00000000000000001000
```

Hey, wouldn't this be a great way to represent each line of RAM on the address bus? The CPU identifies the first byte of RAM on the address bus with 00000000000000000000. The CPU identifies the last RAM row with 11111111111111111111. When the CPU turns off all of the address bus wires, it wants the first line of RAM; when it turns on all of the wires, it wants the 1,048,576th line of RAM. Obviously, the address bus also addresses all of the rows of RAM in between. So, by lighting up different patterns of ones and zeros on the address bus, the CPU can access any row of RAM it needs.

Bits and bytes are abbreviated differently. Bits get a lowercase b, whereas bytes get a capital B. So for example, 4 Kb is four kilobits, but 4 KB is four kilobytes.

801

Modern CPUs

CPU manufacturers have achieved stunning progress with microprocessors since the days of the Intel 8088, and the rate of change doesn't show any signs of slowing. At the core, though, today's CPUs function similarly to the processors of your forefathers. The arithmetic logic unit (ALU)—that's the Man in the Box—still crunches numbers many millions of times per second. CPUs rely on memory to feed them lines of programming as quickly as possible.

This section brings the CPU into the present. We'll first look at models you can buy today, and then we'll turn to essential improvements in technology you should understand.

Manufacturers

When IBM awarded Intel the contract to provide the CPUs for its new IBM PC back in 1980, it established for Intel a virtual monopoly on all PC CPUs. The other home-computer CPU makers of the time faded away: MOS Technology, Zilog, Motorola—no one could compete directly with Intel. Over time, other competitors have risen to challenge Intel's market-segment share dominance. In particular, a company called Advanced Micro Devices (AMD) began to make clones of Intel CPUs, creating an interesting and rather cutthroat competition with Intel that lasts to this day.

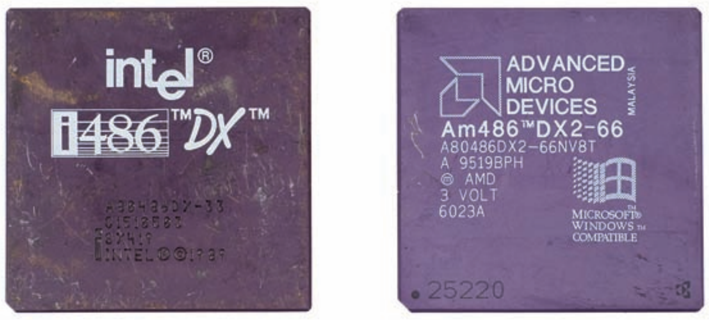
Intel

Intel Corporation thoroughly dominated the personal computer market with its CPUs and motherboard support chips. At nearly every step in the evolution of the PC, Intel has led the way with technological advances and surprising flexibility for such a huge corporation. Intel CPUs—and more specifically, their instruction sets—define the personal computer. Intel currently produces a dozen or so models of CPU for both desktop and portable computers. Most of Intel's desktop and laptop processors are sold under the Celeron, Pentium, and Core brands. Their very low-power portable/smartphone chips are branded Atom; their high-end workstation/server chips are called Xeon and Itanium.

AMD

You can't really talk about CPUs without mentioning Advanced Micro Devices. AMD makes superb CPUs for the PC market and provides competition that keeps Intel on its toes. Like Intel, AMD doesn't just make CPUs, but their CPU business is certainly the part that the public notices. AMD has made CPUs that clone the function of Intel CPUs. If Intel invented the CPU used in the original IBM PC, how could AMD make clone CPUs without getting sued? Well, chipmakers have a habit of exchanging technologies through cross-license agreements. Way back in 1976, AMD and Intel signed just such an agreement, giving AMD the right to copy certain types of CPUs.

The trouble started with the Intel 8088. Intel needed AMD's help to supply enough CPUs to satisfy IBM's demands. But after a few years, Intel had grown tremendously and no longer wanted AMD to make CPUs. AMD said, "Too bad. See this agreement you signed?" Throughout the 1980s and into the 1990s, AMD made pin-for-pin identical CPUs that matched the Intel lines of CPUs (see Figure 6.19). You could yank an Intel CPU out of a system and snap in an AMD CPU—no problem!



• Figure 6.19 Identical Intel and AMD 486 CPUs from the early 1990s

In January 1995, after many years of legal wrangling, Intel and AMD settled and decided to end the licensing agreements. As a result of this settlement, AMD chips are no longer compatible with sockets or motherboards made for Intel CPUs—even though in some cases the chips look similar. Today, if you want to use an AMD CPU, you must purchase a motherboard designed for AMD CPUs. If you want to use an Intel CPU, you must purchase a motherboard designed for Intel CPUs. So you now have a choice: Intel or AMD.

Model Names

Intel and AMD differentiate product lines by using different product names, and these names have changed over the years. For a long time, Intel used *Pentium* for its flagship model, just adding model numbers to show successive generations—Pentium, Pentium II, Pentium III, and so on. AMD used the *Athlon* brand in a similar fashion.

Most discussions on PC CPUs focus on four end-product lines: desktop PC, budget PC, portable PC, and server computers. Table 6.1 displays many of the current product lines and names.

Table 6.1 Current Intel and AMD Product Lines and Names

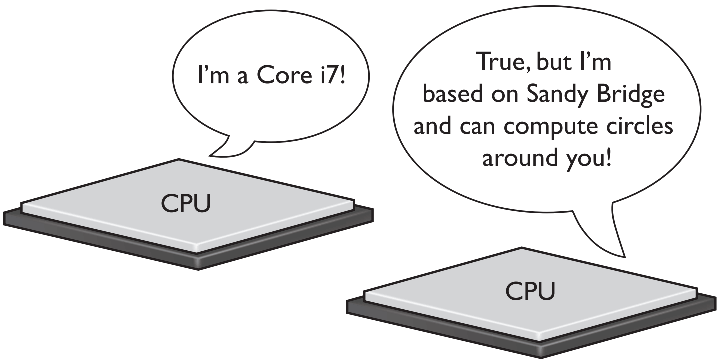
Market	Intel	AMD
Mainstream and enthusiast desktop	Core i7/i5/i3, Core 2 Duo	Phenom II, A-Series, Phenom, Athlon X2
Budget desktop	Pentium, Celeron	Sempron, Athlon II
Mobile	Core i7/i5/i3 (mobile), Core 2 Duo (mobile), Atom	Turion
Server	Xeon/Itanium	Opteron

You'll notice that both Intel and AMD reuse model names for products aimed at different markets. Yesterday's Pentium brand used to be for the highest end, for example, but now Intel uses the brand for its budget market. The same thing happened to the Athlon brand. To add a little more confusion, the budget CPUs are not the older CPUs still being sold, but low-end versions of current model lines.

Code Names

Both Intel and AMD continue to refine the CPU manufacturing process after releasing a new model, but they try to minimize the number of model names in use. This means that they release CPUs labeled as the same model, but the CPUs inside can be very different from earlier versions of that model. Both companies use **code names** to keep track of different variations within models (see Figure 6.20). As a tech, you need to know both the models and code names to be able to make proper recommendations for your clients. One example illustrates the need: the Intel Core i7.

Intel released the first Core i7 in the summer of 2008. By spring of 2012, the original microarchitecture—code


I'm a Core i7!
True, but I'm based on Sandy Bridge and can compute circles around you!
CPU
CPU

• **Figure 6.20** Same branding, but different capabilities named Nehalem—had gone through five variations, none of which worked on motherboards designed for one of the other variations. Plus, in 2011, Intel introduced the Sandy Bridge version of the Core i7 that eventually had two desktop versions and a mobile version, all of which used still other sockets. (And I'm simplifying the variations here.)

At this point, a lot of new techs throw their hands in the air. How do you keep up? How do you know which CPU will give your customer the best value for his or her money and provide the right computing firepower for his or her needs? Simply put, you need to *research efficiently*.

Your first stop should be the manufacturers' Web sites. Both companies put out a lot of information on their products.

- www.intel.com
- www.amd.com

Wikipedia is a user-generated, self-regulated resource. I've found it to be accurate on technical issues the vast majority of the time, but you should always check other references as well. Nicely, most article authors on the site will tell you their sources through footnotes. You can often use the Wikipedia articles as jump-off points for deeper searches.

You can also find many high-quality tech Web sites devoted to reporting on the latest CPUs. When a client needs an upgrade, surf the Web for recent articles and make comparisons. Because you'll understand the underlying technology from your CompTIA A+ studies, you'll be able to follow the conversations with confidence. Here's a list of some of the sites I use:

- www.arstechnica.com
- www.hardocp.com
- www.tomshardware.com
- www.bit-tech.net

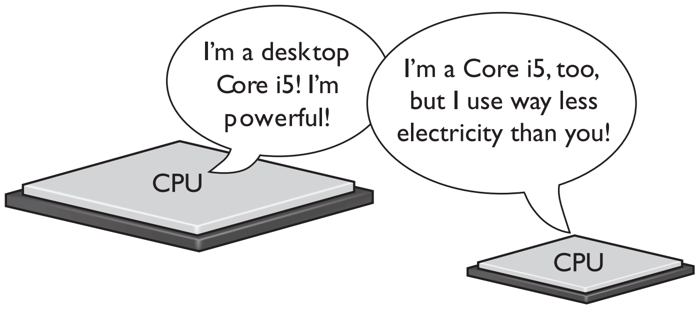
Finally, you can find great, exhaustive articles on all things tech at Wikipedia:

- www.wikipedia.org

Desktop Versus Mobile

Mobile devices, such as portable computers, have needs that differ from those of desktop computers, notably the need to consume as little electricity as possible. This helps in two ways: extending battery charge and creating less heat.

Both Intel and AMD have engineers devoted to making excellent mobile versions of their CPUs that sport advanced energy-saving features (see Figure 6.21). Intel's SpeedStep technology, for example, enables the


I'm a desktop Core i5! I'm powerful!
I'm a Core i5, too, but I use way less electricity than you!
CPU
CPU

• **Figure 6.21** Desktop vs. mobile, fight!

CPU to run in very low power mode and scale up automatically if the user demands more power from the CPU. If you're surfing the Web at an airport terminal, the CPU doesn't draw too much power. When you switch to playing an action game, the CPU kicks into gear. Saving energy by making the CPU run more slowly when demand is light is generically called **throttling**.

Many of the technologies developed for mobile processors have migrated into their more power-hungry desktop siblings, too. That's an added bonus for the planet.

Technology

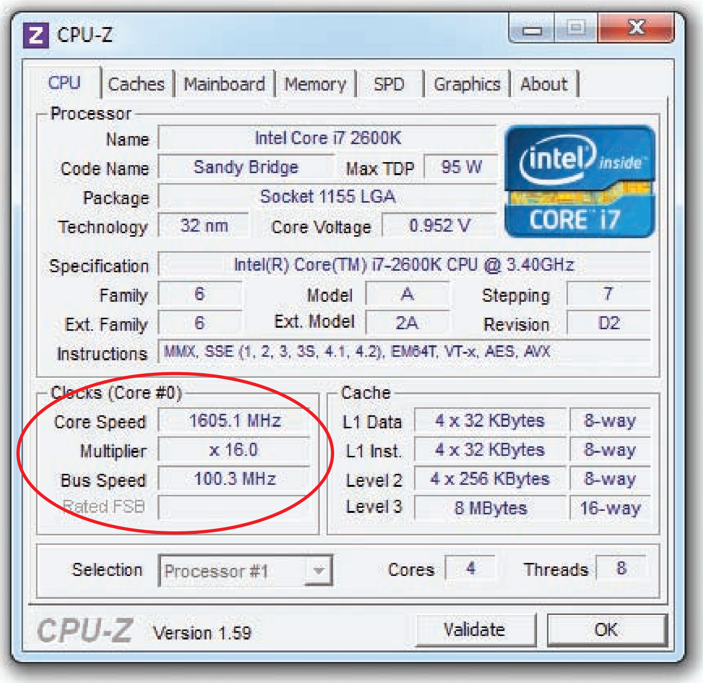
Although microprocessors today still serve the same function as the venerable 8088—crunching numbers—they do so far more efficiently. Engineers have altered, enhanced, and improved CPUs in a number of ways. This section looks at seven features:

- Clock multipliers
- 64-bit processing
- Virtualization support
- Parallel execution
- Multicore processing
- Integrated memory controller (IMC)
- Integrated graphics processing unit (GPU)

Clock Multipliers

All modern CPUs run at some multiple of the system clock speed. The system bus on my Core i7 machine, for example, runs at 100 MHz. The clock multiplier goes up to ×35 at full load to support the 3.4 GHz maximum speed. Originally, CPUs ran at the speed of the bus, but engineers early on realized the CPU was the only thing doing any work much of the time. If the engineers could speed up just the internal operations of the CPU and not anything else, they could speed up the whole computing process. Figure 6.22 shows a nifty program called CPU-Z displaying my CPU details. Note that all I'm doing is typing at the moment so SpeedStep has dropped the clock multiplier down to ×16 and the CPU core speed is only 1600 MHz.

