CHAPTER 3

CPUs

In this chapter, you will learn how to

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

The *central processing unit* (*CPU*), also called the *microprocessor*, is a single silicon-based electronic chip that makes your computer…well, a computer. Desktop computers, laptops, smartphones, even tiny computers in a smartwatch or a washing machine have a CPU. A CPU invariably hides on the motherboard below a heat sink and often a fan assembly as well. CPU makers name their microprocessors in a fashion similar to the automobile industry: CPUs get a make and a model, such as Intel Core i9, Qualcomm Snapdragon 835, or AMD Ryzen 7. 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 fourth section gets into practical work, selecting and installing CPUs. The final section covers troubleshooting CPUs in detail.

EXAM TIP CompTIA only uses the term *CPU*, not *microprocessor*. Expect to see CPU on the 1001 exam.

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 technician needs to understand some basic CPU functions to support computing devices, 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 Intel 8088, invented in the late 1970s. This CPU defined the idea of the modern microprocessor and contains the same basic parts used in the most advanced CPUs today.

The Man in the Box

Begin by visualizing the CPU as a man in a box (see Figure 3-1). This is one clever guy. He can perform virtually any mathematical function, manipulate data, and give answers *very quickly*.

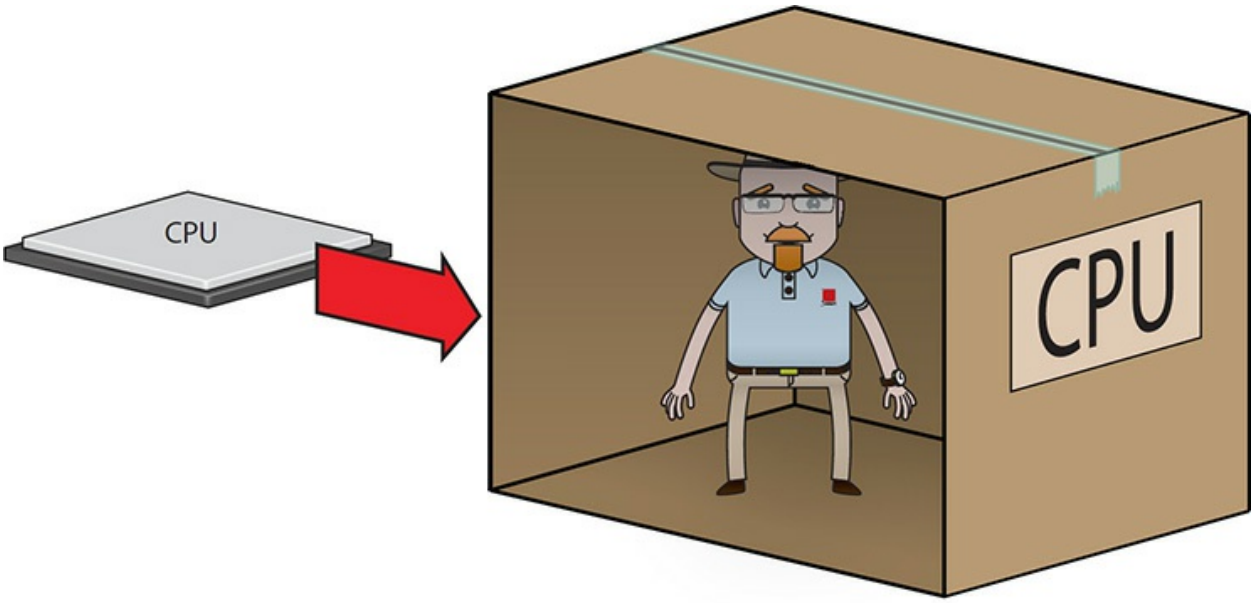

Figure 3-1 Imagine the CPU as a man in a box.

This guy is potentially very useful to us, but there's a catch—he lives in a tiny, closed box. Before he can work with us, we must come up with a way to exchange information with him (see Figure 3-2).

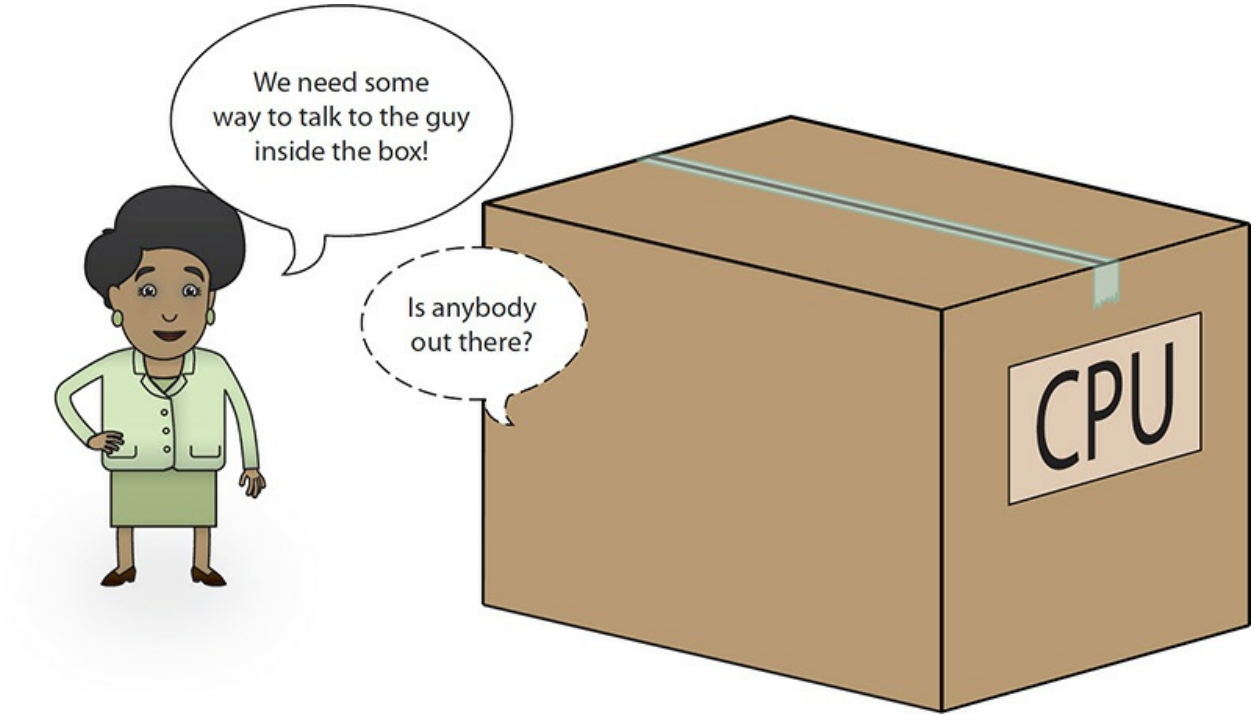

Figure 3-2 How do we talk with the Man in the Box?

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 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 3-3 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.

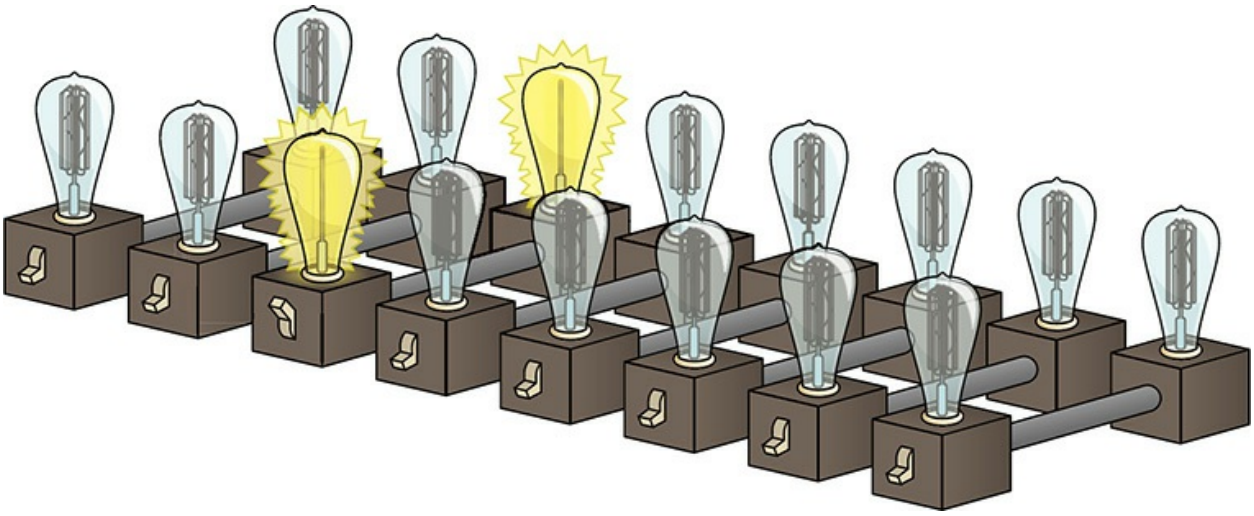

Figure 3-3 Cutaway of the external data bus—note that one light bulb pair is on.

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 many CPUs (see Figure 3-4). 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.

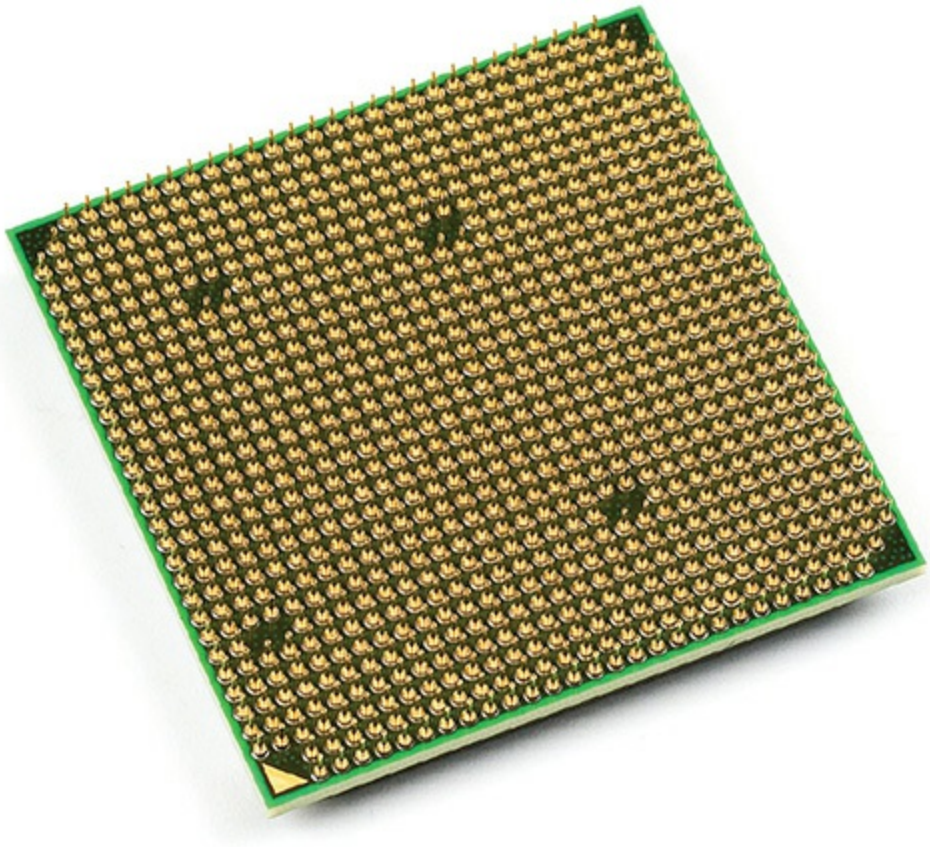

Figure 3-4 Close-up of the underside of a CPU

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 3-5). 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."