The clock speed and the multiplier on early clock-multiplying systems had to be



Processor
Name: Intel Core i7 2600K
Code Name: Sandy Bridge Max TDP: 95 W
Package: Socket 1155 LGA
Technology: 32 nm Core Voltage: 0.952 V
Specification: Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz
Family: 6 Model: A Stepping: 7
Ext. Family: 6 Ext. Model: 2A Revision: D2
Instructions: MMX, SSE (1, 2, 3, 3S, 4.1, 4.2), EM64T, VT-x, AES, AVX

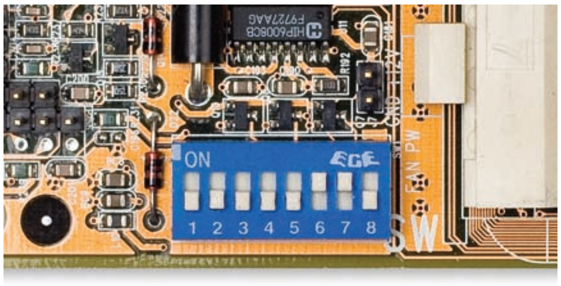
Clocks (Core #0)
Core Speed: 1605.1 MHz
Multiplier: x 16.0
Bus Speed: 100.3 MHz
Rated FSB:

Cache
L1 Data: 4 x 32 KBytes, 8-way
L1 Inst.: 4 x 32 KBytes, 8-way
Level 2: 4 x 256 KBytes, 8-way
Level 3: 8 MBytes, 16-way

Selection: Processor #1 Cores: 4 Threads: 8

CPU-Z Version 1.59

• **Figure 6.22** CPU-Z showing the clock speed, multiplier, and bus speed of a Core i7 processor hardly breaking a sweat manually configured via jumpers or dual in-line package (DIP) switches on the motherboard (see Figure 6.23). Today's CPUs report to the motherboard through a function called CPUID (CPU identifier), and the speed and multiplier are set automatically. (You can manually override this automatic setup on many motherboards. See "Overclocking," later in this chapter, for details.)



• **Figure 6.23** DIP switches on a motherboard

64-Bit Processing

Over successive generations of microprocessors, engineers have upgraded many physical features of CPUs. The EDB gradually increased in size, from 8- to 16- to 32- to 64-bits wide. The address bus similarly jumped, going from 20- to 24- to 32-bits wide (where it stayed for a decade).

The technological features changed as well. Engineers added new and improved registers, for example, that used fancy names like multimedia extensions (MMX) and Streaming SIMD Extensions (SSE). A mighty shift started a couple of years ago and continues to evolve: the move to 64-bit computing.

Most new CPUs support **64-bit processing**, meaning they can run a compatible 64-bit operating system, such as Windows 7, and 64-bit applications. They also support 32-bit processing for 32-bit operating systems, such as Windows XP, and 32-bit applications. The general-purpose registers also make the move up to 64-bit. The primary benefit to moving to 64-bit computing is that modern systems can support much more than the 4 GB of memory supported with 32-bit processing.

With a 64-bit address bus, CPUs can address 2^{64} bytes of memory, or more precisely, 18,446,744,073,709,551,616 bytes of memory—that's a lot of RAM! This number is so big that gigabytes and terabytes are no longer convenient, so we now go to an exabyte (2^{60}), abbreviated *EB*. A 64-bit address bus can address 16 EB of RAM.

In practical terms, 64-bit computing greatly enhances the performance of programs that work with large files, such as video editing applications. You'll see a profound improvement moving from 4 GB to 8 GB or 12 GB of RAM with such programs.

Virtualization Support

Intel and AMD have built in support for running more than one operating system at a time, a process called *virtualization*. Virtualization is very cool and gets its own chapter later in the book (Chapter 30), so I'll skip the details here. The key issue from a CPU standpoint is that virtualization used to work entirely through software. Programmers had to write a ton of code to enable a CPU—that was designed to run one OS at a time—to run more than one OS at the same time. Think about the issues involved. How does the memory get allocated, for example, or how does the CPU know which OS to update when you type something or click an icon? With hardware-based virtualization support, CPUs took a lot of the burden off the programmers and made virtualization a whole lot easier.

Parallel Execution

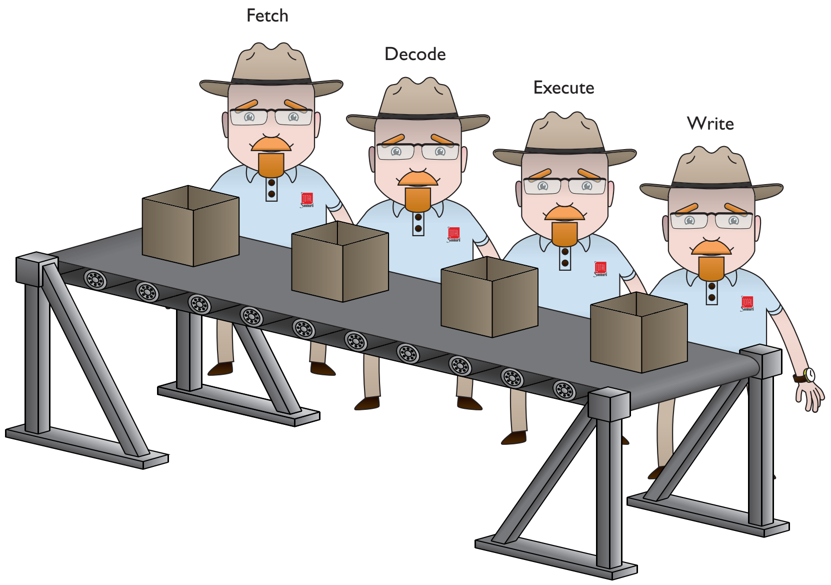
Modern CPUs can process multiple commands and parts of commands in parallel, known as **parallel execution**. Early processors had to do everything in a strict, linear fashion. The CPUs accomplish this parallelism through multiple pipelines, dedicated cache, and the capability to work with multiple threads or programs at one time. To understand the mighty leap in efficiency gained from parallel execution, you need insight into the processing stages.

Pipelining To get a command from the data bus, do the calculation, and then get the answer back out on the data bus, a CPU takes at least four steps (each of these steps is called a *stage*):

1. **Fetch** Get the data from the EDB.
2. **Decode** Figure out what type of command needs to be executed.
3. **Execute** Perform the calculation.
4. **Write** Send the data back onto the EDB.

Smart, discrete circuits inside the CPU handle each of these stages. In early CPUs, when a command was placed on the data bus, each stage did its job and the CPU handed back the answer before starting the next command, requiring at least four clock cycles to process a command. In every clock cycle, three of the four circuits sat idle. Today, the circuits are organized in a conveyer-belt fashion called a **pipeline**. With pipelining, each stage does its job with each clock-cycle pulse, creating a much more efficient process. The CPU has multiple circuits doing multiple jobs, so let's add pipelining to the Man in the Box analogy. Now, it's *Men* in the Box (see Figure 6.24)!

Pipelines keep every stage of the processor busy on every click of the clock, making a CPU run more efficiently without increasing the clock speed. Note that at this point, the CPU has four stages: fetch, decode, execute,



• **Figure 6.24** Simple pipeline

Tech Tip

x86 Versus x64

The terminology of CPUs can trip up new techs, so here's the scoop. CPUs from the early days can be lumped together as x86 CPUs, because they used an instruction set that built upon the earliest Intel CPU architecture. The Intel Core 2 Duo, for example, could run a program written for an ancient 80386 processor that was in fashion in the early 1990s.

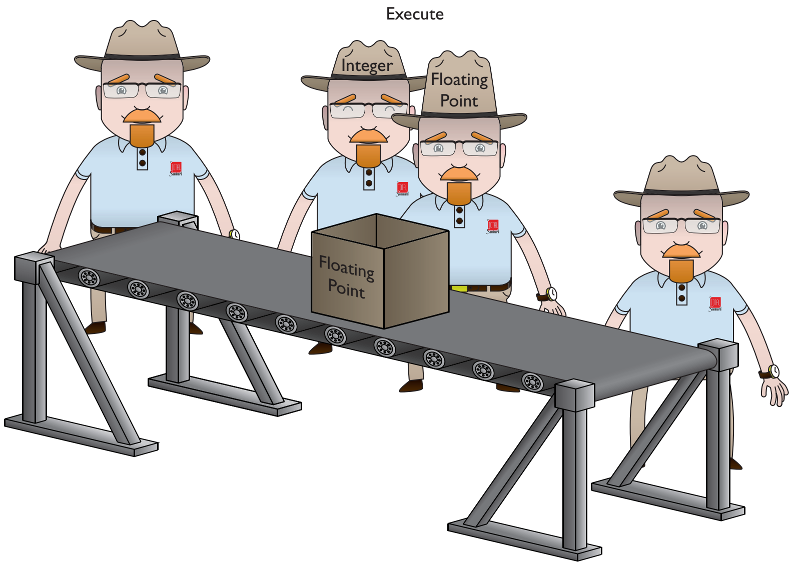
When the 64-bit CPUs went mainstream, marketing folks needed some way to mark applications, operating systems, and so on such that consumers could quickly tell the difference between something compatible with their system or something not compatible. Since you generally cannot return software after you open it, this is a big deal. The marketing folks went with x64, and that created a mess.

The earlier 32-bit stuff had been marketed as x86, not x32, so now we have x86 (old, 32-bit stuff) vs. x64 (new, 64-bit stuff). It's not pretty, but do you get the difference? To make matters even worse, however, x64 processors quite happily handle x86 code and are, by definition, x86 processors too!

and write—a four-stage pipeline. No CPU ever made has fewer than four stages, but advancements in caching (see "Cache," next) have increased the number of stages over the years. Current CPU pipelines contain many more stages, up to 20 in some cases.

Pipelining isn't perfect. Sometimes a stage hits a complex command that requires more than one clock cycle, forcing the pipeline to stop. Your CPU tries to avoid these stops, or *pipeline stalls*. The decode stage tends to cause the most pipeline stalls; certain commands are complex and therefore harder to decode than other commands. Current processors use multiple decode stages to reduce the chance of pipeline stalls due to complex decoding.

The inside of the CPU is composed of multiple chunks of circuitry to handle the many types of calculations your PC needs to do. For example, one part, the **arithmetic logic unit (ALU)** (or *integer unit*), handles integer math: basic math for numbers with no decimal point. A perfect example of integer math is 2 + 3 = 5. The typical CPU spends most of its work doing integer math. CPUs also have special circuitry to handle complex numbers, called the **floating point unit (FPU)**. With a single pipeline, only the ALU or the FPU worked at any execution stage. Worse yet, floating point calculation often took many, many clock cycles to execute, forcing the CPU to stall the pipeline until the FPU finished executing the complex command (see Figure 6.25). Current CPUs offer multiple pipelines to keep the processing going (see Figure 6.26).

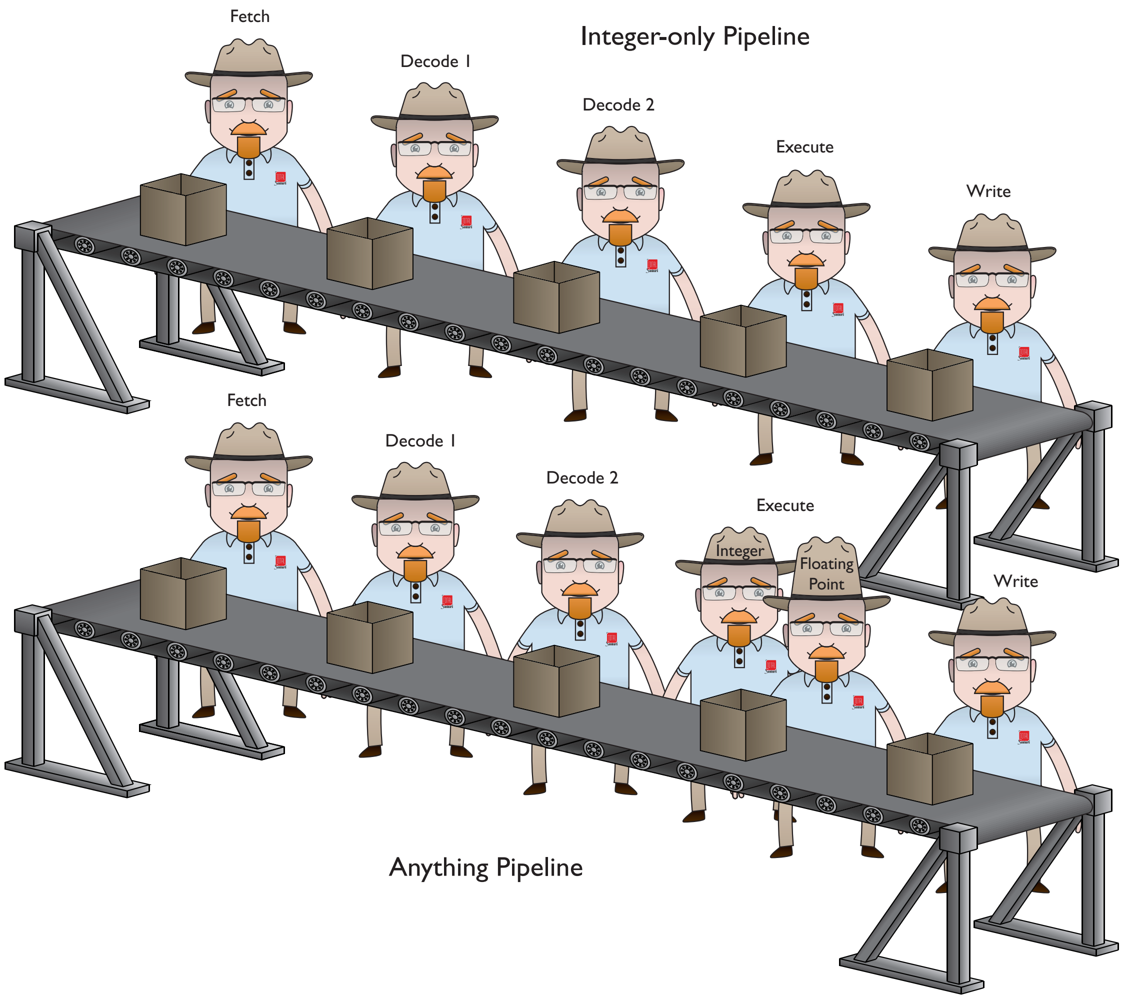


• **Figure 6.25** Bored integer unit

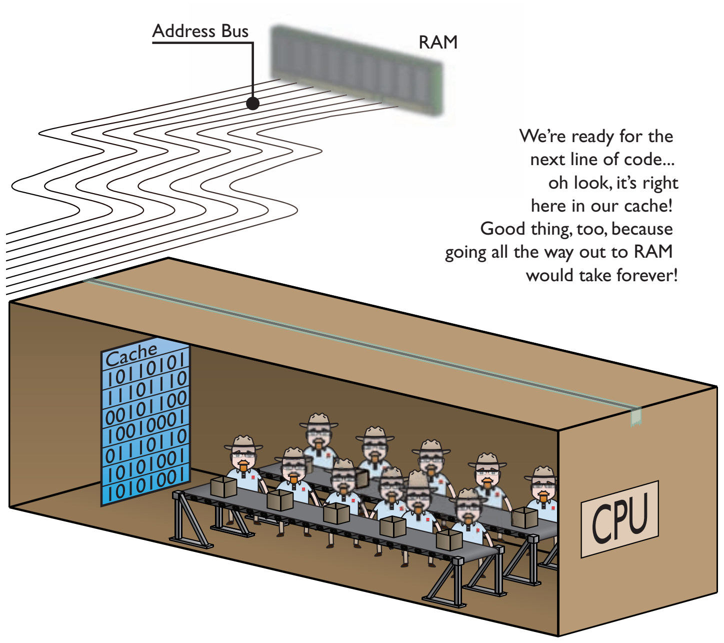
Cache When you send a program to the CPU, you actually run lots of little programs all at the same time. Okay, let's be fair here: *you* didn't run all of these little programs—you just started your Web browser or some other program. The moment you double-clicked that icon, Windows started sending many programs to the CPU. Each of these programs breaks down into some number of little pieces, called *threads*, and data. Each thread is a series of instructions designed to do a particular job with the data.

Modern CPUs don't execute instructions sequentially—first doing step 1, then step 2, and so on—but rather process all kinds of instructions. Most applications have certain instructions and data that get reused, sometimes many times.

Pipelining CPUs work fantastically well as long as the pipelines stay filled with instructions. Because the CPU runs faster than the RAM can supply it with code, you'll always get pipeline stalls—called **wait states**—because the RAM can't keep up with the CPU. To reduce wait states, CPUs come with built-in, very high-speed RAM called **static RAM (SRAM)**. This SRAM preloads as many instructions as possible and keeps copies of already run instructions and data in case the CPU needs to work on them again (see Figure 6.27). SRAM used in this fashion is called a **cache**.



• Figure 6.26 Multiple pipelines


Address Bus
RAM
We're ready for the next line of code... oh look, it's right here in our cache! Good thing, too, because going all the way out to RAM would take forever!
Cache
10110101
11101110
00101100
10010001
01110110
10101001
10101001
CPU

• Figure 6.27 RAM cache

The SRAM cache inside the early CPUs was tiny, only about 16 KB, but it improved performance tremendously. In fact, it helped so much that many motherboard makers began adding a cache directly to the motherboards. These caches were much larger, usually around 128 to 512 KB. When the CPU looked for a line of code, it first went to the built-in cache; if the code wasn't there, the CPU went to the cache on the motherboard. The cache on the CPU was called the *L1 cache* because it was the one the CPU first tried to use. The cache on the motherboard was called the *L2 cache*, not because it was on the motherboard, but because it was the second cache the CPU checked.

Typically, the CompTIA A+ exams expect you to know that L1 cache will be the smallest and fastest cache; L2 will be bigger and slower than L1; and L3 will be the biggest and slowest cache.

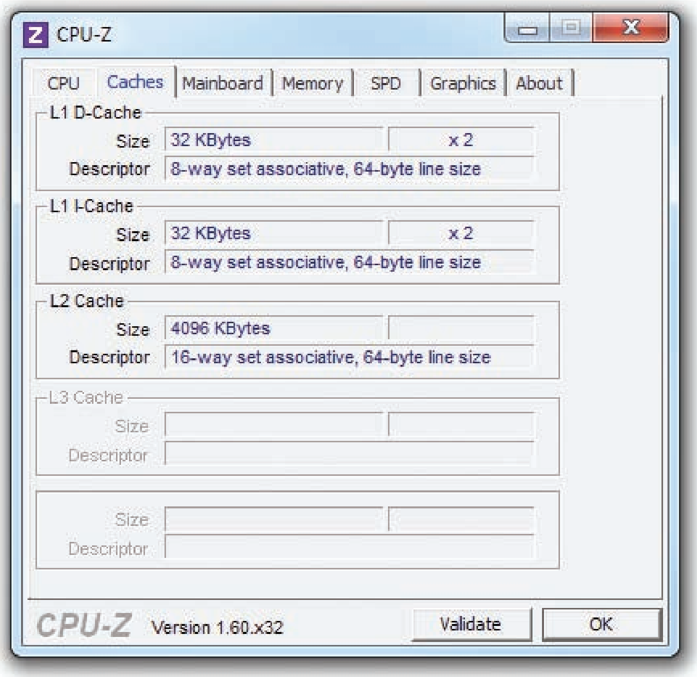
Eventually, engineers took this cache concept even further and added the L2 cache onto the CPU package (see Figure 6.28). Some CPUs even include three caches: an L1, an L2, and an L3 cache.

The L2 cache on the early CPUs that had L2 cache included on the CPU package ran at a slower clock speed than the L1 cache. The L1 cache was in the CPU and thus ran at the speed of the CPU. The L2 cache connected to the CPU via a tiny set of wires on the CPU package. The first L2 caches ran at half the speed of the CPU.

The inclusion of the L2 cache on the chip gave rise to some new terms to describe the connections between the CPU, MCC, RAM, and L2 cache. The address bus and external data bus (connecting the CPU, MCC, and RAM) were lumped into a single term called the **frontside bus**, and the connection between the CPU and the L2 cache became known as the **backside bus** (see Figure 6.29).

To keep up with faster processors, motherboard manufacturers began to double and even quadruple the size of the frontside bus. Techs sometimes refer to these as *double-pumped* and *quad-pumped* frontside buses.

The end of the road for the terms frontside bus and backside bus seems to be at the Intel Core 2 Duo and Core 2 Quad processors, as far as the CompTIA A+ certification exams are concerned. The terms are also used

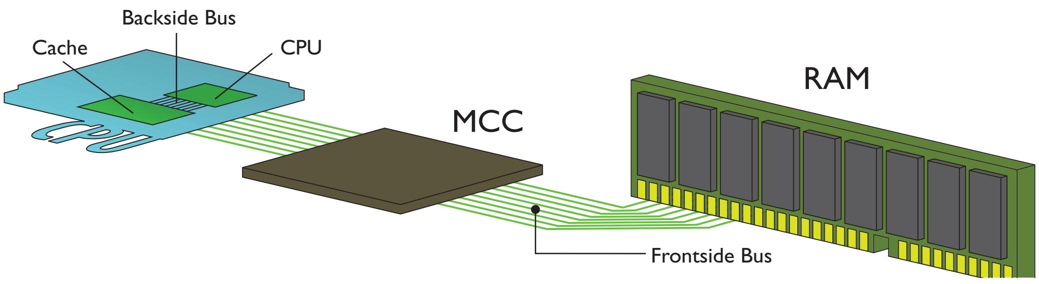


```
L1 D-Cache
        Size  32 KBytes                                    x 2
  Descriptor  8-way set associative, 64-byte line size
L1 I-Cache
        Size  32 KBytes                                    x 2
  Descriptor  8-way set associative, 64-byte line size
L2 Cache
        Size  4096 KBytes
  Descriptor  16-way set associative, 64-byte line size
L3 Cache
        Size
  Descriptor

        Size
  Descriptor

Version 1.60.x32
```

• **Figure 6.28** CPU-Z displaying the cache information for a Core i7 processor

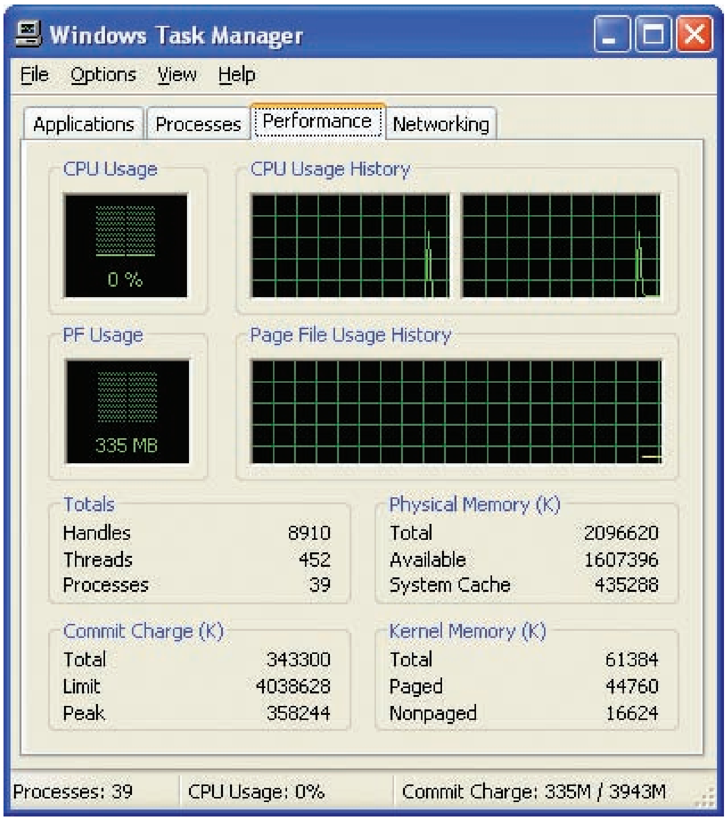


• **Figure 6.29** Frontside and backside buses

by a lot of techs to refer to simply "the connection between the CPU and system RAM."

Multithreading At the peak of the single-CPU 32-bit computing days, Intel released a CPU called the Pentium 4 that took parallelism to the next step with Hyper-Threading. *Hyper-Threading* enabled the Pentium 4 to run multiple threads at the same time, what's generically called *simultaneous multithreading*, effectively turning the CPU into two CPUs on one chip—with a catch.

Figure 6.30 shows the Task Manager in Windows XP on a system running a Hyper-Threaded Pentium 4. Note how the CPU box is broken into two groups—Windows thinks this one CPU is two CPUs.



Totals	
Handles	8910
Threads	452
Processes	39

Physical Memory (K)	
Total	2096620
Available	1607396
System Cache	435288

Commit Charge (K)	
Total	343300
Limit	4038628
Peak	358244

Kernel Memory (K)	
Total	61384
Paged	44760
Nonpaged	16624

Processes: 39 CPU Usage: 0% Commit Charge: 335M / 3943M

• **Figure 6.30** Windows Task Manager with the Performance tab displayed for a system running a Hyper-Threaded Pentium 4

Tech Tip

The Return of Hyper-Threading

Though its use had faded after the Pentium 4, Intel brought back Hyper-Threading to mainstream processors with the Core i7 in 2011 (and a couple of years earlier with the low-powered Atom processors), and Microsoft optimized Windows 7 to support the technology. This time, it seems, Hyper-Threading is here to stay.