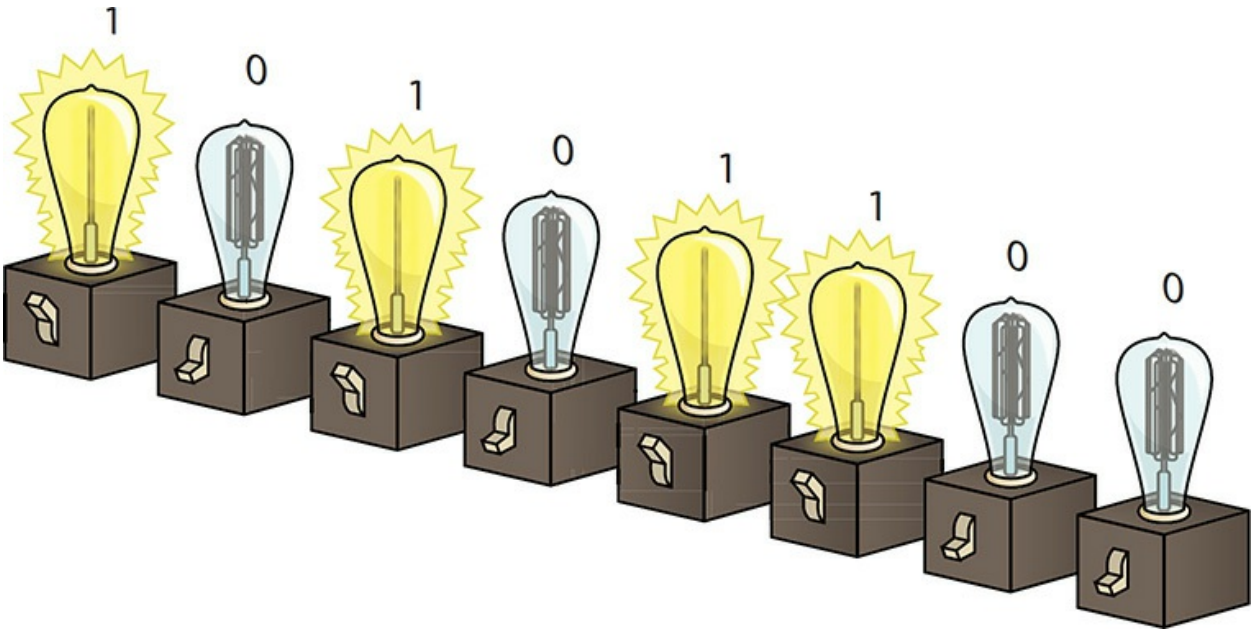

Figure 3-5 Here “1” means on, “0” means off.

In computers, wires repeatedly turn 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 computing, but this is a great place to start.

Registers

The Man in the Box provides good insight into the workspace inside a CPU. The EDB gives you a way to 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 3-6) and store internal commands and data.

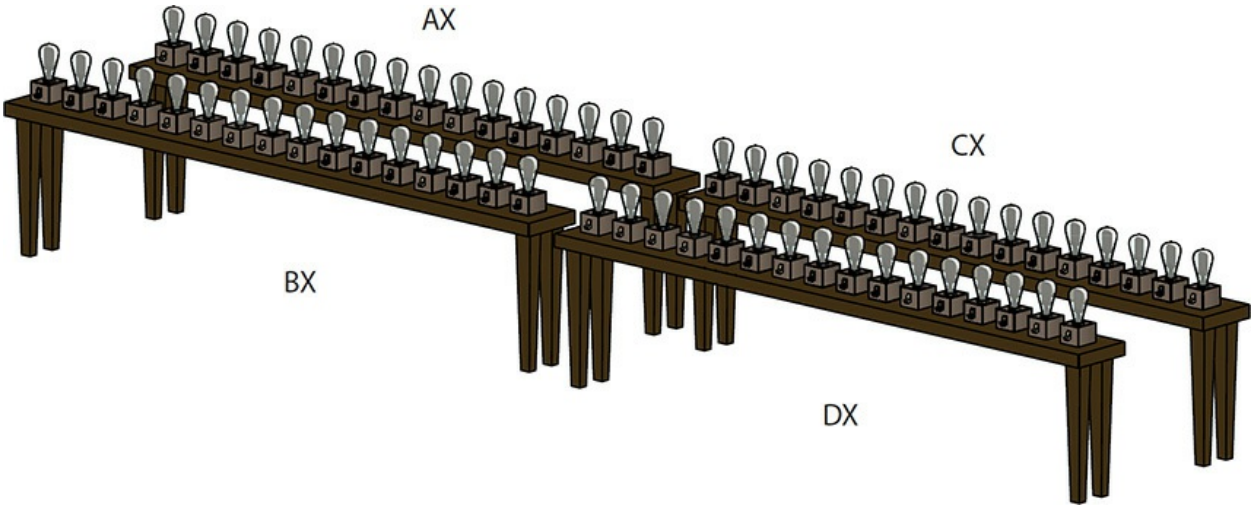

Figure 3-6 The four general-purpose registers

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.

NOTE The 8088 was the first CPU to use the four AX–DX general-purpose registers, and they still exist in even the latest CPUs. (But they have a lot more light bulbs!) In 32-bit processors, the registers add an E for extended, so EAX, EBX, and so on. The 64-bit registers get an R (I don't know why), thus RAX, RBX, and so on.

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 3-7 shows the codebook we'll use. We'll give one copy to him and make a second for us.

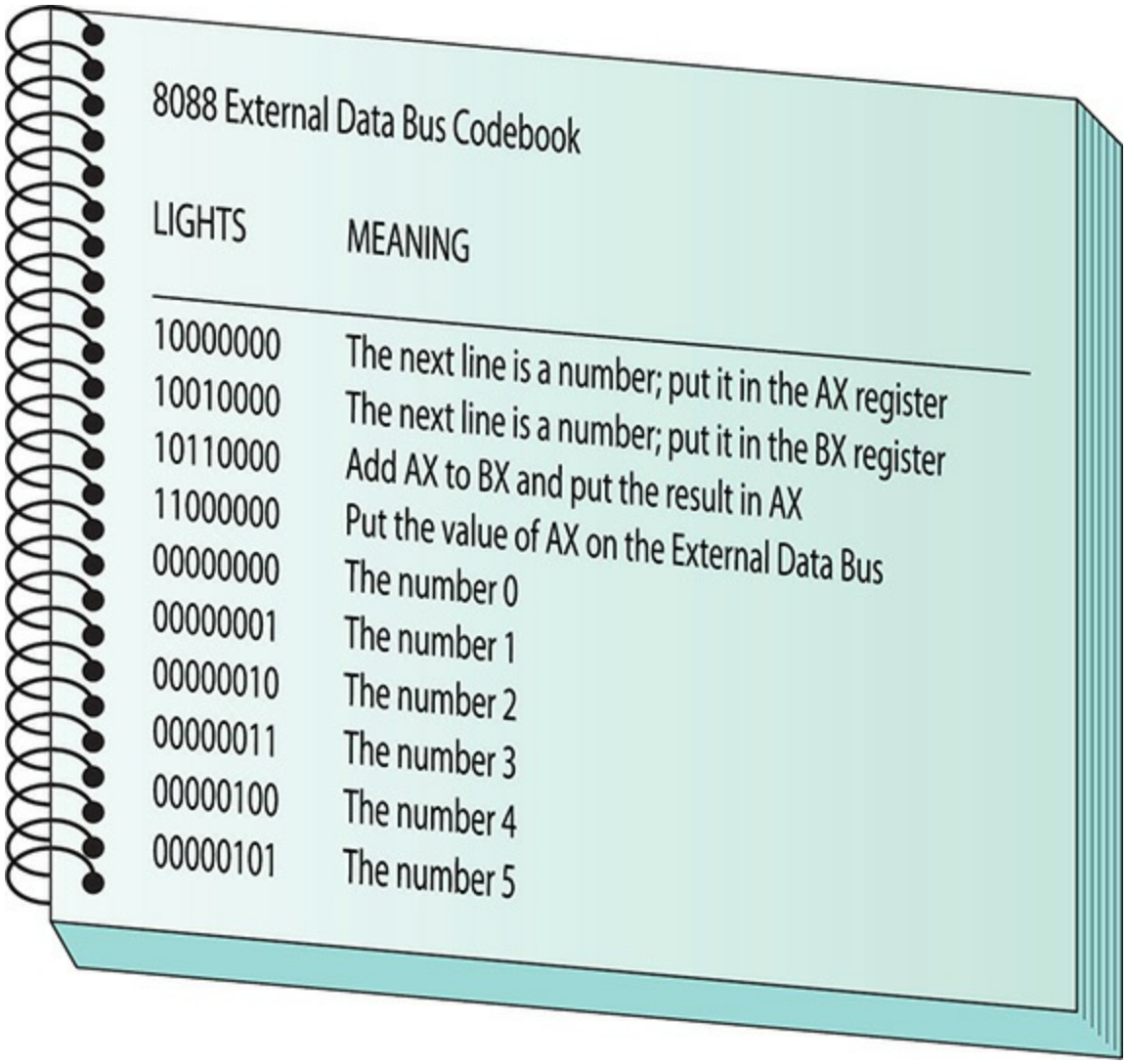

Figure 3-7 CPU codebook

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 3-8).

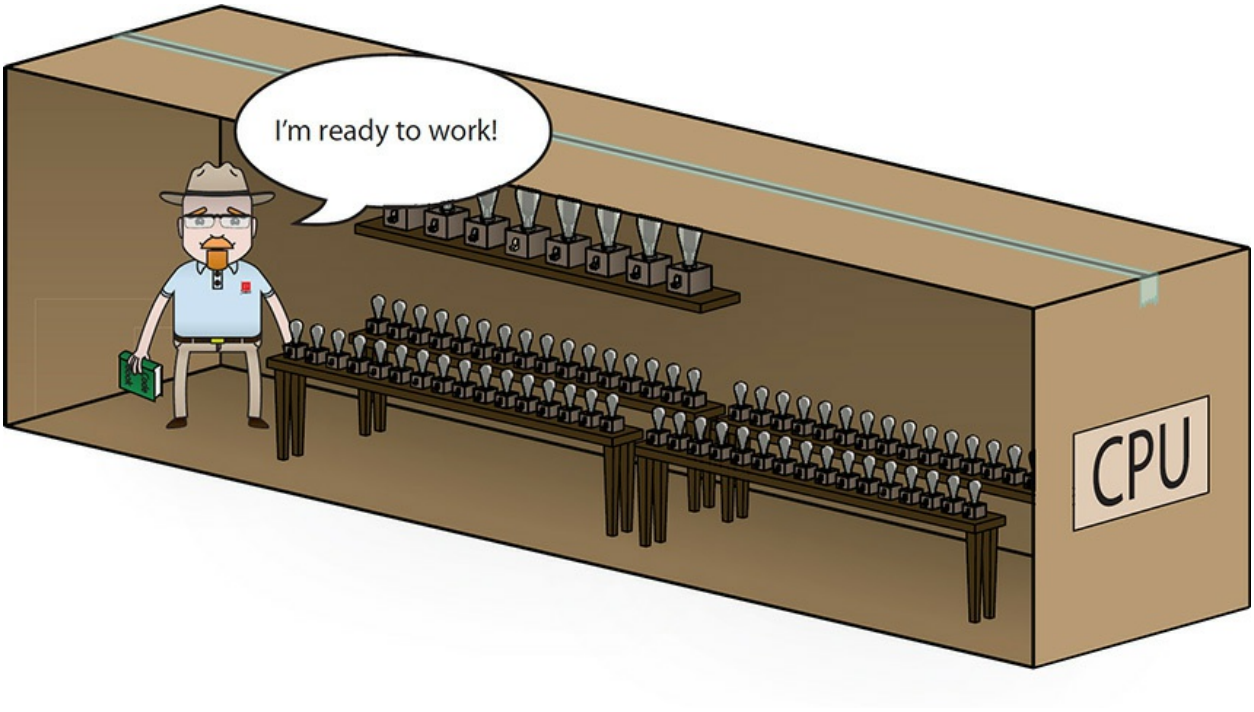

Figure 3-8 The CPU so far

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?

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 3-9).