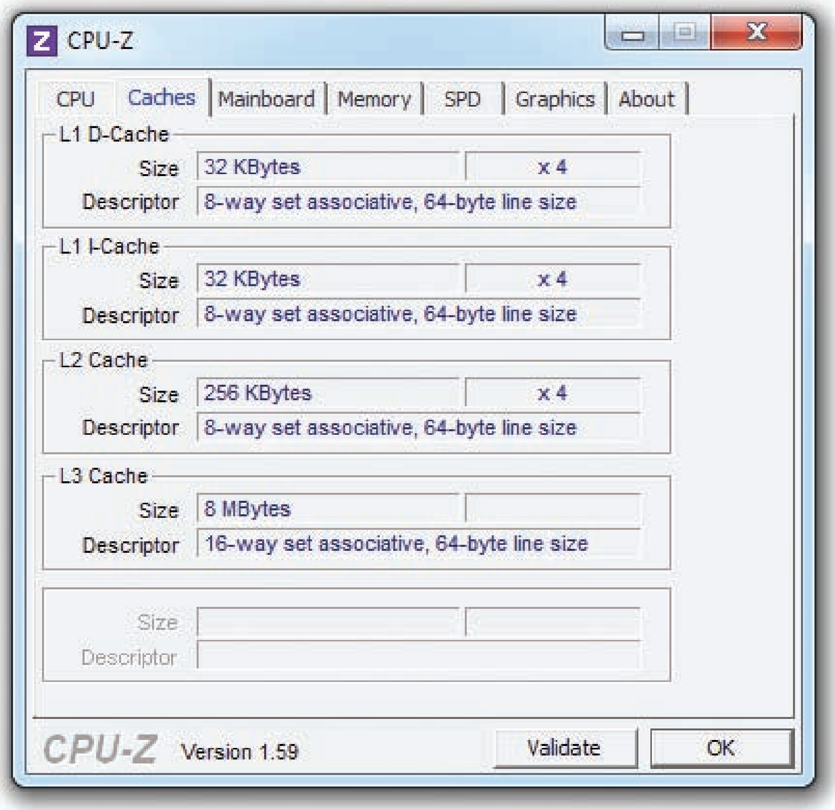
Multithreading enhances a CPU's efficiency but with a couple of limitations. First, the operating system and the application have to be designed to take advantage of the feature. Second, although the CPU simulates the actions of a second processor, it doesn't double the processing power because the main execution resources are not duplicated.

Multicore Processing

CPU clock speeds hit a practical limit of roughly 4 GHz around 2002–2003, motivating the CPU makers to find new ways to get more processing power for CPUs. Although Intel and AMD had different opinions about 64-bit CPUs, both decided at virtually the same time to combine two CPUs (or *cores*) into a single chip, creating a **dual-core** architecture. A dual-core CPU has two execution units—two sets of pipelines—but the two sets of pipelines share caches and RAM.

Today, dual-core CPUs like the Intel Core 2 Duo are common, and multicore CPUs—with four, six, or eight cores—grace the desktop PCs of many enthusiasts. With each generation of multicore CPU, both Intel and AMD have tinkered with the mixture of how to allocate the cache among the cores. Figure 6.31 shows another screenshot of CPU-Z, this time displaying the cache breakdown of a Core i7.

Figure 6.31 reveals specific details about how this Intel CPU works with cache. The Core i7 has L1, L2, and L3 caches of 64 KB, 256 KB, and 8 MB, respectively. (The L1 cache divides into 32 KB to handle data—the



L1 D-Cache
Size: 32 KBytes x 4
Descriptor: 8-way set associative, 64-byte line size

L1 I-Cache
Size: 32 KBytes x 4
Descriptor: 8-way set associative, 64-byte line size

L2 Cache
Size: 256 KBytes x 4
Descriptor: 8-way set associative, 64-byte line size

L3 Cache
Size: 8 MBytes
Descriptor: 16-way set associative, 64-byte line size

Size
Descriptor

CPU-Z Version 1.59

• **Figure 6.31** CPU-Z showing the cache details of a Sandy Bridge Core i7

D-Cache—and another 32 KB for instructions—the *I-Cache*.) Each core has dedicated L1 and L2 caches. (You can tell this by the ×4 to the right of the capacity listing.) All four cores share the giant L3 cache. That pool of memory enables the cores to communicate and work together without having to access the radically slower main system RAM as much. CPU manufacturers engineered the cores in multicore CPUs to divide up work independently of the OS, known as **multicore processing**. This differs from Hyper-Threading, where the OS and applications have to be written specifically to handle the multiple threads. Note that even with multicore processors, applications have to be modified or optimized for this parallelism to have a huge impact on performance.

Integrated Memory Controller

Almost all current microprocessors have an **integrated memory controller (IMC)**, moved from the motherboard chip into the CPU to optimize the flow of information into and out from the CPU. An IMC enables faster control over things like the large L3 cache shared among multiple cores.

Just like in so many other areas of computing, manufacturers implement a variety of IMCs in their CPUs. In practice, this means that different CPUs handle different types and capacities of RAM. I'll save the details on those RAM variations for Chapter 7. For now, add "different RAM support" to your list of things to look at when making a CPU recommendation for a client.

Tech Tip

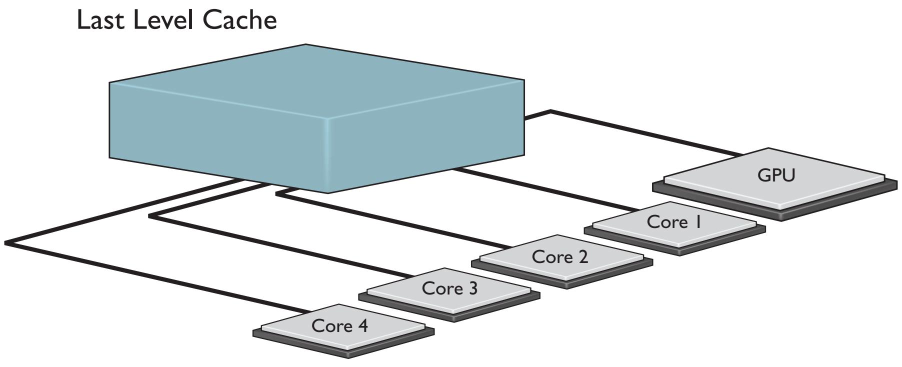
NVIDIA Tegra and "Project Denver"

One of the two major players in the performance GPU market, NVIDIA, also manufactures microprocessors that have combined CPUs and GPUs. The current offering, called the Tegra, is almost a complete computer on a single chip. You'll find it in many mobile devices (see Chapter 27 for the scoop on these delicious gadgets). NVIDIA hasn't made a push for the PC desktop market yet, although that might change with the release of their integrated microprocessor code named "Project Denver" in 2013.

Integrated Graphics Processing Unit

As you'll read about in much more detail in Chapter 21, the video processing portion of the computer—made up of the parts that put a changing image on the monitor—traditionally has a discrete microprocessor that differs in both function and architecture from the CPUs designed for general-purpose computing. The generic term for the video processor is a **graphics processing unit (GPU)**. I'll spare you the details until we get to video in Chapter 21, but it turns out that graphics processors can handle certain tasks much more efficiently than the standard CPU. Integrating a GPU into the CPU enhances the overall performance of the computer while at the same time reducing energy use, size, and cost. With the proliferation of mobile devices and portable computers today, all of these benefits have obvious merit.

Both major CPU manufacturers have released CPUs with integrated GPUs. The architecture differs in some ways, such as how they use the cache on the chip. The *Intel HD Graphics* integrated into many Core i3/i5/i7 processors, for example, has the CPU cores and the GPU core sharing the "last level cache," which is either L2 or L3, depending on the processor (see Figure 6.32). With the AMD *accelerated processing unit* (*APU*), such as the AMD Fusion, the GPU has access to all levels of cache on the CPU.



• **Figure 6.32** Dedicated cache (in each core) and shared cache

Selecting, Installing, and Troubleshooting CPUs

Now that you know how CPUs work, it's time to get practical. This last section discusses selecting the proper CPU, installing several types of processors, and troubleshooting the few problems techs face with CPUs.

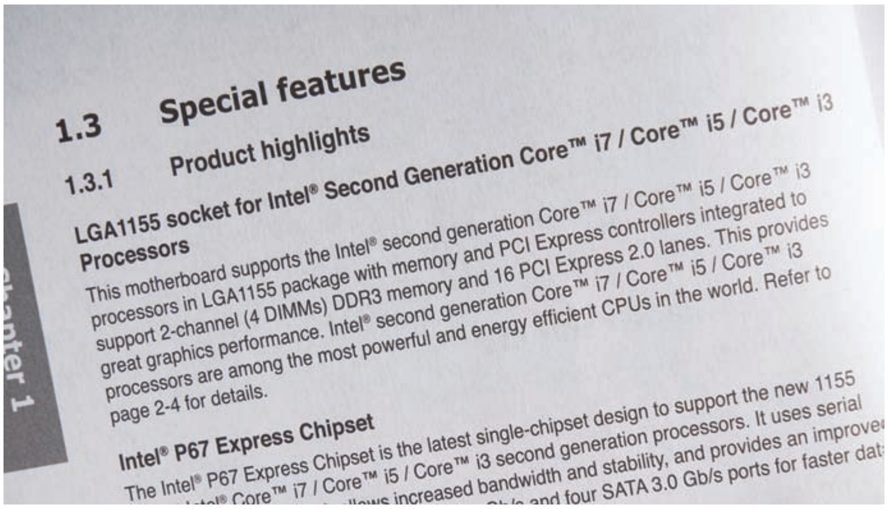
Selecting a CPU

When selecting a CPU, you need to make certain you get one that the motherboard can accommodate. Or, if you're buying a motherboard along with the CPU, then get the right CPU for the intended purpose. Chapter 31 discusses computer roles and helps you select the proper components for each role. You need to have a lot more knowledge of all the pieces around the CPU to get the full picture, so we'll wait until then to discuss the "why" of a particular processor. Instead, this section assumes you're placing a new CPU in an already-acquired motherboard. You need to address two key points in selecting a CPU that will work. First, does the motherboard support Intel or AMD CPUs? Second, what socket does the motherboard have?

To find answers to both those questions, you have two sources: the motherboard book or manual and the manufacturer's Web site. Figure 6.33 shows a manual for an Asus motherboard open to reveal the supported processors and the socket type.

Just as Intel and AMD make many types of CPUs, motherboards are manufactured with various different types of sockets. The CompTIA A+ exams expect you to know which sockets go with which family of CPU. Table 6.2 charts the important Intel ones; Table 6.3 lists the AMD-based sockets. I would show you all the pictures, but, frankly, CPU sockets aren't the sexiest part of the computer.

I've included the number of socket pins for the AMD-based sockets because related questions have been on the CompTIA A+ certifications in exam objectives prior to 801. You don't need to memorize the Intel numbers, because Intel names the sockets by the number of pins.



1.3 Special features

1.3.1 Product highlights

LGA1155 socket for Intel® Second Generation Core™ i7 / Core™ i5 / Core™ i3 Processors

This motherboard supports the Intel® second generation Core™ i7 / Core™ i5 / Core™ i3 processors in LGA1155 package with memory and PCI Express controllers integrated to support 2-channel (4 DIMMs) DDR3 memory and 16 PCI Express 2.0 lanes. This provides great graphics performance. Intel® second generation Core™ i7 / Core™ i5 / Core™ i3 processors are among the most powerful and energy efficient CPUs in the world. Refer to page 2-4 for details.

Intel® P67 Express Chipset

The Intel® P67 Express Chipset is the latest single-chipset design to support the new 1155 Intel® Core™ i7 / Core™ i5 / Core™ i3 second generation processors. It uses serial [illegible] increased bandwidth and stability, and provides an improve [illegible] and four SATA 3.0 Gb/s ports for faster dat

• Figure 6.33 Supported processors and socket type

Table 6.2 Intel-based Sockets

Socket	CPU
LGA 775[1]	Pentium 4, Celeron, Pentium 4 Extreme Edition, Core 2 Duo, Core 2 Quad, Xeon, and many others
LGA 1155[2]	Core i3/i5/i7, Pentium, Celeron, Xeon
LGA 1156[3]	Core i3/i5/i7, Pentium, Celeron, Xeon
LGA 1366[4]	Core i7, Xeon, Celeron

1 The LGA 775 socket was the only desktop or server socket used for many years by Intel and thus just about every branded Intel CPU used it at one time or another.

2 Socket LGA 1155 CPUs are based on Sandy Bridge or Ivy Bridge architecture.

3 Socket LGA 1156 CPUs are based on the pre–Sandy Bridge architecture.

4 The very first Core i7 processors used LGA 1366.

This list of Intel CPUs covers only those listed on the CompTIA A+ 801 exam objectives. The CompTIA A+ 701/702 exams mentioned a couple of other processors, such as the *Pentium Pro* (very old, 32-bit) and the *Itanium* (also old; used for servers exclusively). You might see them as incorrect answers on the 801 exam.

Table 6.3 AMD-based Sockets

Socket	Pins	CPU
940[1]	940	Opteron, Athlon 64 FX
AM2[2]	940	Athlon 64, Athlon 64 X2, Athlon 64 FX, Opteron, Sempron, Phenom
AM2+[2]	940	Athlon 64, Athlon 64 X2, Athlon II, Opteron, Phenom, Phenom II
AM3[3]	941	Phenom II, Athlon II, Sempron, Opteron
AM3+[4]	942	FX
FM1	905	A[5]
F	1207	Opteron, Athlon FX

1 You would usually find only an Opteron server-based CPU in a socket 940. The Athlon 64 FX was a super high-end CPU for the day and is very unlikely to be on the exams.

2 AMD says that CPUs designed for AM2 sockets will work in a Socket AM2+ and vice versa, though at the slower memory movement speeds of the AM2 component (socket or CPU). Many manufacturers have not released motherboard updates that would enable support for a Socket AM2+ CPU on a Socket AM2 motherboard, thus making it impossible to upgrade just the CPU.