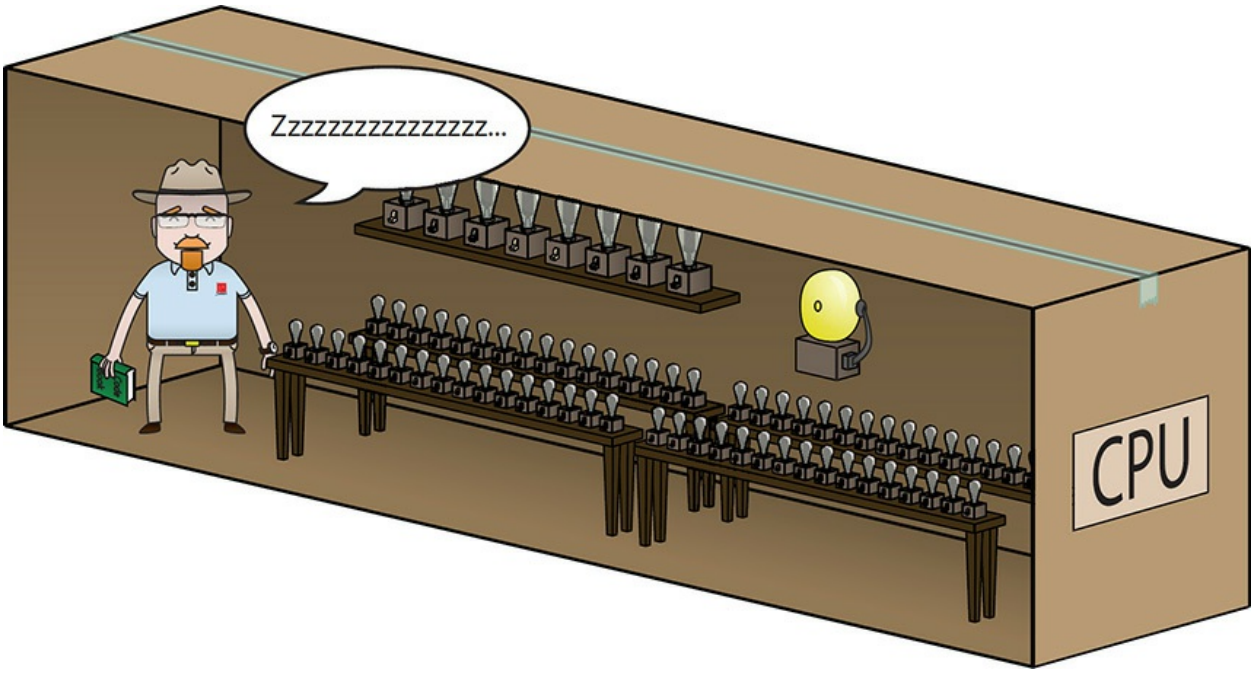

Figure 3-9 The CPU does nothing until activated by the clock.

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. In fact, a CPU may require hundreds of clock cycles to process some commands (see Figure 3-10).

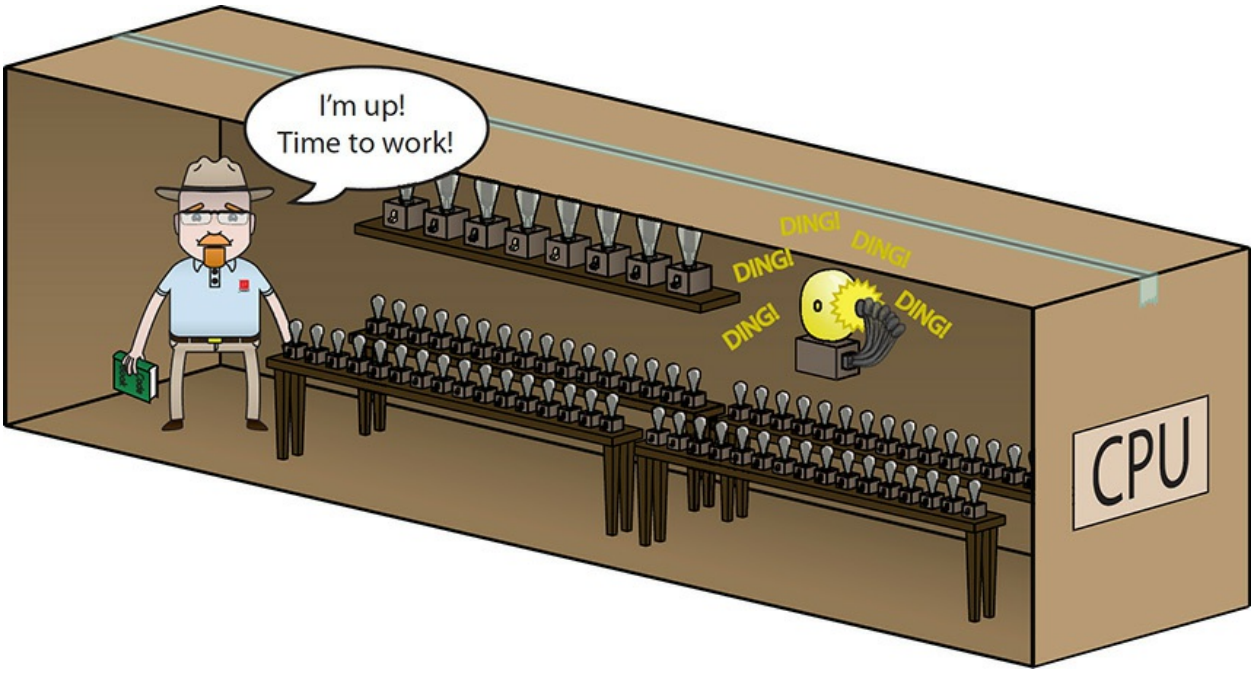

Figure 3-10 The CPU often needs more than one clock cycle to get a result.

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 per second), extremely slow by modern standards, but still a pretty big number compared to using a pencil and paper. High-end CPUs today run at speeds in excess of 5 GHz (5 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. Many CPU models have the clock speed printed clearly (see Figure 3-11). Other models might have a cryptic code.

Figure 3-11 Clock speed printed on a CPU (3.30GHZ)

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 3-12).

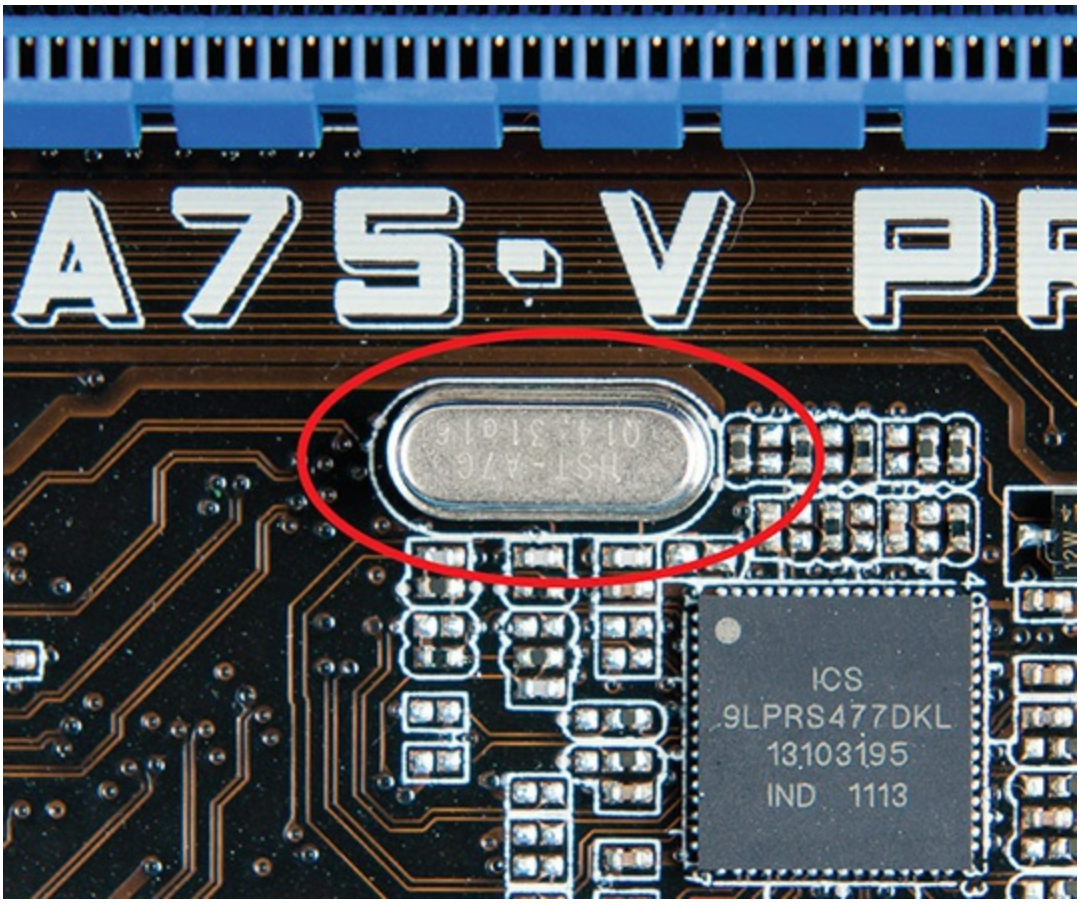

Figure 3-12 One of many types of system crystals

NOTE 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.

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 computer is turned on, the quartz oscillator, through the clock chip, 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.

NOTE 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.

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 3-13 is a diagram of an 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.

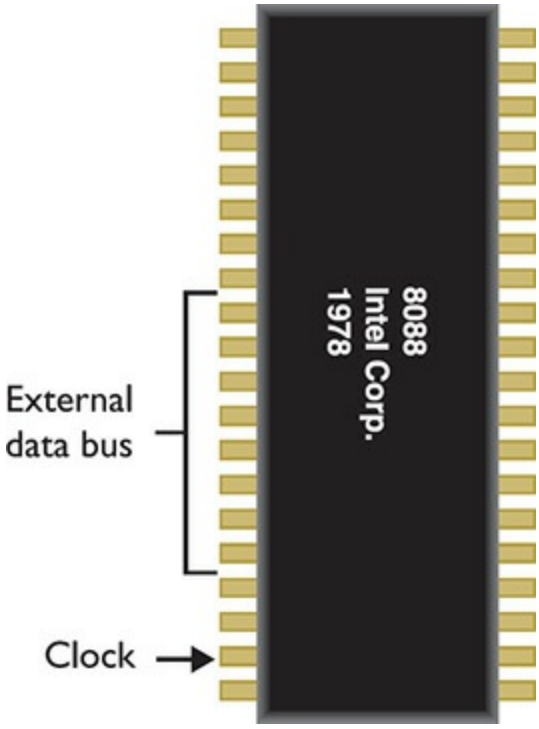

Figure 3-13 Diagram of an Intel 8088 showing the external data bus and clock wires

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 3-7 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.

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 store not only programs but also 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 4, “RAM,” develops the concept in detail, so for now let’s look at RAM as an electronic spreadsheet, like one you can generate in Microsoft Excel (see Figure 3-14). 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 8 bits across to match the EDB of the 8088. Each row of 8 bits is called a *byte*. In PCs, 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 3-14 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 3-15). In Chapter 4, 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.

Figure 3-15 Typical RAM

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. 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 8 through 10 discuss permanent storage in intimate detail.

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 3-16).

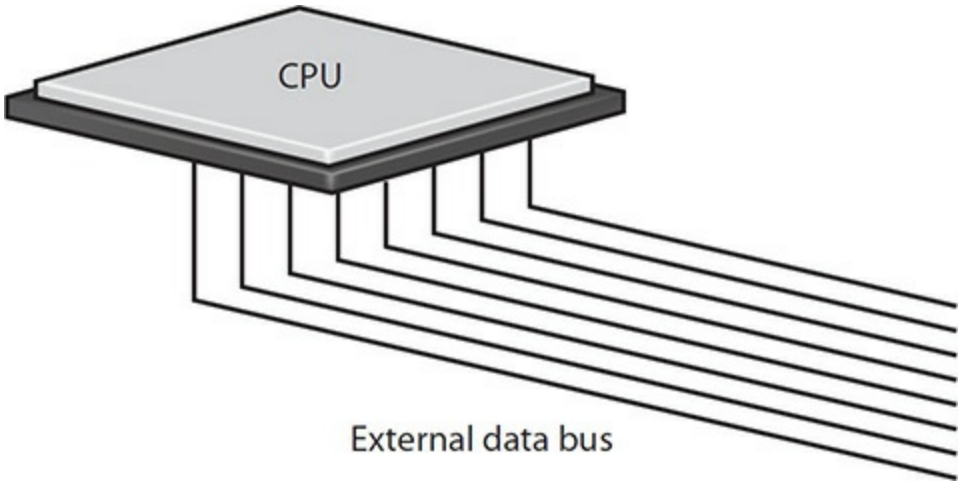

Figure 3-16 Extending the EDB

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. 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 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 3-17).

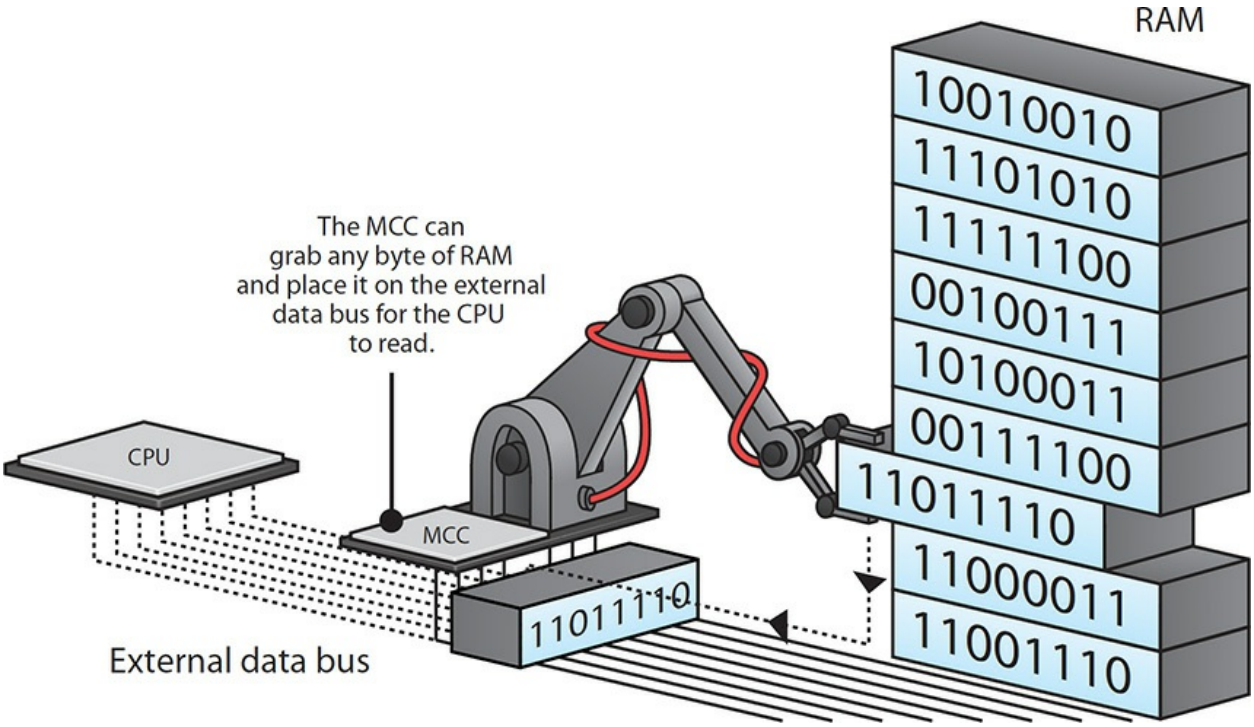

Figure 3-17 The MCC grabs a byte of RAM.

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 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 3-18).

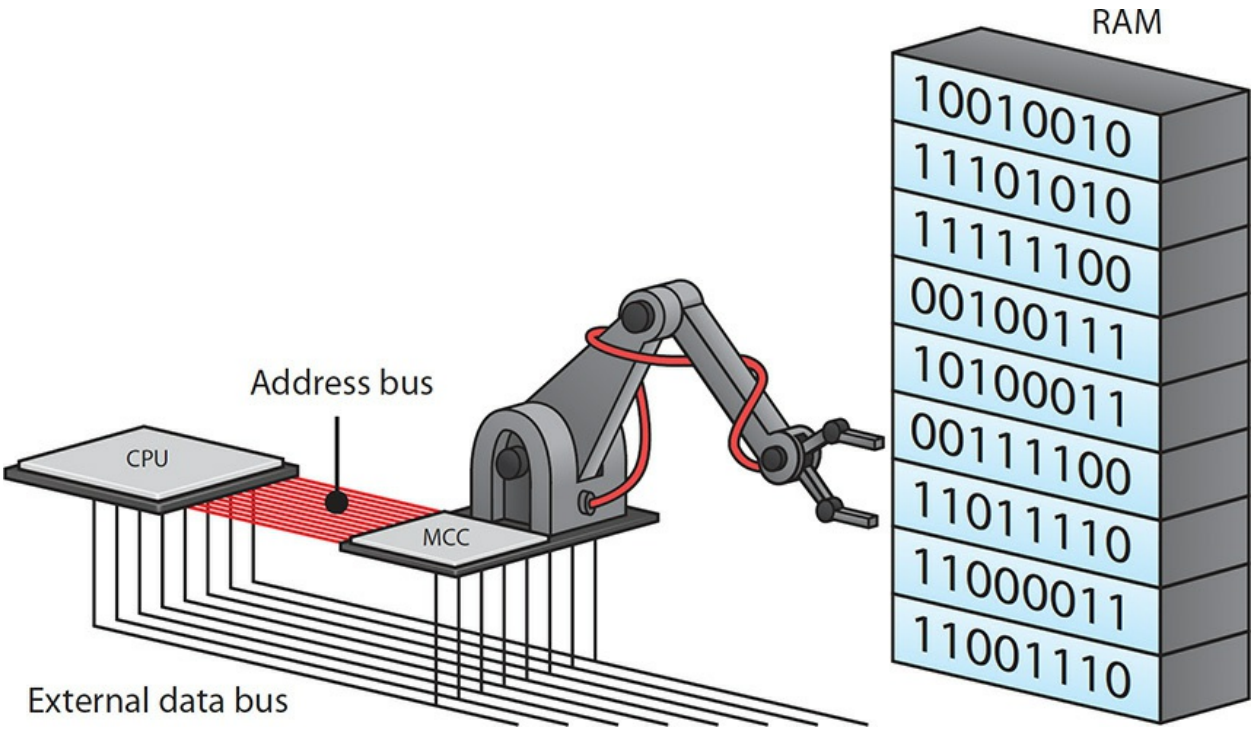

Figure 3-18 Address bus

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?

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 65,536 bytes—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 computing. The following list explains.

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")

Metric System and Computer Memory

There's a problem with that list you just read. If you asked a metric system expert for explanation, she would say that a *kilo* is equal to exactly *1000*, not *1024*! Am I lying to you?

Well, yes, I am, but not out of malice. I'm just the messenger of yet another weird aspect to computing. Here's what happened, a long time ago.

In the early days of computing there arose a need to talk about large values, but the words hadn't been invented. In one case, the memory address folks were trying to describe permutations. They used values based on powers of two as just described. No one had ever invented terms for 1024 or 1,048,576, so they used kilo and mega, as 1000 was close enough to 1024 and 1,000,000 was close enough to 1,048,576.

In the meantime, computer people measuring quantities such as CPU speeds and hard drive capacities didn't count with powers of two. They just needed regular 1000 for kilo and 1,000,000 for mega.

From the early 1980s until around 1990, nobody cared about this weird thing where one word could mean two values. Everything was fine until the math nerds and the attorneys started making trouble. To fix this, in 1998 the International Electrotechnical Committee (IEC) invented special prefixes for binary values I call the *ibis* (pronounced *eee-bees*).

1 kibi = 2^{10} = 1024 (abbreviated as "Ki")

1 mebi = 2^{20} = 1,048,576 (abbreviated as "Mi")

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

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

To follow this revised naming convention, you should say, "the 8088 processor could address one mebibyte (MiB) of memory." The problem is that *no one* but math nerds use these ibis. If you buy RAM, the manufacturers use the term gigabyte even though technically they should use gibibyte. Welcome to the weird world of counting in IT. Let's get back to memory.

NOTE The jury is still out on correct pronunciation of the ibis. You will

find ardent supporters of “keebeebyte” and equally passionate supporters of “kehbeebyte.” It doesn’t really matter, because the rest of us just say “kilobyte.”

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 the address bus wires, it wants the first line of RAM; when it turns on all the wires, it wants the 1,048,576th line of RAM. Obviously, the address bus also addresses all 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.

NOTE Bits and bytes are abbreviated differently. Bits get a lowercase b, whereas bytes get a capital B. So for example, 4 Kb is 4 kilobits, but 4 KB is 4 kilobytes. The big-B little-b standard applies all the way up the food chain, so 2 Mb = 2 megabits; 2 MB = 2 megabytes; 4 Gb = 4 gigabits; 4 GB = 4 gigabytes; and so on.

1001

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.

Developers

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. 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.

NOTE The ever-growing selection of mobile devices, such as the Apple iPhone and iPad and most Android devices, use a CPU architecture developed by ARM Holdings, called *ARM*. ARM-based processors use a simpler, more energy-efficient design, the reduced instruction set computing (RISC) architecture. They can'T match the raw power of the Intel and AMD complex instruction set computing (CISC) chips, but the savings in cost and battery life make ARM-based processors ideal for mobile devices. (Note that the clear distinction between RISC and CISC processors has blurred. Each design today borrows features of the other design to increase efficiency.)

ARM Holdings designs ARM CPUs, but doesn't manufacture them. Many other companies—most notably, Qualcomm—license the design and manufacture their own versions. See Chapter 24, "Understanding Mobile Devices."

Intel

Intel Corporation thoroughly dominates 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 Core, Pentium, and Celeron brands. Their high-end server chips are called Xeon.

AMD

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? 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 3-19). You could yank an Intel CPU out of a system and snap in an AMD CPU—no problem!

Figure 3-19 Electronically 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. You 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 3-1 displays many of the current product lines and names.

Market	Intel	AMD
Enthusiast	Core i7/i9	Ryzen, Ryzen Threadripper
Mainstream desktop	Core i7/i5/i3	A-Series Pro, Ryzen
Budget desktop	Pentium, Celeron	A-Series, FX
Portable/Mobile	Core i7/i5/i3 (mobile), Mobile Celeron	Ryzen, A-Series
Server	Xeon	Opteron, EPYC
Workstation	Xeon	Ryzen PRO, Ryzen Threadripper

Table 3-1 Current Intel and AMD Product Lines and Names

Microarchitecture

Intel and AMD continually develop faster, smarter, and generally more capable CPUs. In general, each company comes up with a major new design, called a *microarchitecture*, about every three years. They try to minimize the number of model names in use, however, most likely for marketing purposes. 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 3-20). 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.

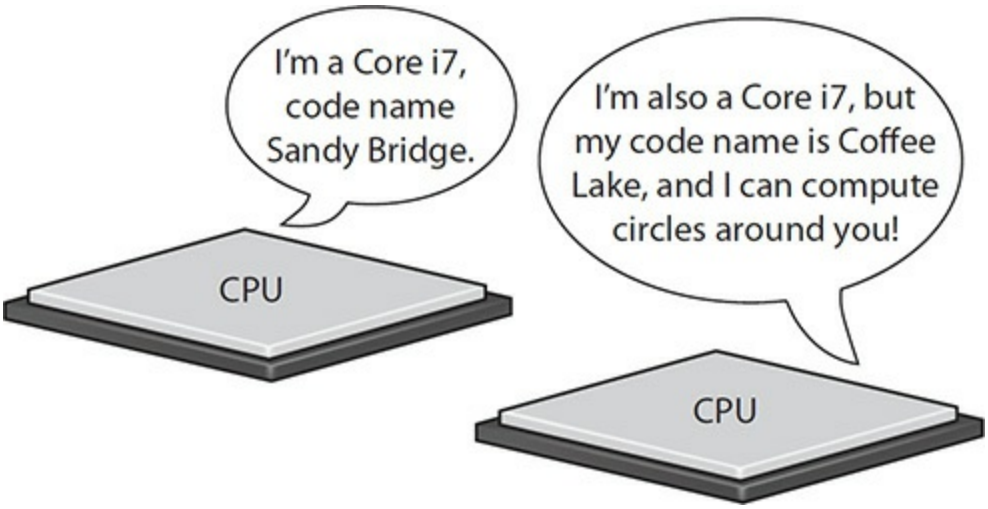

Figure 3-20 Same branding, but different capabilities

Intel released the first Core i7 in the summer of 2008. By spring of 2012, the original microarchitecture—code 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. Just about every year since then has seen a new Core i7 based on improved architectures with different code names such as Ivy Bridge, Haswell, Broadwell, and so on. (And I'm simplifying the variations here.)

NOTE The processor number helps a lot when comparing processors once you decode the meanings. We need to cover more about modern processors before introducing processor numbers. Look for more information in the upcoming section, "Deciphering Processor Numbers."

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

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:

- http://arstechnica.com
- www.anandtech.com
- www.tomshardware.com
- www.bit-tech.net

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

- www.wikipedia.org

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

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 3-21). These mobile CPUs consume much less power than their desktop counterparts. They also 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*.

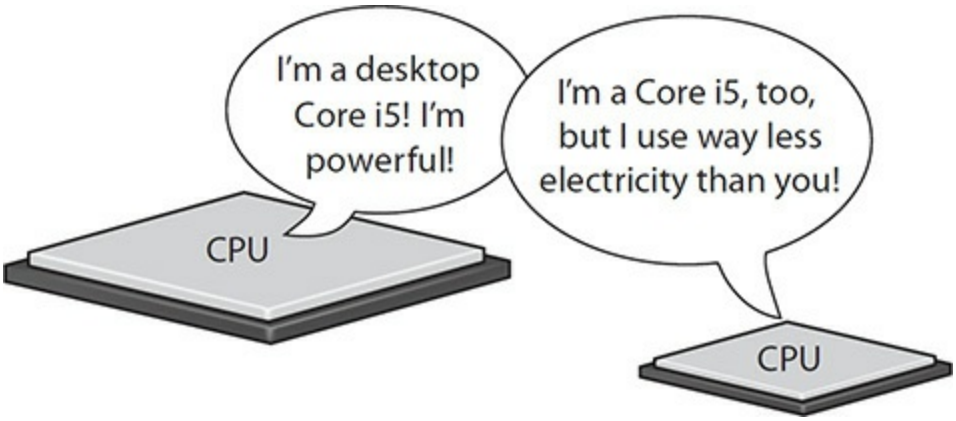

Figure 3-21 Desktop vs. mobile, fight!