3 Though the names of some of the processors designed for Socket AM3 match the names of CPUs designed for earlier sockets, they're *not* the same CPUs. They are specific to AM3 because they support different types of RAM (see Chapter 7). Just to make things even crazier, though, AM3 CPUs work just fine in Socket AM2/2+ motherboards.

4 AMD had only released the FX-branded CPUs that use the Bulldozer core at the time of this writing. Motherboards with AM3+ sockets already available tout support for Phenom II, Athlon II, and Sempron CPUs, in addition to the FX series.

5 The A series features integrated GPUs and other chips.

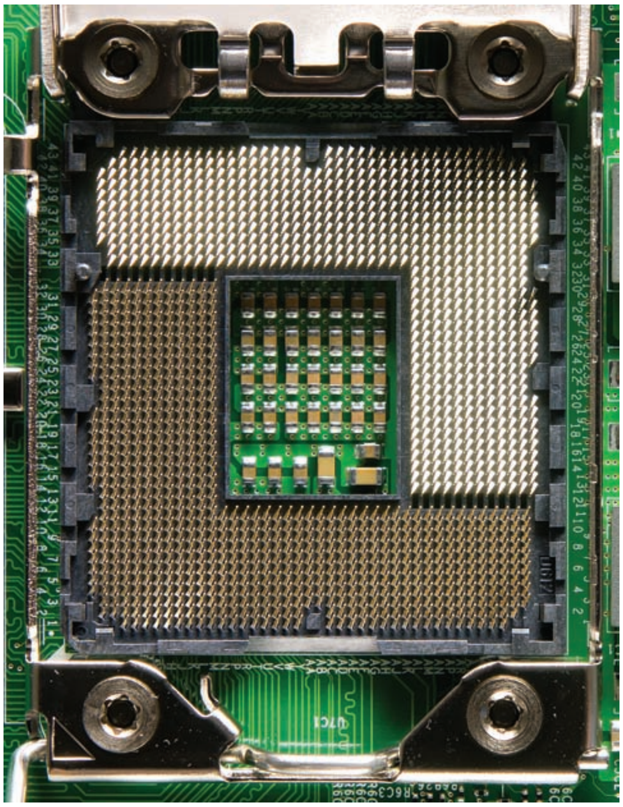
This list of AMD CPUs covers only those listed on the CompTIA A+ 801 exam objectives. As with the Intel processors, the CompTIA A+ 700-series exams mentioned a couple of other processors, such as the *Athlon XP* (old, 32-bit) and the *Duron* (also old; label used for low-end CPUs). You might see them as incorrect answers on the 801 exam.

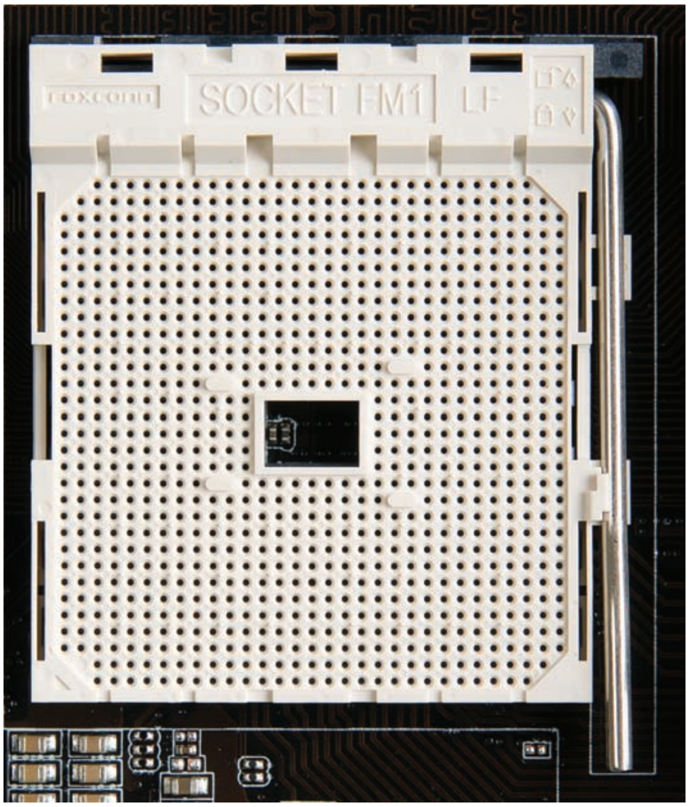
Installation Issues

When installing a CPU, you need to use caution with the tiny pins. Plus, you must make certain that the power supply can supply enough electricity for the processor to function along with all the other components on the computer. You have to provide adequate cooling. Finally, you can decide whether to leave the CPU at stock settings or overclock it.

Socket Types

When installing a CPU, you need to exercise caution not to bend any of the tiny pins. The location of the pins differs between Intel and AMD. With


• **Figure 6.34** Intel-based socket with pins

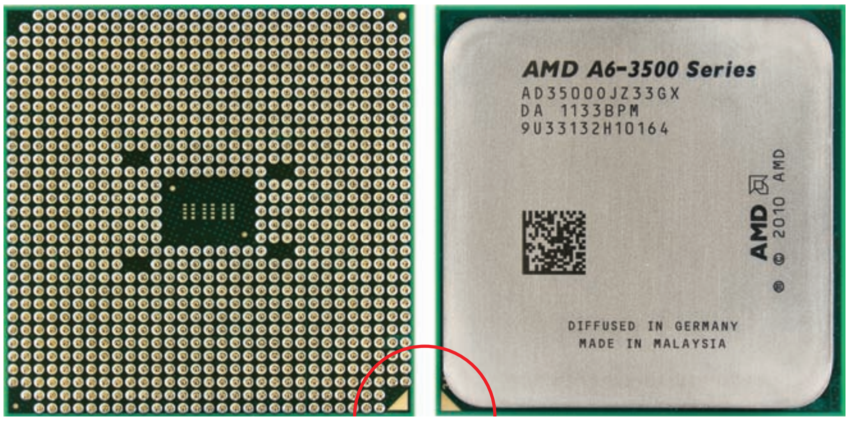


• **Figure 6.35** AMD-based socket without pins

Intel-based motherboards, the sockets have hundreds of tiny pins that line up with contacts on the bottom of the CPU (see Figure 6.34). Intel CPUs use a *land grid array* (*LGA*) package, where the underside of the CPU has hundreds of contact points that line up with the socket pins.

AMD CPUs have the pins (see Figure 6.35); the sockets have holes. The pins on the AMD *pin grid array* (*PGA*) CPUs align with the holes in the sockets.

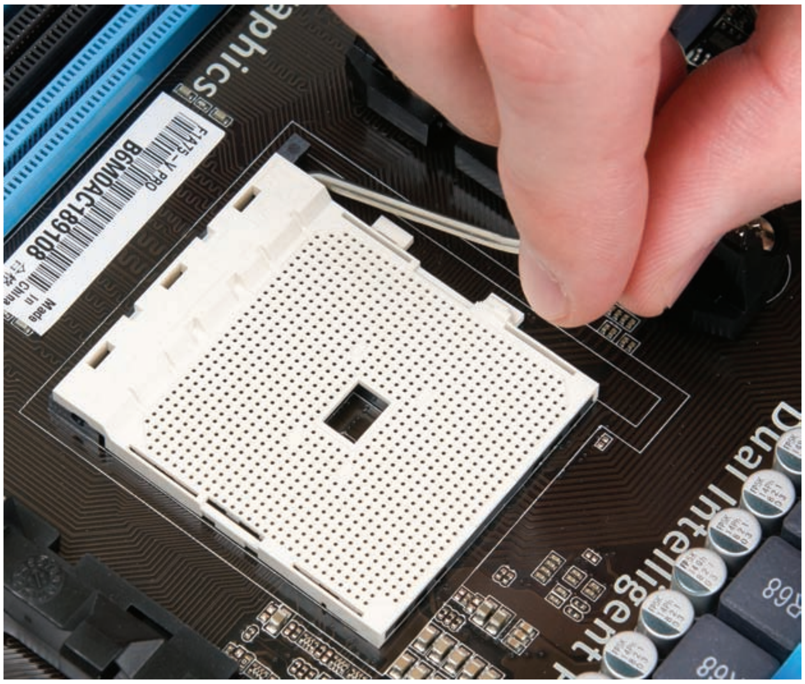
All CPUs and sockets are keyed so you can't (easily) insert them incorrectly. Look at the underside of the CPU in Figure 6.36 (left). Note that



• **Figure 6.36** Underside and top of a CPU the pins do not make a perfect square, because a few are missing. Now look at the top of the CPU (right). See the little mark at the corner? The socket also has tiny markings so you can line the CPU up properly with the socket.

In both socket styles, you release the retaining mechanism by pushing the little lever down slightly and then away from the socket (see Figure 6.37). You next raise the arm fully, and then move the retaining bracket (see Figure 6.38).

Align the processor with the socket and gently drop the processor into place. If it doesn't go in easily, check the orientation and try again. These sockets are generically called **zero insertion force (ZIF) sockets**, which means you never have to use any force at all.



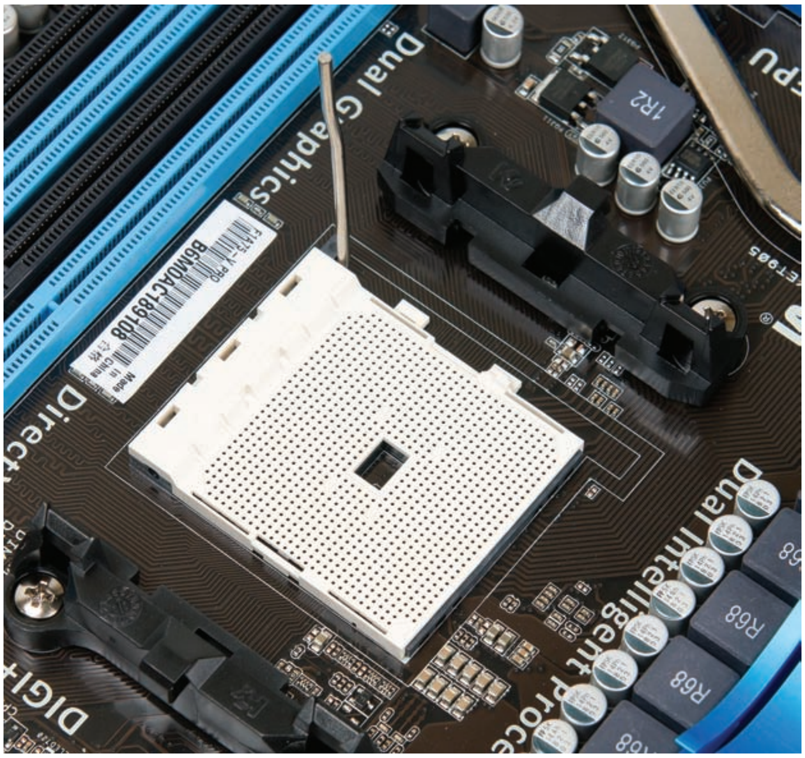
• Figure 6.37 Moving the release arm

Cooling

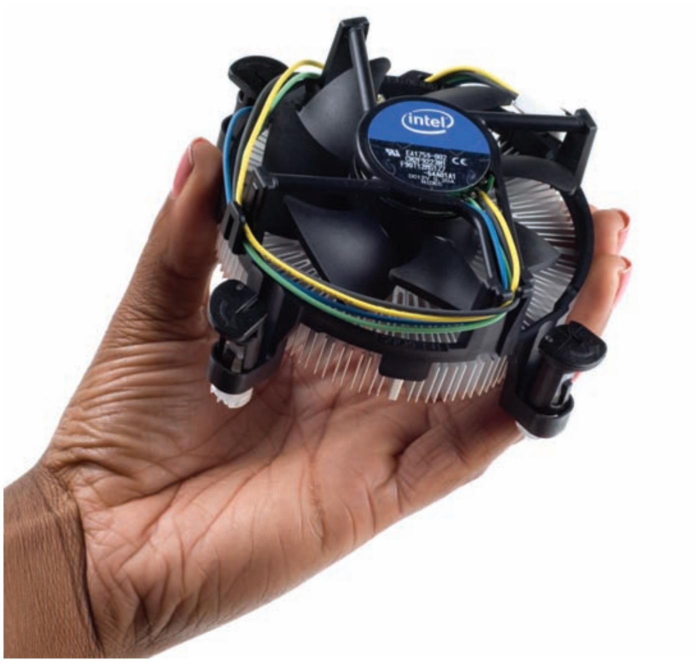
CPUs work very hard and thus require power to function. In electrical terms, CPUs consume *wattage* or *watts*, a unit of electrical power, just like a 100-watt light bulb consumes power whenever it's on. (See Chapter 10 for more details about electricity.) Have you ever touched a light bulb after it's been on for a while? Ouch! CPUs heat up, too.

To increase the capability of the CPUs to handle complex code, CPU manufacturers have added a lot of microscopic transistors over the years. The more transistors the CPU has, the more power they need and thus the hotter they get. CPUs don't tolerate heat well, and modern processors need active cooling solutions just to function at all. Almost every CPU uses a combination of a heat-sink and fan assembly to wick heat away from the CPU. Figure 6.39 shows the standard Intel heat sink and fan.