Unfortunately this picture gets more complicated when you throw in heat. Because most portable and mobile computing devices are very compact, they can't dissipate heat as quickly as a well-cooled desktop system. Mobile CPUs can scale up to handle demanding tasks, but they'll start accumulating heat quickly. As this heat nears levels that could damage the CPU, it will engage in *thermal throttling* to protect itself. A system trying to do demanding work with only a fraction of its full power available may grind to a halt!

NOTE The industry describes how much heat a busy CPU generates with a figure (measured in watts) called its *thermal design power* (*TDP*). The TDP can give you a rough idea of how much energy a CPU draws and what kind of cooling it will need. It can also help you select more efficient CPUs.

TDP has been trending down over time (especially in recent years), but it may help to have a sense of what these values look like in the real world. The CPUs in a smartphone or tablet typically have a TDP from 2 to 15 watts, laptop CPUs range from 7 to 65 watts, and desktop CPUs tend to range from 50 to 140 watts.

Many of the technologies developed for mobile processors migrate back into their more power-hungry desktop siblings. That's a bonus for the planet (and maybe your power bill).

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 eight features:

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

Clock Multipliers

All modern CPUs run at some multiple of the system clock speed. The system bus on my Ryzen 7 machine, for example, runs at 100 MHz. The clock multiplier goes up to ×32 at full load to support the 3.2 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 3-22 shows a nifty program called CPU-Z displaying my CPU details. Note that all I'm doing is typing at the moment, so the CPU has dropped the clock multiplier down to ×15.5 and the CPU core speed is only 1546 MHz.

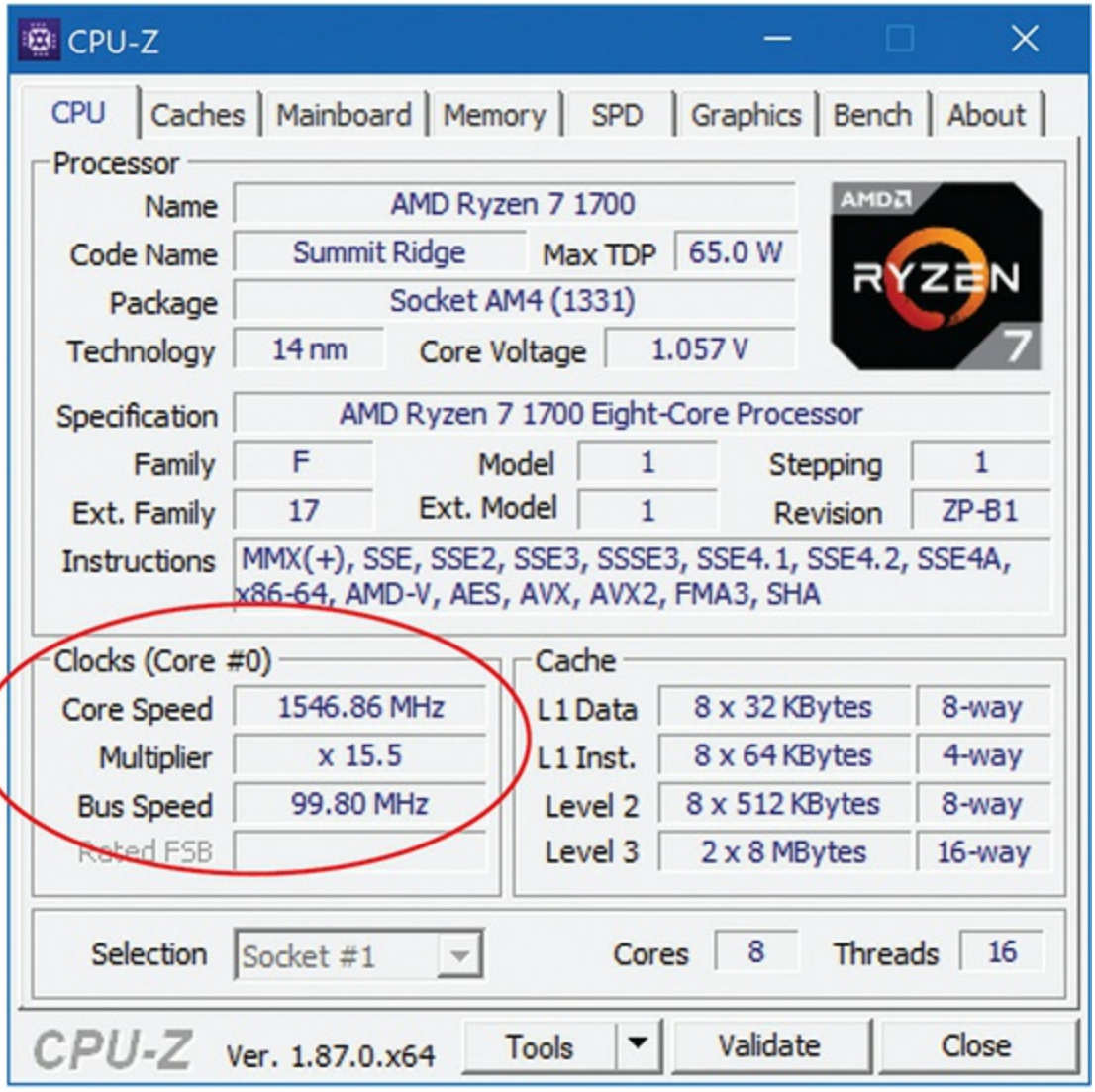

Figure 3-22 CPU-Z showing the clock speed, multiplier, and bus speed of a Ryzen 7 processor hardly breaking a sweat

Try This! CPU-Z

Imagine a scenario where you're dumped into an office full of unfamiliar PCs. There's no documentation about the systems at all, so your boss tells you to get cracking and find out as much as possible about each PC ASAP. Try this! 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 a CPU. Copy it to a thumb drive, then insert it into a bunch of different computers. (Ask permission, of course!) What kinds of processors do you find in your neighbors' computers? What can you tell about the different capabilities?

The clock speed and the multiplier on early clock-multiplying systems had to be manually configured via jumpers or dual in-line package (DIP) switches on the motherboard (see Figure 3-23). 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.)

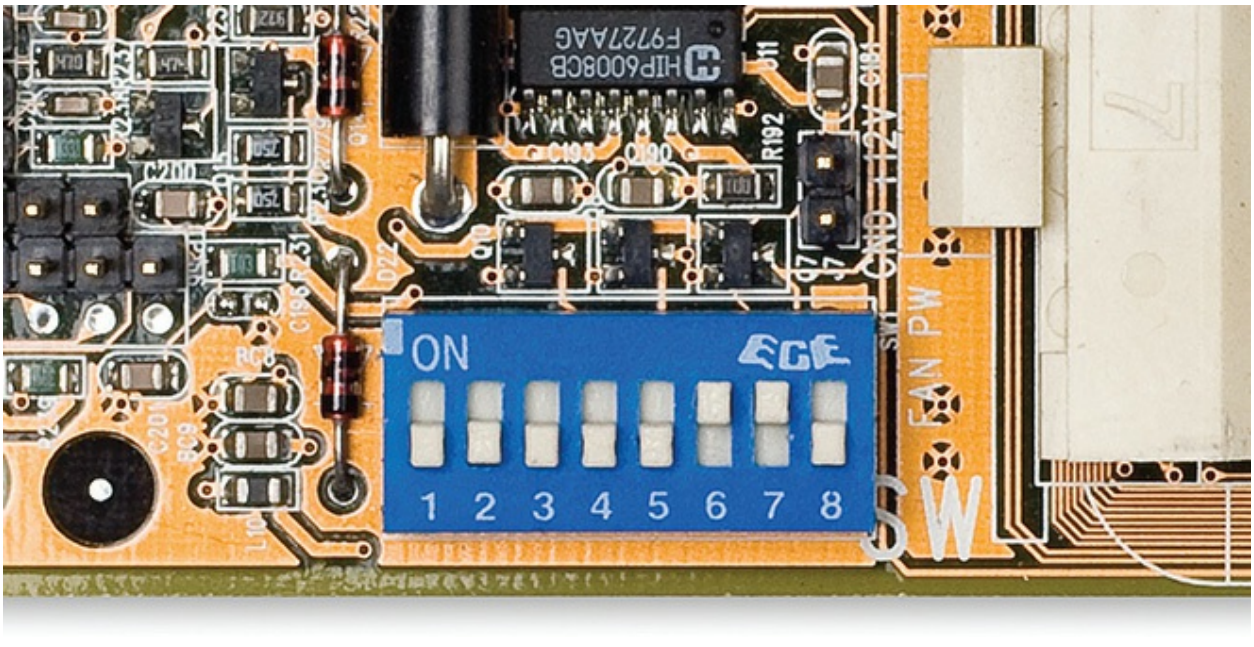

Figure 3-23 DIP switches on a motherboard

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 several 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 10, and 64-bit applications. They also support 32-bit processing for 32-bit operating systems, such as some Linux distributions, 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 16 GB of RAM with such programs.

EXAM TIP The primary benefit of 64-bit computing is to support more than 4 GB of memory, the limit with 32-bit processing.

x86 CPUs from the early days can be lumped together as *x*86 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.

x64 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 *x*64, and that created a mess.

x86-64 The earlier 32-bit stuff had been marketed as *x*86, not *x*32, so now we have *x*86 (old, 32-bit stuff) versus *x*64 (new, 64-bit stuff). It's not pretty, but do you get the difference? To make matters even worse, however, *x*64 processors quite happily handle *x*86 code and are, by definition, *x*86 processors too! It's common to marry the two terms and describe current 64-bit CPUs as *x*86-64 processors.

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 22), 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.

EXAM TIP The CompTIA A+ 1001 objectives refer to virtualization support as the *virtual technology* CPU feature.

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 send the answer back out onto 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 3-24)!

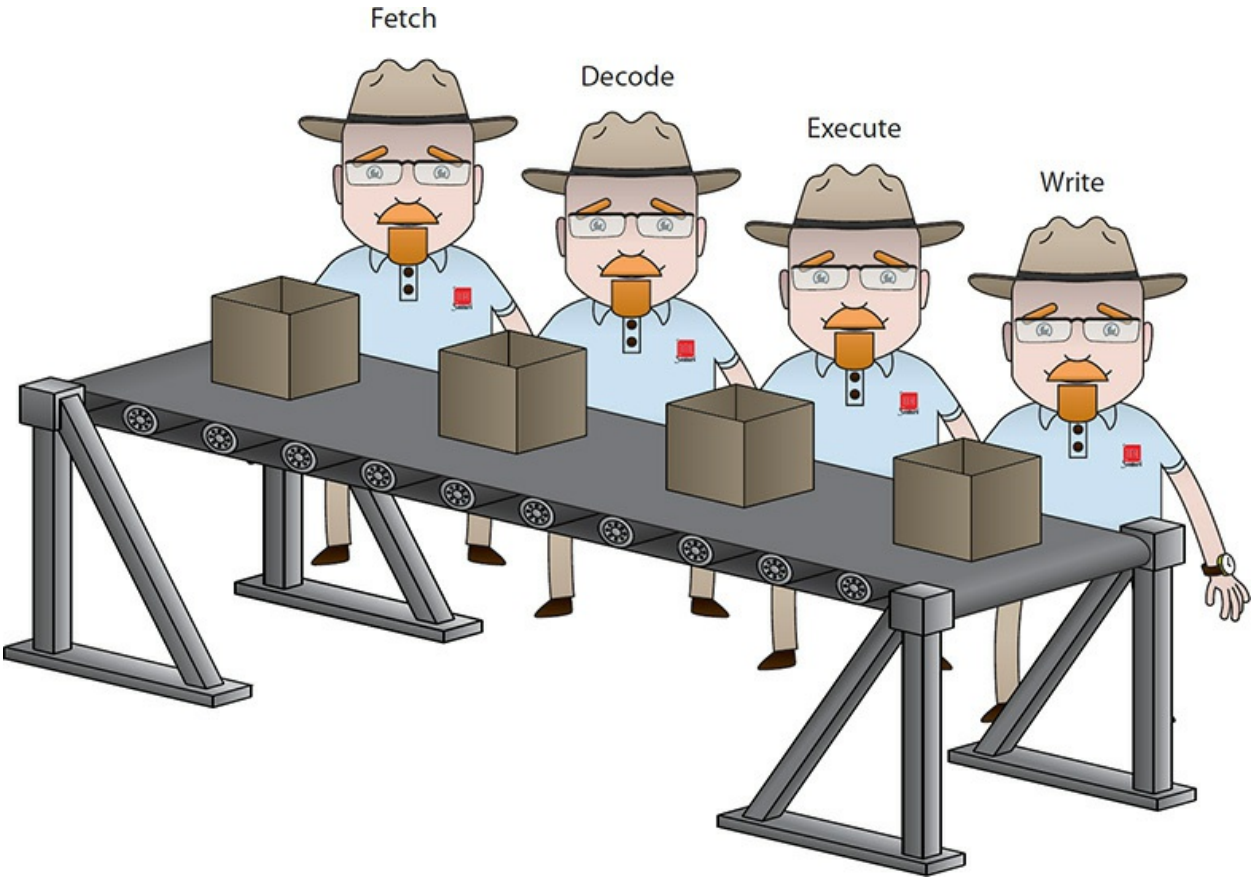

Figure 3-24 Simple pipeline

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, 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 3-25). Current CPUs offer multiple pipelines to keep the processing going (see Figure 3-26).