There was a time, long ago, when CPUs didn't need any type of cooling device. You just snapped in the CPU and it worked. Well, those days are gone. Long gone. If you're installing a



• Figure 6.38 Fully opened socket



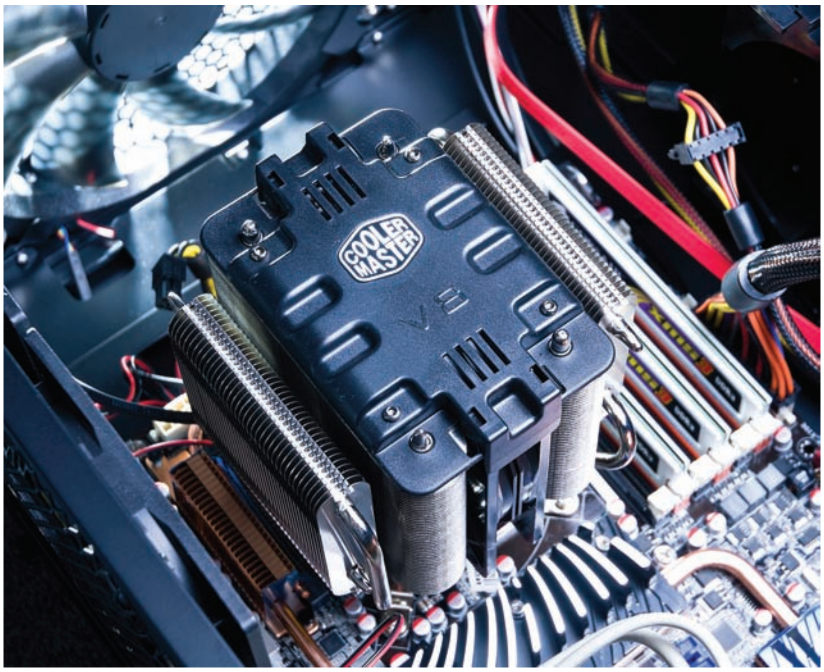
• **Figure 6.39** Intel OEM heat-sink and fan assembly

modern CPU, you will have to cool it. Fortunately, you have choices.

- **OEM CPU coolers** OEM heat-sink and fan assemblies are included with a retail-boxed CPU. OEM CPUs, on the other hand, don't normally come bundled with CPU coolers. Crazy, isn't it? OEM CPU coolers have one big advantage: you know absolutely they will work with your CPU.
- **Specialized CPU coolers** Many companies sell third-party heat-sink and fan assemblies for a variety of CPUs. These usually exceed the OEM heat sinks in the amount of heat they dissipate. These CPU coolers invariably come with eye-catching designs to look really cool inside your system—some are even lighted (see Figure 6.40).

The last choice is the most impressive of all: liquid cooling! **Liquid cooling** works by running some liquid—usually water—through a metal block that sits on top of your CPU, absorbing heat. The liquid gets heated by the block, runs out of the block and into something that cools the liquid, and is then pumped through the block again. Any liquid-cooling system consists of three main parts:

- A hollow metal block that sits on the CPU
- A pump to move the liquid around
- Some device to cool the liquid

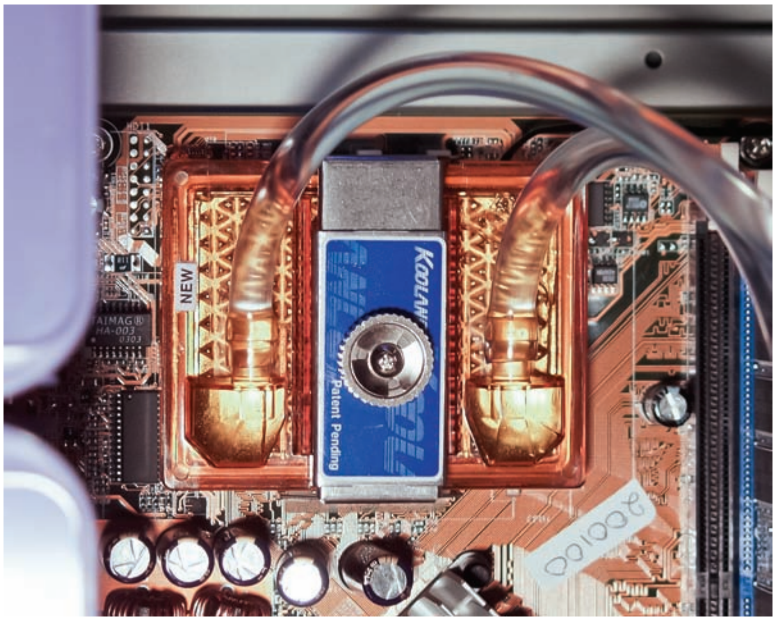


• **Figure 6.40** Cool retail heat sink

And of course, you need plenty of hosing to hook them all together. Figure 6.41 shows a typical liquid-cooled CPU.

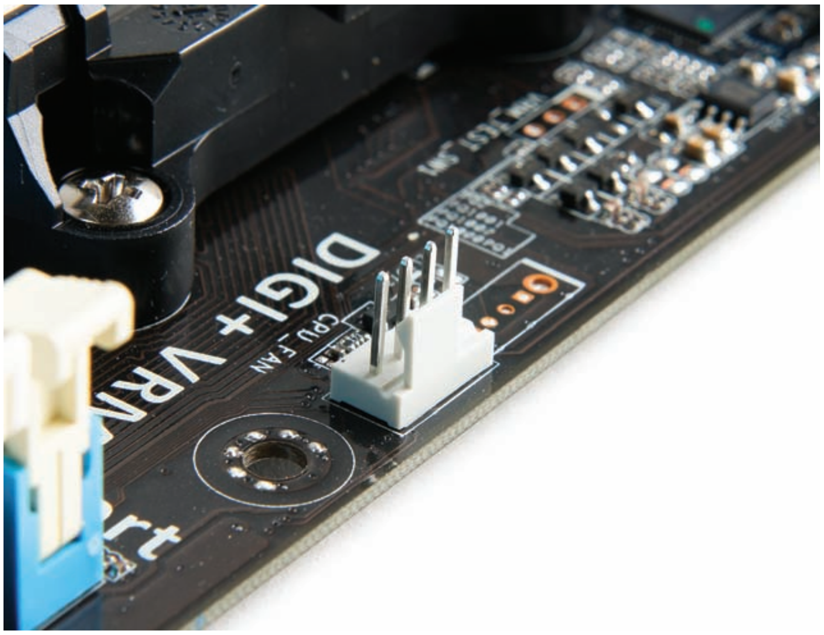
A number of companies sell these liquid-based cooling systems. Although they look impressive and certainly cool your CPU, unless you're overclocking or want a quiet system, a good fan will more than suffice.

Once you've got a heat-sink and fan assembly sorted out, you need to connect them to the motherboard. To determine the orientation of the heat-sink and fan assembly, check the power cable from the fan. Make sure it can easily reach the three- or four-wire standout on the motherboard (see Figure 6.42). If it can't, rotate the heat sink until it can. (Check the motherboard manual if you have trouble locating the CPU fan power standout.)

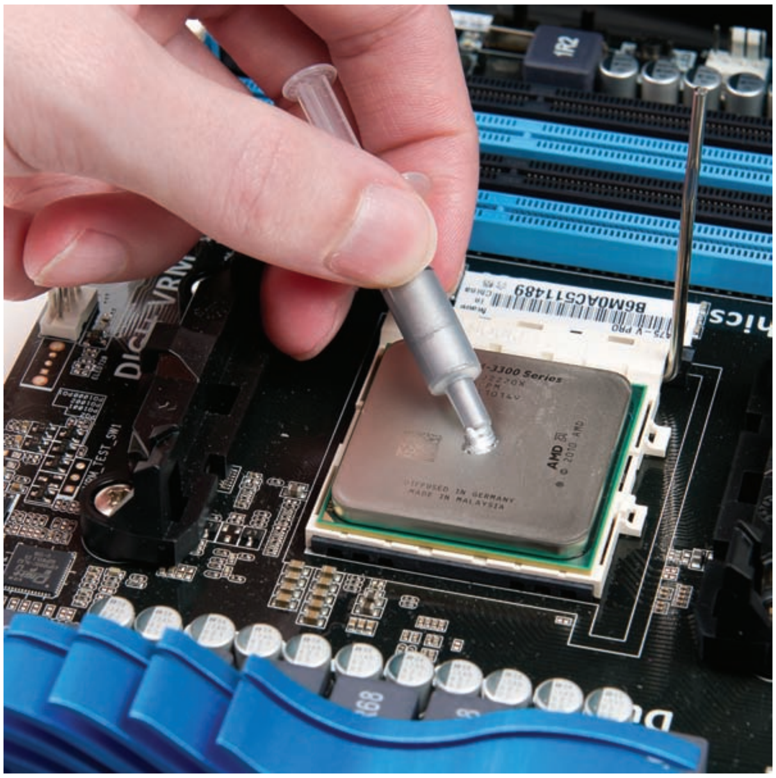


• **Figure 6.41** Liquid-cooled CPU

Next, before inserting the heat sink, you need to add a small amount of **thermal compound** (also called *heat dope* or *nasty silver goo*). Many heat sinks come with some thermal compound already on them; the thermal compound on these pre-doped heat sinks is covered by a small square of tape—take the tape off before you snap it to the CPU. If you need to put heat dope on from a tube, know that you need to use only a tiny amount of this compound (see Figure 6.43). Spread it on as thinly, completely, and evenly as you can. Unlike so many other things in life, you *can* have too much heat dope!



• **Figure 6.42** CPU fan power standout on motherboard


• Figure 6.43 Applying thermal compound

You secure heat sinks in various ways, depending on the manufacturer. Stock Intel heat sinks have four plungers that you simply push until they click into place in corresponding holes in the motherboard. AMD stock heat sinks generally have a bracket that you secure to two points on the outside of the CPU socket and a latch that you swivel to lock it down (see Figure 6.44).

Finally, you can secure many aftermarket heat-sink and fan assemblies by screwing them down from the underside of the motherboard (see Figure 6.45). You have to remove the motherboard from the case or install the heat sink before you put the motherboard in the case.

For the final step, plug the fan power connector into the motherboard standout. It won't work if you don't!

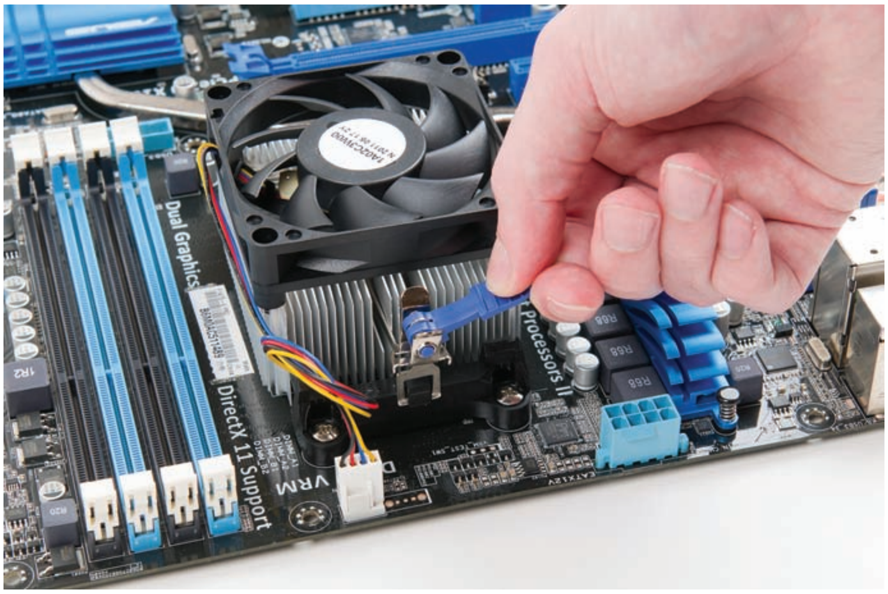
Overclocking

For the CPU to work, the motherboard speed, multiplier, and voltage must be set properly. In most modern systems, the motherboard uses the CPUID functions to set these options automatically. Some motherboards enable you to adjust these settings manually by moving a jumper, changing a CMOS setting, or using software; many enthusiasts deliberately change these settings to enhance performance.