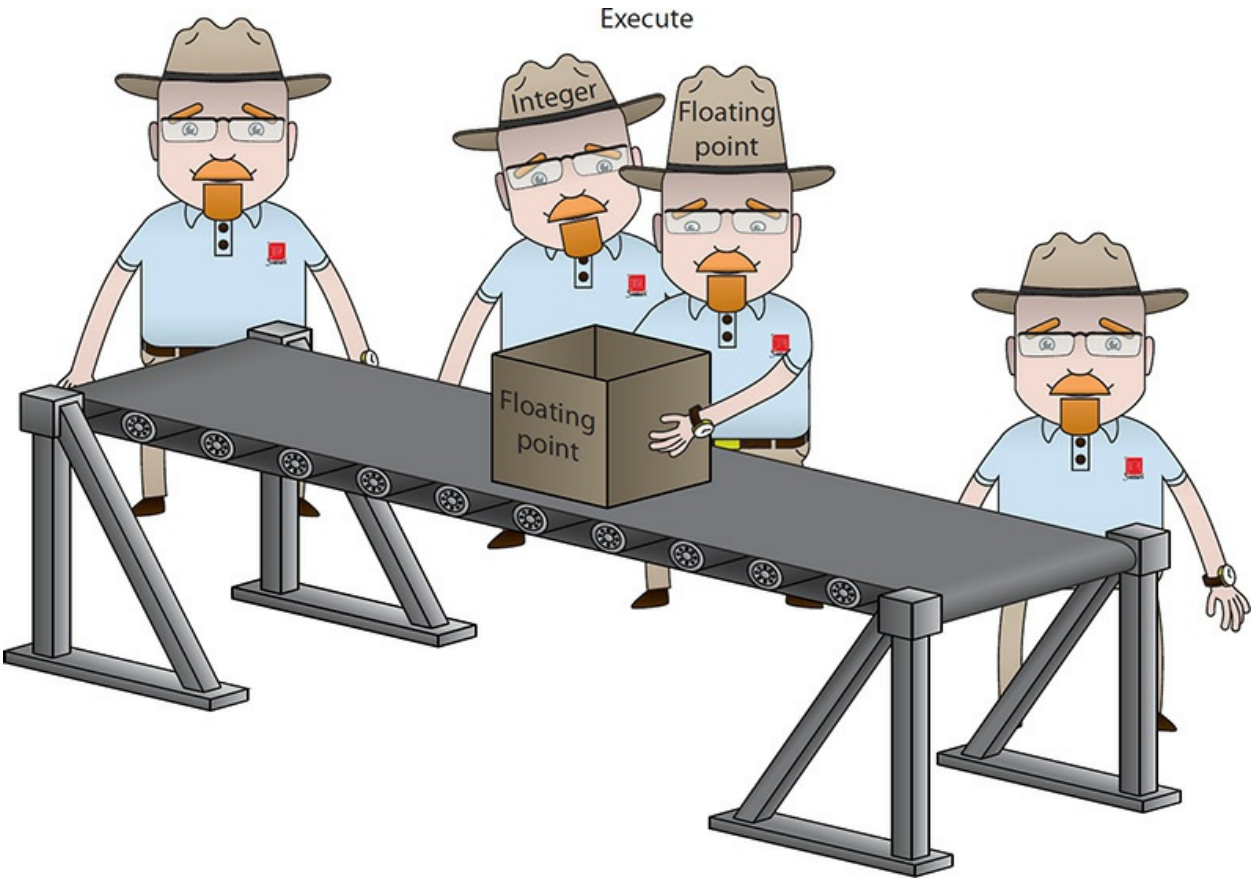

Figure 3-25 Bored integer unit

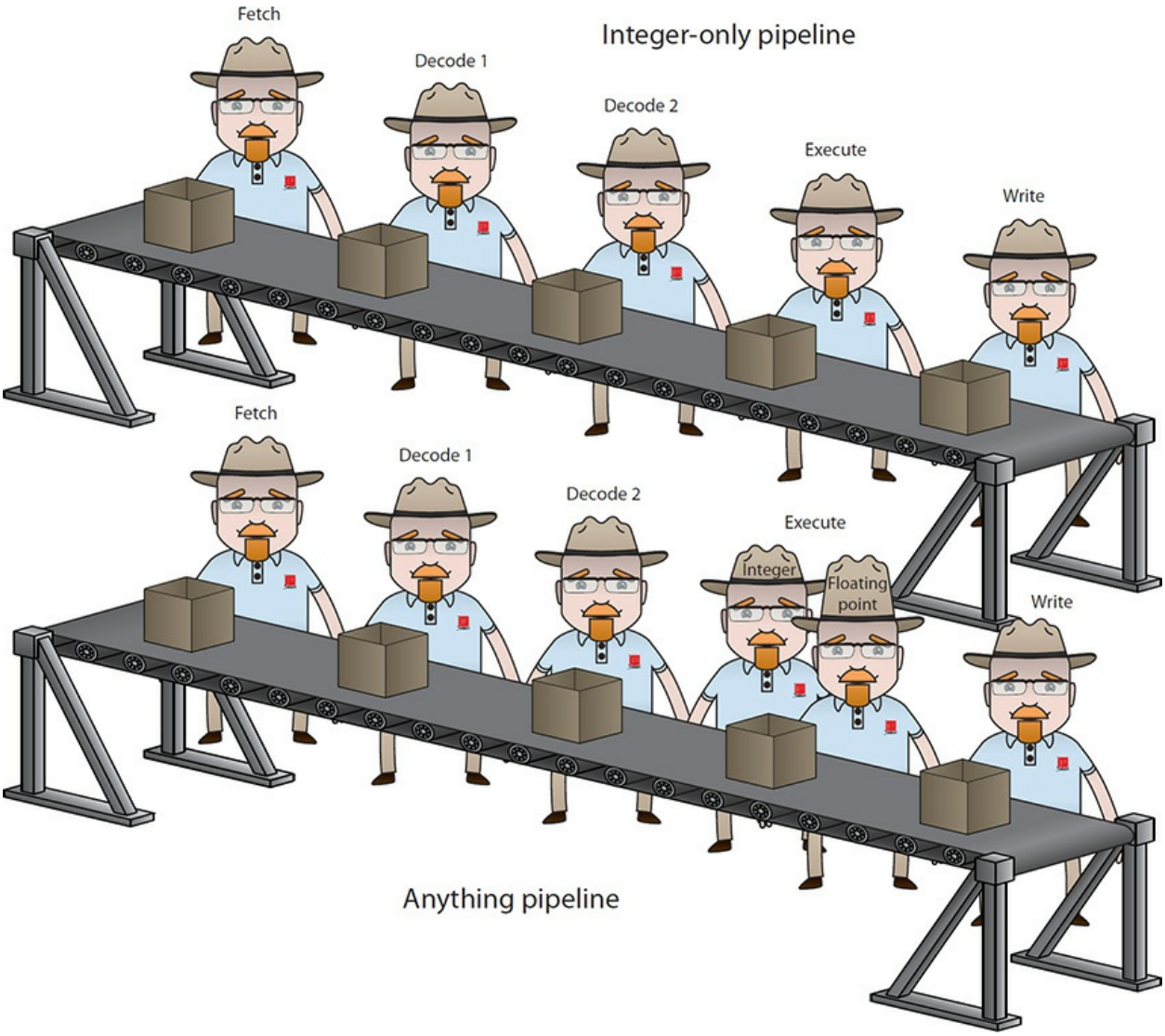

Figure 3-26 Multiple pipelines

Cache When you send a program to the CPU, you run lots of little programs all at the same time. Okay, let's be fair here: *you* didn't run all 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 3-27). SRAM used in this fashion is called a *cache*.

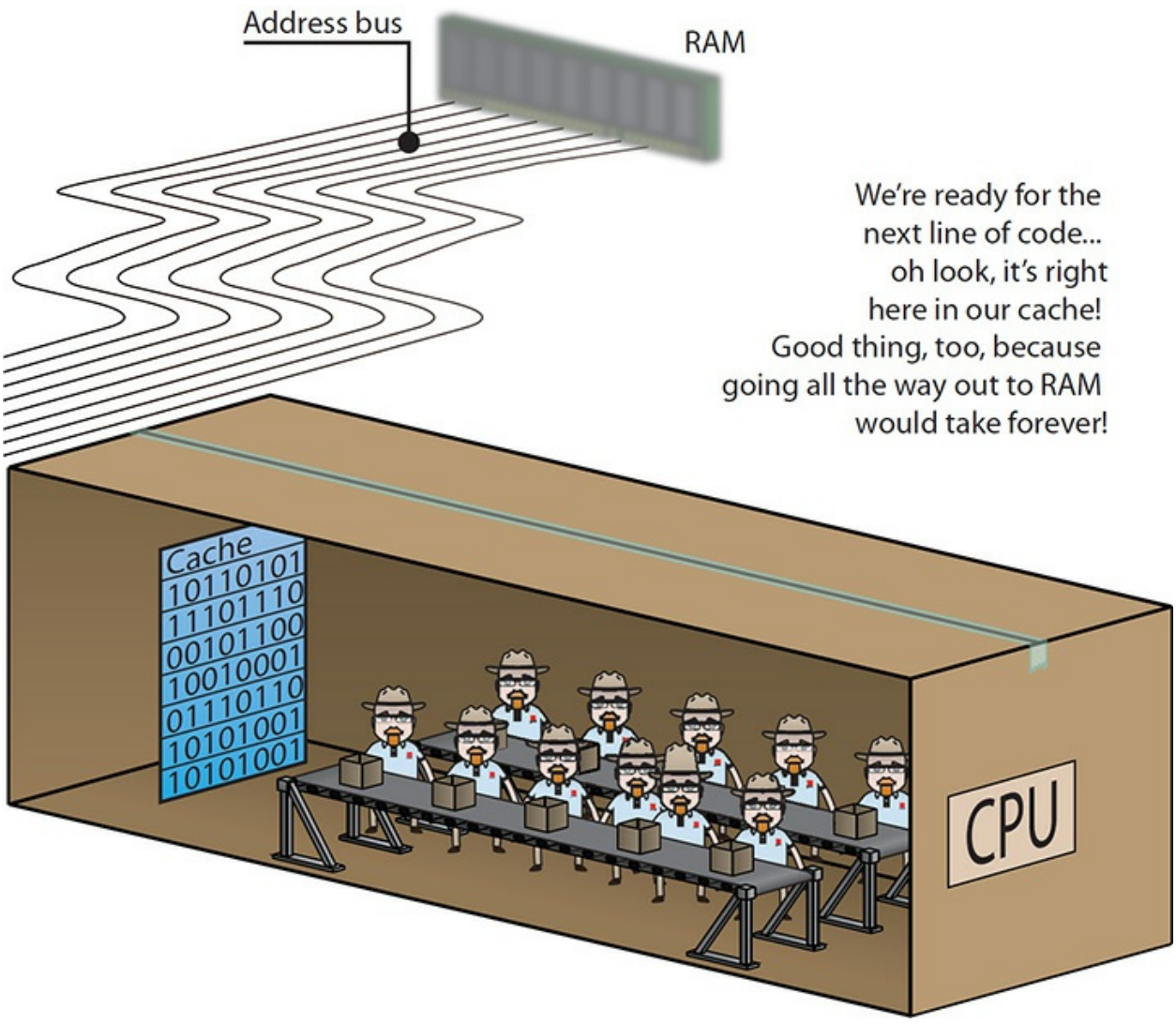

Figure 3-27 SRAM cache

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.

Eventually, engineers took this cache concept even further and added the L2 cache onto the CPU package. Many modern CPUs include three caches: an L1, an L2, and an *L3 cache* (see Figure 3-28).

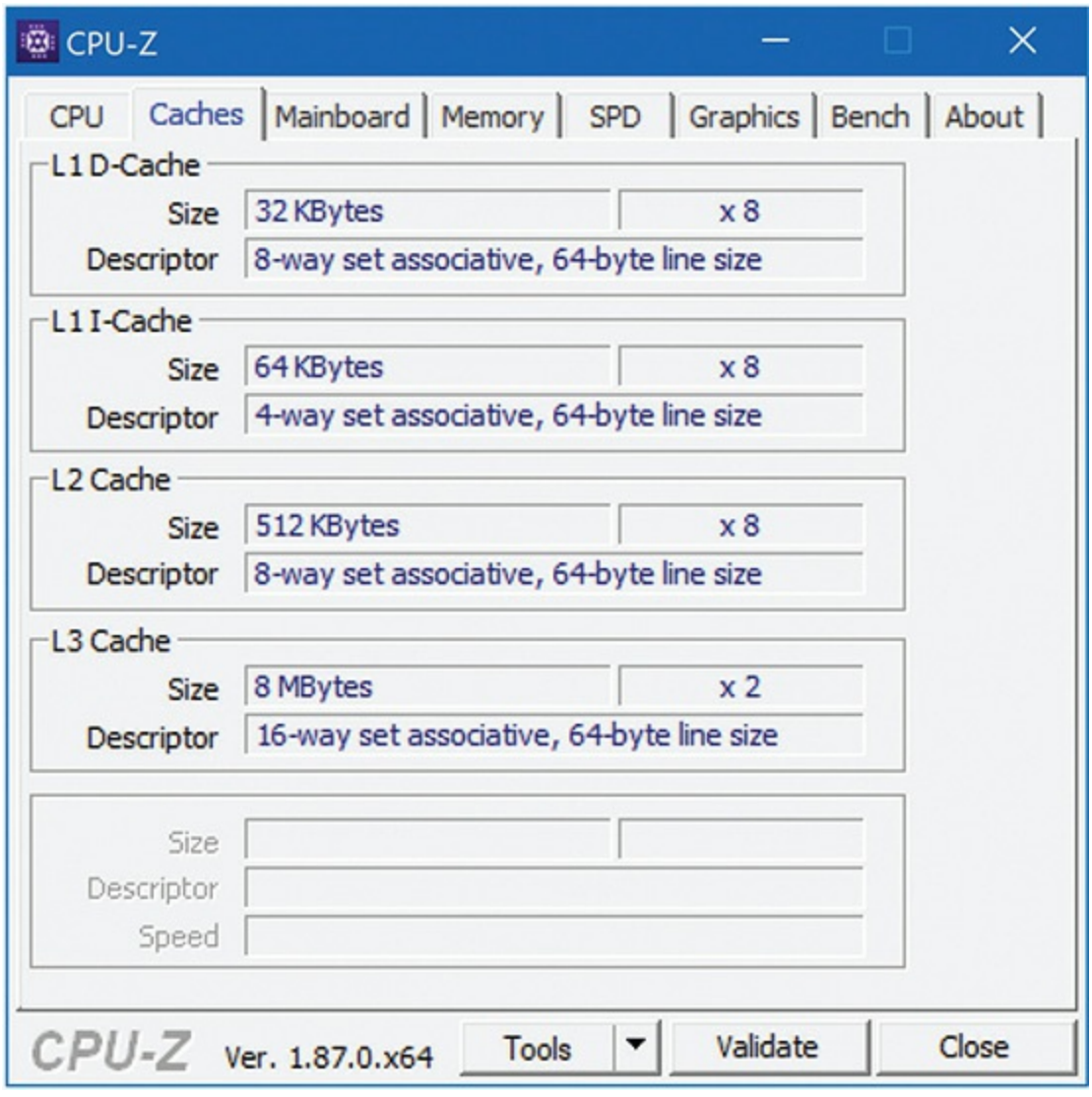

Figure 3-28 CPU-Z displaying the cache information for a Ryzen 7 processor

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 3-29). (These terms don't apply well to current computers, so they have fallen out of use. See the "Integrated Memory Controller" section later in this chapter.)

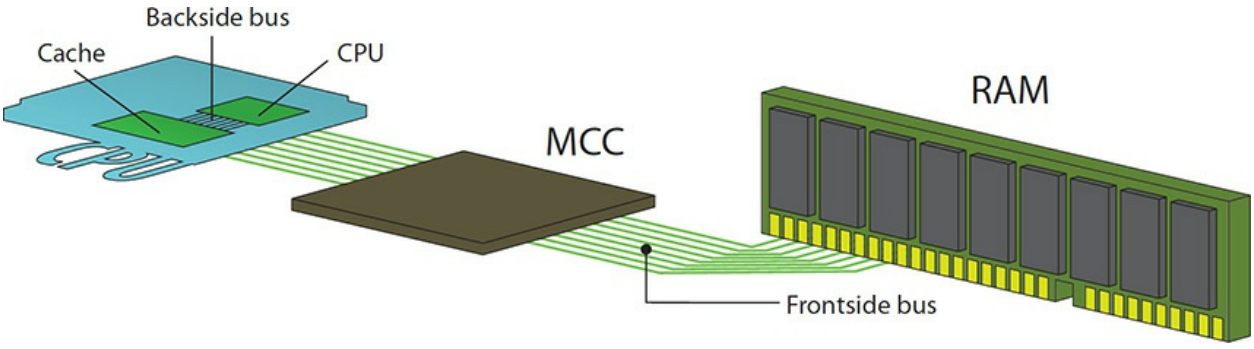

Figure 3-29 Frontside and backside buses

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

EXAM TIP 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. (This is not completely true anymore, with L1 and L2 running the same speed in many CPUs, but it is how it will appear on the exams.)

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 3-30 shows the Task Manager in an ancient Windows XP computer 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.

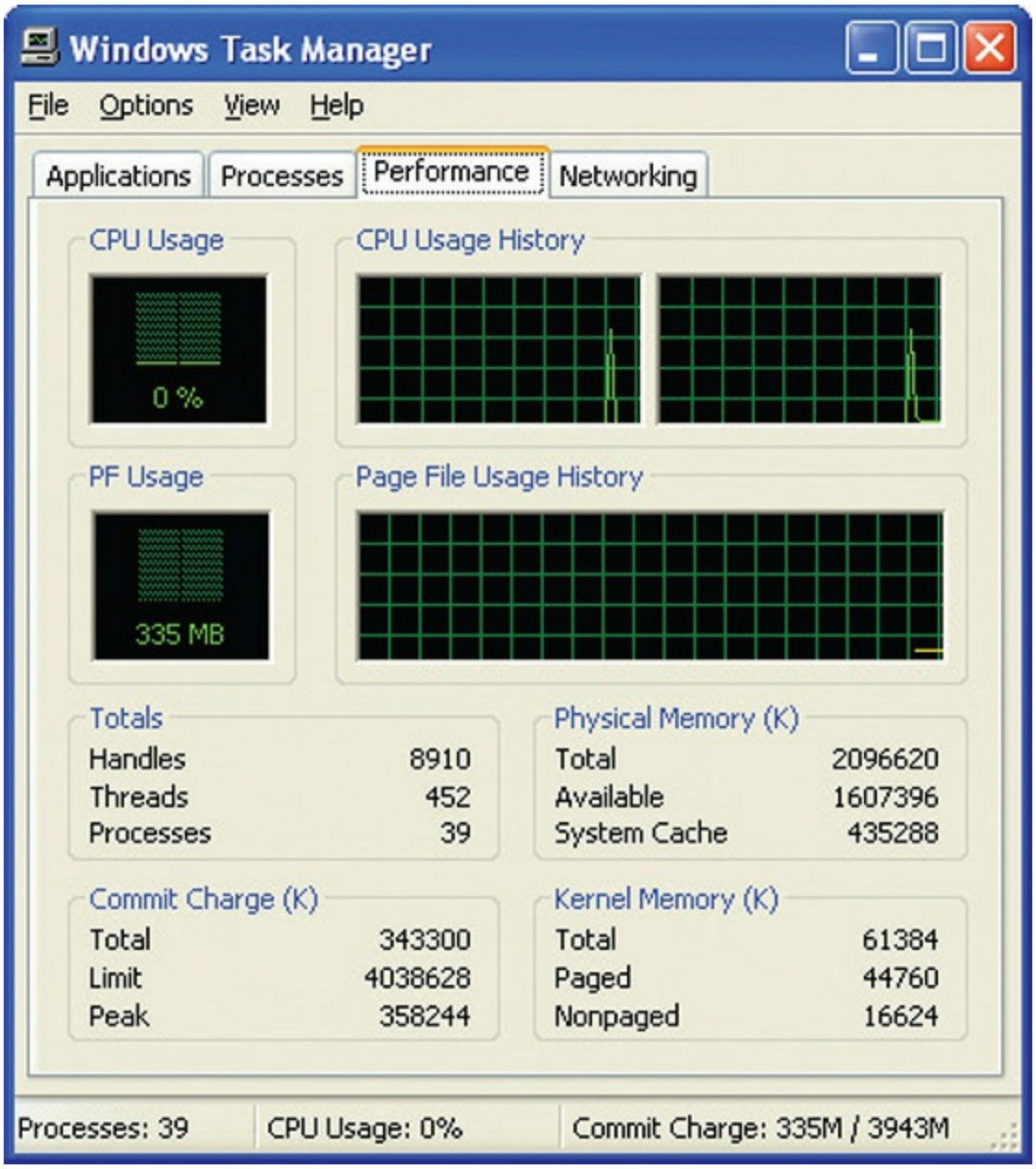

Figure 3-30 Windows Task Manager with the Performance tab displayed for a system running a Hyper-Threaded Pentium 4

Multithreading enhances a CPU's efficiency but with a couple of limitations. First, the operating system and the application must 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

Microarchitecture hit a plateau back in 2002 when CPU clock speeds hit a practical limit of roughly 4 GHz, 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 move beyond the *single-core* CPU and 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. A single-core CPU has only one set of everything.

Today, multicore CPUs—with four, six, or eight cores—are common. Higher-end CPUs have up to 32 cores! 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 3-31 shows another screenshot of CPU-Z, this time displaying the cache breakdown of a Haswell-based Core i7.