Tech Tip

CMOS

Chapter 8 goes into gory detail about the system setup utility and the area in which it stores important data (called CMOS), but invariably students want to experiment at this point, so I'll give you some information now. You can access the system setup utility by pressing some key as the computer starts up. This is during the text phase, well before it ever says anything about starting Windows. Most systems require you to press the DELETE *key, but read the screen for details. Just be careful once you get into the system setup utility not to change anything you don't understand. And read Chapter 8!*

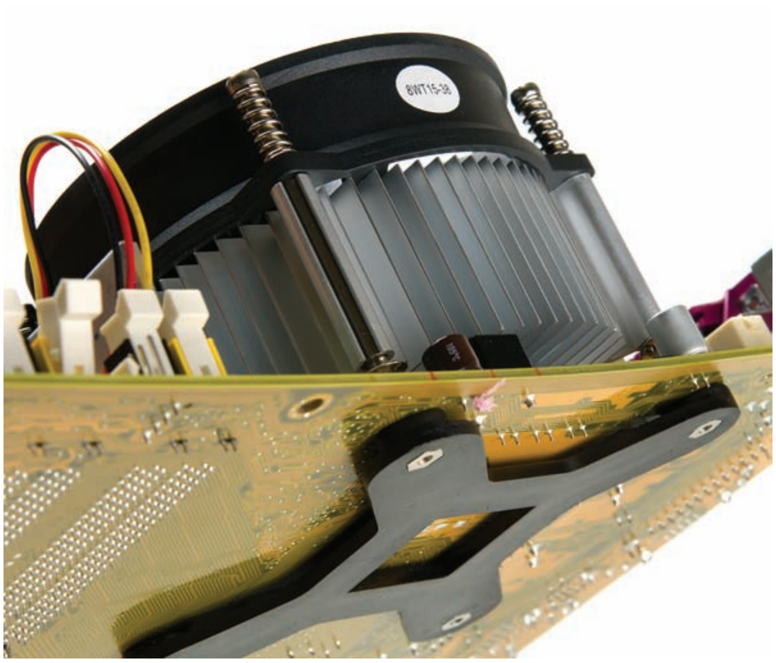

• Figure 6.44 AMD OEM heat-sink and fan assembly

Starting way back in the days of the Intel 80486 CPU, people intentionally ran their systems at clock speeds higher than the CPU was rated, a process called **overclocking**, and it worked. Well, *sometimes* the systems worked, and sometimes they didn't. Intel and AMD have a reason for marking a CPU at a particular clock speed—that's the highest speed they guarantee will work.

Before I say anything else, I must warn you that intentional overclocking of a CPU immediately voids most warranties. Overclocking has been known to destroy CPUs. Overclocking might make your system unstable and prone to lockups and reboots. I neither applaud nor decry the practice of overclocking. My goal here is simply to inform you of the practice. You make your own decisions.

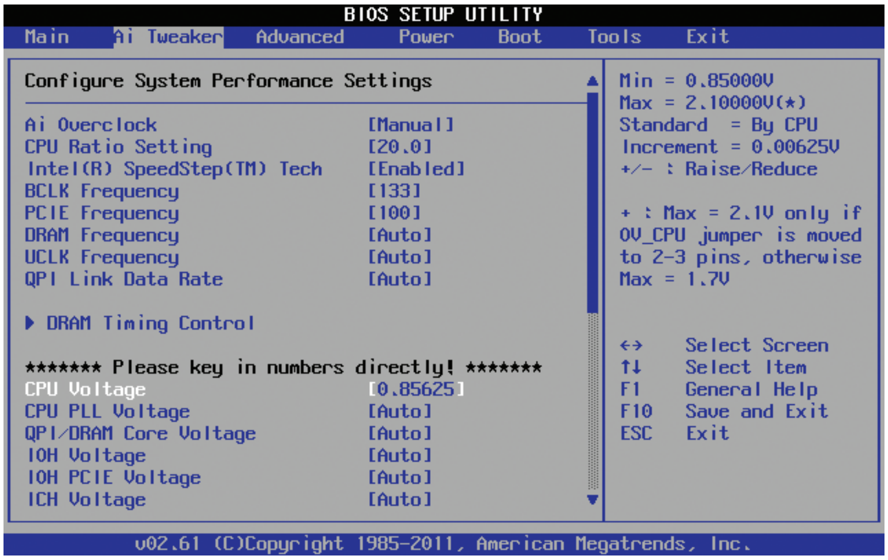
CPU makers dislike overclocking. Why would you pay more for a faster processor when you can take a cheaper, slower CPU and just make it run faster? To that end, CPU makers, especially Intel, have gone to great lengths to discourage the practice. For example, both AMD and Intel now make all of their CPUs with locked multipliers and special overspeed electronics to deter the practice.

Most people make a couple of adjustments to overclock successfully. First, through jumpers, CMOS settings, or software configuration, you would increase the bus speed for the system. Second, you often have to increase the voltage going into the CPU by just a little to provide stability. You do that by changing a jumper or CMOS setting (see Figure 6.46).



• **Figure 6.45** Heat-sink and fan assembly mounted to motherboard with screws

If you want to know exactly what type of CPU you're running, download a copy of the very popular and free CPU-Z utility from www.cpuid.com. CPU-Z gives you every piece of information you'll ever want to know about your CPU.



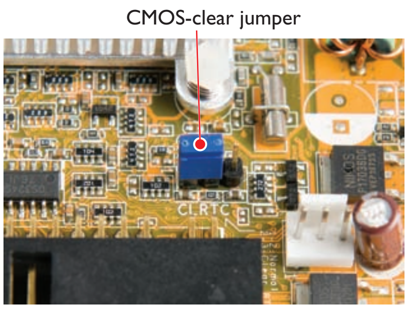
```
                         BIOS SETUP UTILITY
 Main   Ai Tweaker  Advanced   Power   Boot   Tools   Exit

 Configure System Performance Settings             Min = 0.85000V
                                                   Max = 2.10000V(*)
 Ai Overclock                  [Manual]            Standard  = By CPU
 CPU Ratio Setting             [20.0]              Increment = 0.00625V
 Intel(R) SpeedStep(TM) Tech   [Enabled]           +/- : Raise/Reduce
 BCLK Frequency                [133]
 PCIE Frequency                [100]               * : Max = 2.1V only if
 DRAM Frequency                [Auto]              OV_CPU jumper is moved
 UCLK Frequency                [Auto]              to 2-3 pins, otherwise
 QPI Link Data Rate            [Auto]              Max = 1.7V

 ▶ DRAM Timing Control
                                                   ←→    Select Screen
 ******* Please key in numbers directly! *******   ↑↓    Select Item
 CPU Voltage                   [0.85625]           F1    General Help
 CPU PLL Voltage               [Auto]              F10   Save and Exit
 QPI/DRAM Core Voltage         [Auto]              ESC   Exit
 IOH Voltage                   [Auto]
 IOH PCIE Voltage              [Auto]
 ICH Voltage                   [Auto]

          v02.61 (C)Copyright 1985-2011, American Megatrends, Inc.
```

• **Figure 6.46** Manually overriding CPU settings in the system setup utility



• **Figure 6.47** CMOS-clear jumper

Overriding the defaults can completely lock up your system, to the point where even removing and reinstalling the CPU doesn't bring the motherboard back to life. (There's also a slight risk of toasting the processor, although all modern processors have circuitry that shuts them down quickly before they overheat.) Most motherboards have a jumper setting called *CMOS clear* (see Figure 6.47) that makes the CMOS go back to default settings. Before you try overclocking on a modern system, find the CMOS-clear jumper and make sure you know how to use it! Hint: Look in the motherboard manual.

To clear the CMOS, turn off the PC. Then locate one of those tiny little plastic pieces (officially called a *shunt*) and place it over the two jumper wires for a moment. Next, restart the PC and immediately go into CMOS and restore the settings you need.

802

Troubleshooting CPUs

Troubleshooting CPU issues falls into two categories: overheating and catastrophic failures, with overheating being far more common than the latter. Once a CPU is installed properly and functioning, it rarely causes problems. The only exception is when you ask a CPU to do too much too quickly. Then you'll get a sluggish PC. The Intel Atom processor in my netbook, for example, does a great job at surfing the Web, working on e-mail, and writing stellar chapters in your favorite textbook. But if you try to play a game more advanced than Half-Life (the original, circa 1998), the machine stutters and complains and refuses to play nice.

The vast majority of problems with CPUs come from faulty installation or environmental issues that cause overheating. Very rarely will you get a catastrophic failure, but we'll look at the signs of that, too.

Symptoms of Overheating

Failure to install a CPU properly results in either nothing—that is, you push the power button and nothing at all happens—or a system lock-up in a short period of time. Because of the nature of ZIF sockets, you're almost guaranteed that the issue isn't the CPU itself, but rather the installation of the heat-sink and fan assembly. Here's a checklist of possible problems that you need to address when faced with a CPU installation problem:

1. Too much thermal paste can impede the flow of heat from the CPU to the heat sink and cause the CPU to heat up rapidly. All modern CPUs have built-in fail-safes that tell them to shut down before getting damaged by heat.
2. Not enough thermal paste or thermal paste spread unevenly can cause the CPU to heat up and consequently shut itself down.
3. Failure to connect the fan power to the motherboard can cause the CPU to heat up and shut itself down.

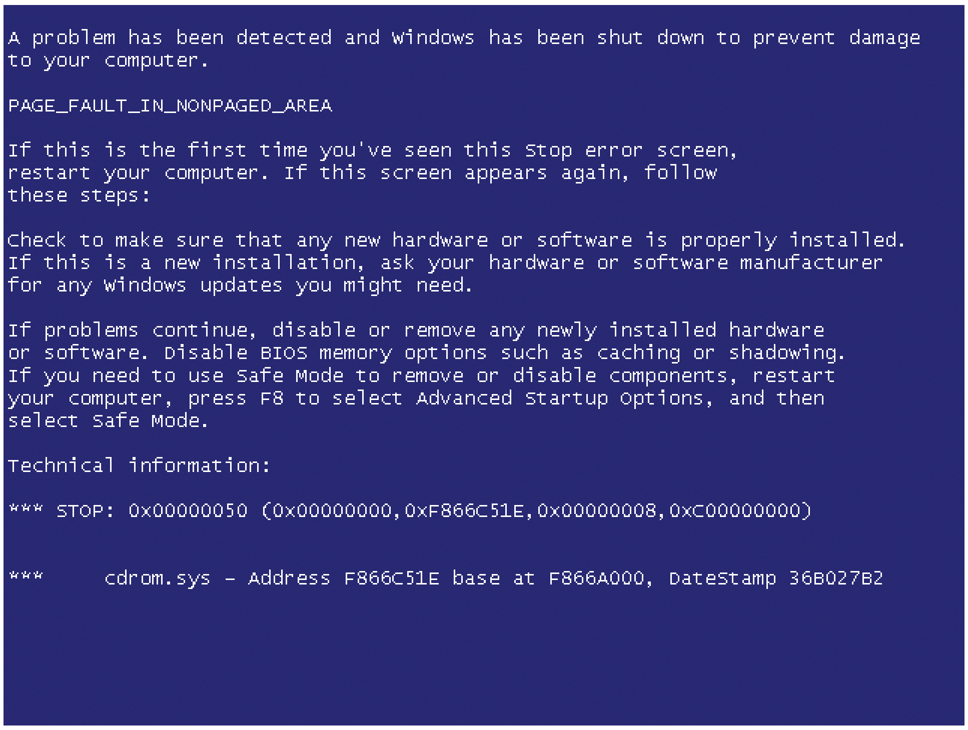
The fan and heat-sink installation failures can be tricky the first few times you encounter them (and yes, even the Alpha Geek has failed to install these things correctly). You might see the text from the system setup. You might even get into an installation of Windows before the crash happens. The key is that as soon as you put the CPU under load—that is, make it work for a living—it heats up beyond where the faulty heat-sink connection can dissipate the heat and then shuts down.

With a system that's been running fine for a while, environmental factors can cause problems. An air conditioning failure in my office last summer, deep in the heart of very hot Texas, for example, caused machines throughout the office to run poorly. Some even shut down entirely. (At that point it was time to close the doors and send the staff to the beach, but that's another story.) A client called the other day to complain about his computer continuously rebooting and running slowly. When I arrived on the scene, I found a house with seven cats. Opening up his computer case revealed the hairy truth: the CPU fan was so clogged with cat hair that it barely spun at all! A quick cleaning with a computer vacuum and a can of compressed air and he was a happily computing client.

The CPU needs adequate ventilation. The CPU fan is essential, of course, but the inside of the case also needs to get hot air out through one or more exhaust fans and cool air in through the front vent. If the intake vent is clogged or the exhaust fans stop working or are blocked somehow, the inside of the case can heat up and overwhelm the CPU cooling devices. This will result in a system running slowly or spontaneously rebooting.

Catastrophic Failure

You'll know when a catastrophic error occurs. The PC will suddenly get a Blue Screen of Death (BSoD), what's technically called a Windows Stop error (see Figure 6.48). Or the entire PC will simply stop and go black, perhaps accompanied by a loud pop. The acrid smell of burnt electronics or ozone will grace your nasal passages. You might even see trails of smoke coming out of the case. You might not know immediately that the CPU has smoked, but follow your nose. Seriously. Sniff the inside of the case until you find the strongest smell. If it's the CPU, that's bad news. Whatever electrical short hit it probably caused damage to the motherboard too, and you're looking at a long day of replacement and rebuilding.


```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure that any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0x00000000,0xF866C51E,0x00000008,0xC0000000)


***     cdrom.sys - Address F866C51E base at F866A000, DateStamp 36B027B2
```

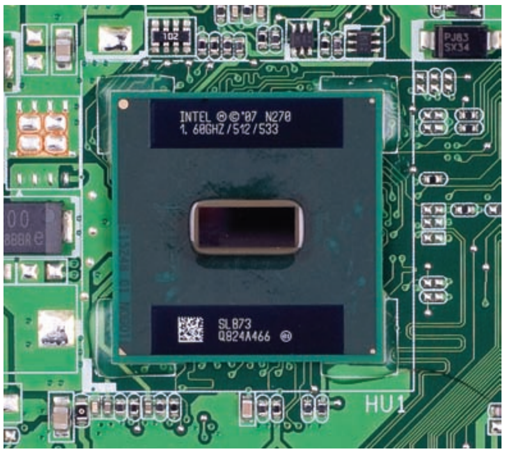
• Figure 6.48 Blue Screen of Death

Beyond A+

Intel Atom

Intel's Atom processors are very power-efficient processors designed for applications such as ultra mobile PCs, mobile Internet devices, netbooks, and low-power desktops. The Atom range of CPUs consists of both 32-bit and 64-bit models; however, only the models aimed at the low-power desktop segment support 64-bit so far. Many Atom processors also support Hyper-Threading, and there are now several dual-core models. Figure 6.49 shows an Atom processor.

As of this writing, Intel Atom processors have only been released in a package that is soldered directly to the motherboard. Atom processors are manufactured using a 45-nanometer process, and many feature Intel's SpeedStep technology to further reduce their power consumption. The Atom line of processors has become extremely popular for use in netbooks, where heat and power consumption are a primary concern.



• **Figure 6.49** Intel Atom processor

Chapter 6 Review

Chapter Summary

After reading this chapter and completing the exercises, you should understand the following facts about microprocessors.

Identify the core components of a CPU

- The central processing unit performs calculations on binary numbers to make the magic of computers work. The CPU interfaces with the motherboard and other components through the external data bus.
- CPUs contain several areas of internal memory, known as registers, in which data and addresses are stored while processing. The commands a CPU knows how to perform are dictated by its instruction set.
- A quartz crystal soldered to the motherboard and known as the system crystal provides a constant pulse known as the clock. The frequency of this pulse, or clock speed, dictates the maximum speed at which a processor can run, measured in megahertz or gigahertz. The processor typically runs at some multiple of this clock pulse, known as the internal clock speed. You can set the internal clock speed by adjusting the multiplier (which multiplies the clock speed by some number) or by configuring motherboard jumpers or making a change to a CMOS setting, or it is set automatically via CPU circuitry. Setting the clock speed higher to force the CPU to run faster than its rating is known as overclocking.

Describe the relationship of CPUs and memory

- A computer uses random access memory to take copies of programs from the hard drive and send them, one line at a time, to the CPU quickly enough to keep up with its demands. The CPU accesses any one row of RAM as easily and as quickly as any other row, which explains the "random access" part of RAM. RAM is not only randomly accessible, but also fast. By storing programs on RAM, the CPU can access and run them very quickly. RAM also stores any data that the CPU actively uses.
- The CPU communicates with RAM on the motherboard via the address bus. The number of wires comprising the address bus dictates the amount of memory the CPU can access.

Explain the varieties of modern CPUs

- Most CPUs in modern PCs are manufactured by Intel or AMD. Both companies use code names to differentiate among different models within a CPU family name. That way techs can tell the difference between, for example, a Nehalem-based Core i7 and a Sandy Bridge Core i7. Both companies offer low-power mobile versions of their desktop CPU architectures.
- Although they share the same function as early CPUs, modern CPUs offer a lot more efficiency. Seven architectural changes define modern CPUs. All are clock multipliers, running at some multiple of the system bus speed. Most current CPUs support 64-bit processing and can handle much more RAM than the 4 GB that 32-bit processors supported.
- With parallel execution and multicore processing, modern CPUs can handle many simultaneous applications efficiently. One Intel-only parallel execution technology, Hyper-Threading, makes a single CPU function like two CPUs if the OS and applications support the technology. A multicore CPU, in contrast, functions as multiple CPUs regardless of the OS, though the applications still need optimization.

Select and install a CPU

- Consult your motherboard documentation to see which CPUs are compatible with your system. Not all CPUs are compatible with all motherboards.
- Cooling is critical. Make sure you have a fan rated to work with your CPU.
- A CPU fits only one way in the ZIF socket. Don't force it. If you find the CPU will not seat properly, take a second look at the orientation markers and verify the CPU is in the correct direction. Check the pins on the underside to make sure none are bent.

- There should be a small amount of heat-sink compound between the CPU and heat-sink and fan assembly. If your fan came with the compound already applied, be sure to remove the protective tape covering the compound before attaching the fan to the CPU. If you are using your own heat dope from a tube, spread it thinly and evenly.

Troubleshoot CPUs

- If your computer is continuously rebooting or won't start up, you may have a CPU cooling problem. Open your case up and make sure the heat-sink is properly seated and all the fans are running.
- If you get a Blue Screen of Death or random shutdown, check for the smell of burnt electronics. If you smell something burning, you may have had a catastrophic CPU failure. The only thing to do to fix that is to replace the CPU.

Key Terms

64-bit processing *(174)*
address bus *(166)*
arithmetic logic unit (ALU) *(176)*
backside bus *(178)*
binary *(158)*
bit *(165)*
byte *(165)*
cache *(177)*
central processing unit (CPU) *(156)*
clock cycle *(161)*
clock speed *(161)*
clock wire *(161)*
code name *(171)*
dual-core *(180)*
dynamic RAM (DRAM) *(165)*
external data bus (EDB) *(158)*
floating point unit (FPU) *(176)*
frontside bus *(178)*
graphics processing unit (GPU) *(181)*
instruction set *(160)*
integrated memory controller (IMC) *(181)*
liquid cooling *(186)*
machine language *(160)*
memory *(164)*
memory controller chip (MCC) *(166)*
microprocessor *(156)*
multicore processing *(181)*
overclocking *(189)*
parallel execution *(175)*
pipeline *(175)*
program *(164)*
random access memory (RAM) *(165)*
registers *(159)*
static RAM (SRAM) *(176)*
system crystal *(162)*
thermal compound *(187)*
throttling *(173)*
wait state *(176)*
zero insertion force (ZIF) socket *(184)*

Key Term Quiz

Use the Key Terms list to complete the sentences that follow. Not all terms will be used.

1. All of the machine language commands that the CPU understands make up the CPU's ________________.
2. By lifting the arm on the ________________, you can easily install a PGA CPU.
3. Computers use ________________ for main system memory.
4. Computers use the ________________ numbering system.
5. Areas inside the CPU where it temporarily stores internal commands and data while it is processing them are called ________________.

6. Divided into L1 and L2, the ________________ consists of a small amount of ________________ that serves as a holding area. This provides data to the CPU faster than getting it from regular memory or when RAM is unavailable because of refreshes.
7. A CPU with a(n) ________________ can handle RAM directly.
8. If a stock heat-sink and fan assembly can't handle an overclocked CPU, a solution based on ________________ might do the trick.
9. In a process known as ________________, a CPU can slow itself down in low-demand times or if it gets too hot.
10. An Intel Core 2 Duo is an example of a(n) ________________ processor.

Multiple-Choice Quiz

1. What device enables a PC to retrieve a specific row of data from system memory and place it on the external data bus?
   - **A.** Advanced micro device
   - **B.** Arithmetic logic unit
   - **C.** Floating point processor
   - **D.** Memory controller chip
2. Which of the following statements is true?
   - **A.** The address bus enables the CPU to communicate with the MCC.
   - **B.** The external data bus enables the CPU to communicate with the MCC.
   - **C.** The address bus enables the CPU to communicate with the hard drive.
   - **D.** The system bus enables the CPU to communicate with the memory.
3. What do 64-bit processors expand beyond what 32-bit processors have?
   - **A.** System bus
   - **B.** Frontside bus
   - **C.** Address bus
   - **D.** Registers
4. What is the first stage in a typical four-stage CPU pipeline?
   - **A.** Decode
   - **B.** Execute
   - **C.** Fetch
   - **D.** Write
5. Which of the following terms are measures of CPU speed?
   - **A.** Megahertz and gigahertz
   - **B.** Megabytes and gigabytes
   - **C.** Megahertz and gigabytes
   - **D.** Frontside bus, backside bus
6. Which of the following statements is true?
   - **A.** If you have an AMD-compatible motherboard, you can install a Celeron processor.
   - **B.** You should always overclock your CPU.
   - **C.** As the size of the address bus increases, the amount of RAM the CPU can use decreases.
   - **D.** You can upgrade your CPU if you make sure that a new CPU will fit into the socket on your motherboard.
7. Which CPU feature enables the microprocessor to support running multiple operating systems at the same time?
   - **A.** Clock multiplying
   - **B.** Caching
   - **C.** Pipelining
   - **D.** Virtualization support
8. Into which socket could you place an AMD Phenom II?
   - **A.** Socket LGA 775
   - **B.** Socket LGA 1154
   - **C.** Socket AM2+
   - **D.** Socket A
9. Joey has a motherboard that advertises support for *x*64 processors. Which CPU will definitely work in that motherboard?
   - **A.** Athlon 64 X2
   - **B.** Core 2 Duo
   - **C.** Core i7
   - **D.** There's not enough information to answer the question.

10. Which feature enables a single-core CPU to function like two CPUs?

- **A.** Hyper-Threading
- **B.** SpeedStep
- **C.** Virtualization
- **D.** *x*64

11. What connects on the backside bus?

- **A.** CPU, MCC, RAM
- **B.** CPU, MCC, L1 cache
- **C.** CPU, L1 cache
- **D.** CPU, L2 cache

12. What improvement have CPU manufacturers put into processors to deal with pipeline stalls?

- **A.** Added multiple pipelines
- **B.** Increased the speed of the SRAM
- **C.** Created new die sizes with more pins
- **D.** Bundled better fans with their retail CPUs

13. What steps do you need to take to install an Athlon 64 X2 CPU into an LGA 775 motherboard?

- **A.** Lift the ZIF socket arm; place the CPU according to the orientation markings; snap on the heat-sink and fan assembly.
- **B.** Lift the ZIF socket arm; place the CPU according to the orientation markings; add a dash of heat dope; snap on the heat-sink and fan assembly.
- **C.** Lift the ZIF socket arm; place the CPU according to the orientation markings; snap on the heat-sink and fan assembly; plug in the fan.
- **D.** Take all of the steps you want to take because it's not going to work.

14. A client calls to complain that his computer starts up, but crashes when Windows starts to load. After a brief set of questions, you find out that his nephew upgraded his RAM for him over the weekend and couldn't get the computer to work right afterward. What could be the problem?

- **A.** Thermal compound degradation
- **B.** Disconnected CPU fan
- **C.** Bad CPU cache
- **D.** There's nothing wrong. It usually takes a couple of days for RAM to acclimate to the new system.

15. Sam has installed a new CPU in a client's computer, but nothing happens when he pushes the power button on the case. The LED on the motherboard is lit up, so he knows the system has power. What could the problem be?

- **A.** He forgot to disconnect the CPU fan.
- **B.** He forgot to apply thermal compound between the CPU and the heat-sink and fan assembly.
- **C.** He used an AMD CPU in an Intel motherboard.
- **D.** He used an Intel CPU in an AMD motherboard.

Essay Quiz

1. It is important for the CPU to stay cool. A number of technical advances have been made in the design of CPUs, along with various devices made to keep the CPU from overheating. Discuss at least two cooling features or cooling options.
2. Where do code names for CPUs come from? Write up a comparison of the code names used by Intel and AMD for their processors. What other aspects of computing use code names?
3. Compare parallel execution with multicore processing. How are they similar? How do they differ?
4. Write a brief essay on the benefits and risks of overclocking a CPU.
5. Do we need 64-bit computing? What can you do with 64-bit that you can't do with 32-bit? What applications do you use on a regular basis that could take advantage of 64-bit versions? If the answer is "none," explain.

Lab Projects

• Lab Project 6.1

Perhaps newer and faster CPUs have come out recently. Go to www.intel.com and www.amd.com and investigate the newest CPUs for desktop computers from each manufacturer. Write a paragraph comparing the newest Intel CPU with the newest AMD CPU. Try to include the following information:

- What is the size of the system bus?
- What is the size of the address bus?
- What is the speed of the CPU?
- What are the sizes of the L1 and L2 caches?
- Does either CPU offer an L3 cache?
- What kind of chip package houses each CPU?
- What kind of socket does each use?
- What other new features does each site advertise for its newest CPU?

• Lab Project 6.2

Imagine that you are going to buy components to build your own computer. What processor will you use? Typically, the latest and greatest CPU is a lot more expensive than less recent models. Intel processors usually cost more than comparable AMD processors. Check CPU features and prices in newspapers or magazines or on the Internet at a site such as www.newegg.com. Decide what CPU you want to use for your computer. Write a paragraph explaining why you selected it and how much you will spend for the CPU.

• Lab Project 6.3

If your school hardware lab has motherboards and processors for hands-on labs, practice removing and installing LGA and PGA processors on the motherboards. Take note of how the mechanical arm on a ZIF socket works. Answer the following about your experience:

- How do you know in which direction to place the CPU?
- How does the mechanical arm lift up? Does it lift straight up or must it clear a lip?
- What effect does lifting the arm have on the socket?
- How does the ZIF socket hold the CPU in place?