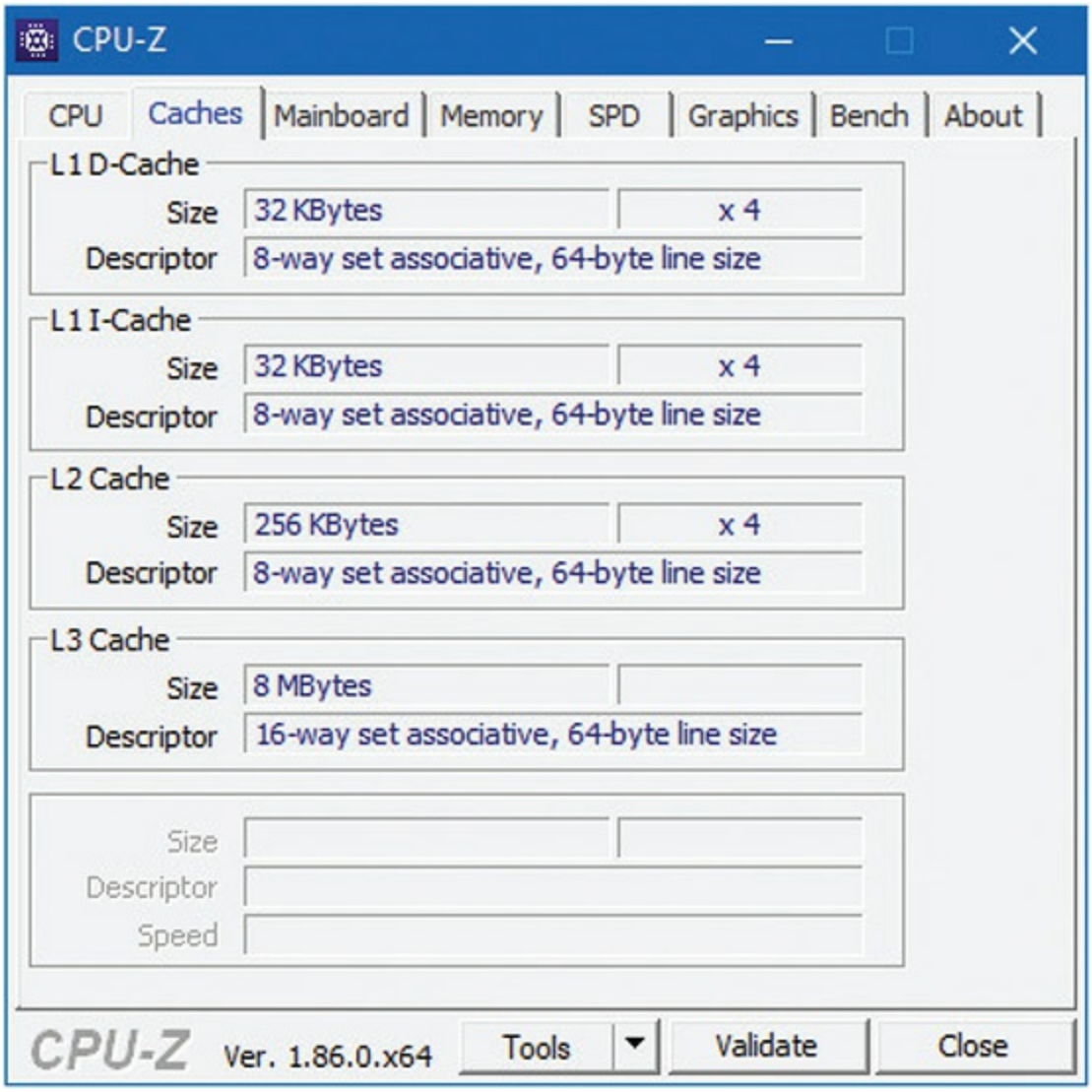

Figure 3-31 CPU-Z showing the cache details of a Haswell Core i7

Figure 3-31 reveals specific details about how this Intel CPU works with cache. The Core i7 has L1, L2, and L3 caches of 32 KB, 256 KB, and 8 MB, respectively. (The L1 cache divides into 32 KB to handle data—the *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 six 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 must be written specifically to handle the multiple threads. Note that even with multicore processors, applications must be modified or optimized for this parallelism to have a huge impact on performance.

Because one great technology advancement isn't enough, both Intel and AMD make multicore CPUs that incorporate Hyper-Threading as well. The Intel Core i9-7960X, for example, sports 16 cores, Hyper-Threading, 16 MB of L2 cache and 22 MB of L3 cache, and Turbo Boost to crank the clock speed over 4 GHz when the system needs it. I get shivers just thinking about it!

SIM This is a great time to head over to the Chapter 3 Show! and Click! sims to see how to download and use the CPU-Z utility. Check out "What is CPU-Z?" here: http://totalsem.com/100x.

Integrated Memory Controller

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 4. For now, add "different RAM support" to your list of things to look at when making a CPU recommendation for a client.

Integrated Graphics Processing Unit

As you'll read about in much more detail in Chapter 17, "Display Technologies," 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 17, 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 these benefits have obvious merit.

Both Intel and AMD produce CPUs with integrated GPUs. For many years, the quality of the GPU performance with demanding graphical programs like games made the choice between the two easy. The *Intel HD Graphics* and *Intel Iris Pro Graphics* integrated into many Core i3/i5/i7 processors pale in comparison with the AMD *accelerated processing unit* (*APU*), such as the AMD A10. AMD bought one of the two dedicated GPU manufacturers—ATI—years ago and used their technology for microprocessors with integrated CPU and GPU. (The Xbox One and PlayStation 4 gaming systems, for example, use AMD APUs.) Intel is slowly closing the gap.

Security

All modern processors employ the *NX bit* technology that enables the CPU to protect certain sections of memory. This feature, coupled with implementation by the operating system, stops malicious attacks from getting to essential operating system files. Microsoft calls the feature Data Execution Prevention (DEP), turned on by default in every OS (see Figure 3-32).

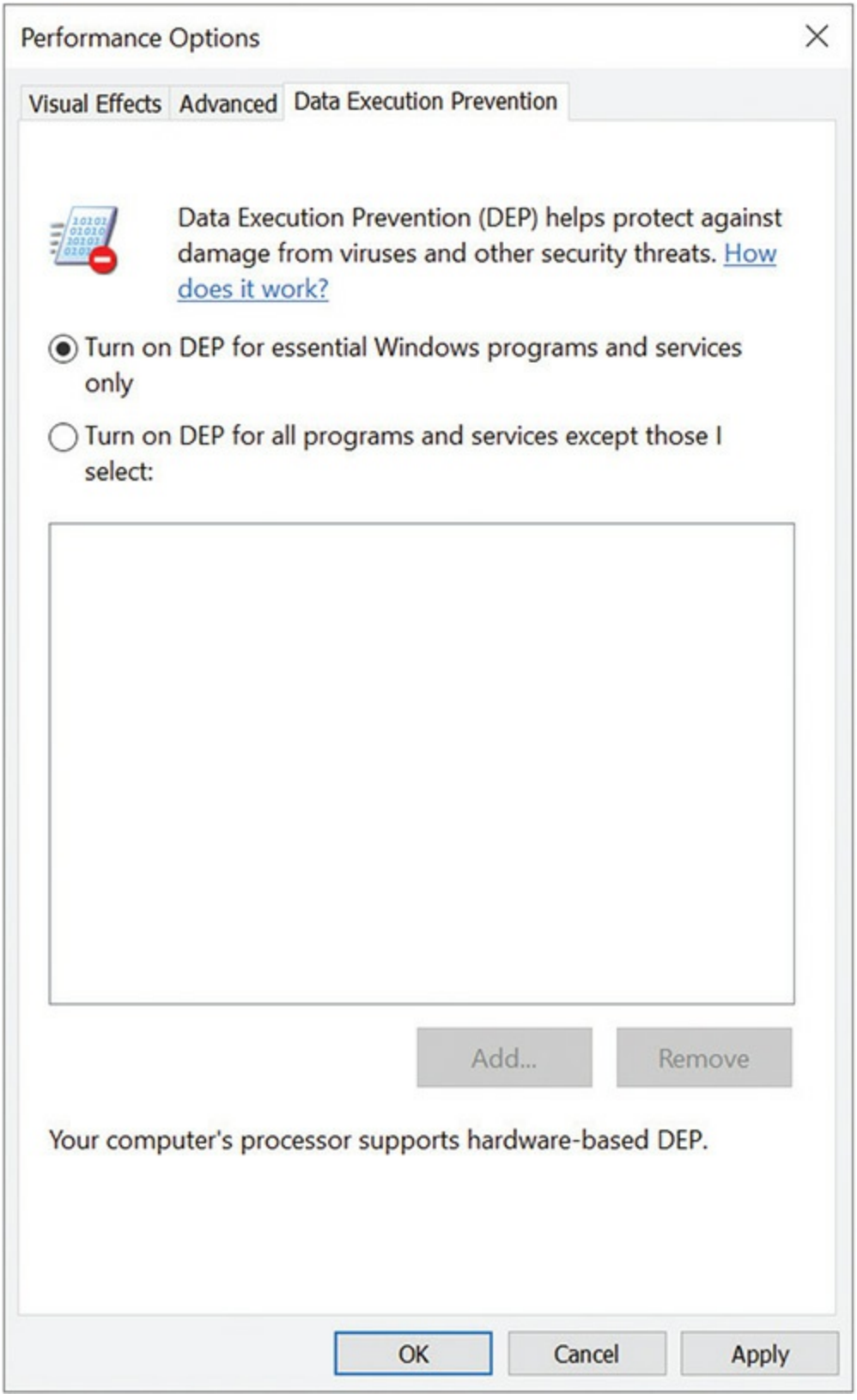

Figure 3-32 DEP in Windows 10

Everybody calls the NX bit technology something different (but you don't need to memorize any of this for the exams):

- **Intel** XD bit (eXecute Disable)
- **AMD** Enhanced Virus Protection
- **ARM** XN (eXecute Never)

Selecting and Installing 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 11, "Building a PC," 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 specific processors. 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 CPUs 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 3-33 shows a manual for an ASUS motherboard open to reveal the supported processors and the socket type.

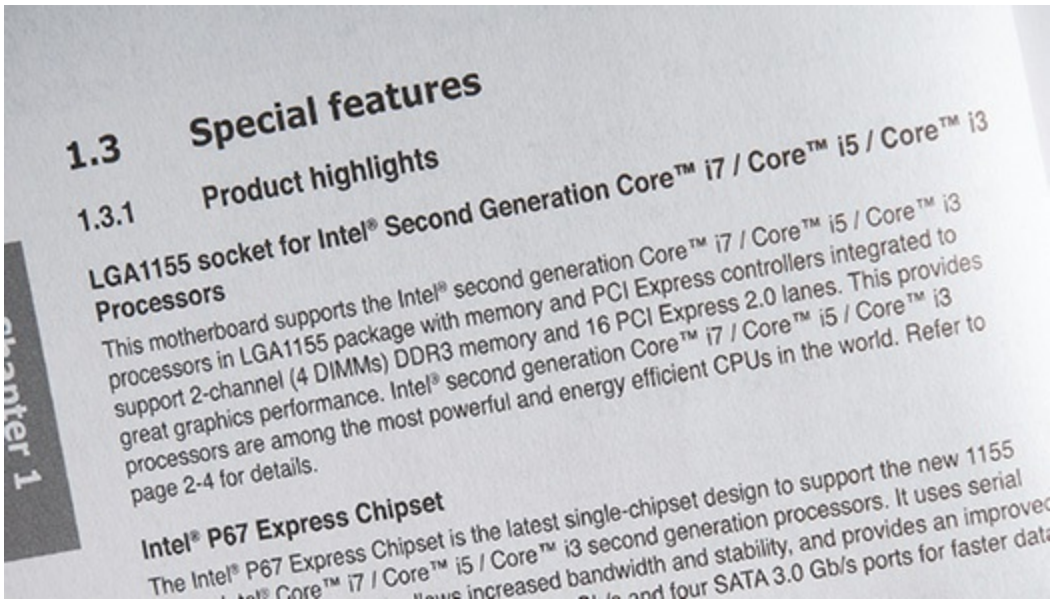

Figure 3-33 Supported processors and socket type

Just as Intel and AMD make many types of CPUs, motherboards are manufactured with various different types of sockets. There have been hundreds of sockets developed over the years. Table 3-2 charts a few of the more popular ones in production today. CompTIA has traditionally tested on both the socket number/name and the alternative name, in the case of Intel sockets. I've included the most recent socket names and alternative names here.

Socket	Alternative Name	Platform	CPU
LGA 1150	H3	Intel	Core i3/i5/i7, Pentium, Celeron, Xeon
LGA 1151	H4	Intel	Core i3/i5/i7, Pentium, Celeron, Xeon
LGA 2011	R or R3	Intel	Core i7, Core i7 Extreme Edition, Xeon
LGA 2066	R4	Intel	Core i5/i7/i9, Xeon
FM2+	n/a	AMD	A-Series
AM3+	n/a	AMD	FX, Opteron
AM4	n/a	AMD	Ryzen, A-Series
TR4	n/a	AMD	Ryzen Threadripper

Table 3-2 Common Sockets

EXAM TIP The CompTIA A+ 1001 objectives do not list any specific processor sockets, but previous versions frequently included questions about them. Hopefully you won't run into one of these questions, but it's not a bad idea to know the sockets just in case. Beyond the exam, just make sure that you understand that every CPU has a specific socket into which it fits and make sure a client's motherboard has the socket that works with a suggested CPU.

Deciphering Processor Numbers

Intel and AMD use different processor numbering schemes that help you compare multiple CPUs with similar names, such as Core i5. AMD and Intel both have fairly similar numbering schemes. Here's the scoop on both.

Intel processor numbers follow a very clear pattern. An Intel Core i7 7500 U processor, for example, maps out like this:

- Intel Core = brand
- i7 = brand modifier
- 7 = generation
- 500 = SKU numbers
- U = alpha suffix (U indicates that it's a desktop processor using ultra-low power)

Contrast the previous processor with an Intel Core i7 8650 U, where the numbers map like this:

- Intel Core = brand
- i7 = brand modifier
- 8 = generation
- 650 = SKU numbers
- U = alpha suffix (U indicates that it's a desktop processor using ultra-low power)

AMD processor nomenclature is similar. Here's the breakdown for an AMD Ryzen 7 2700X:

- AMD Ryzen = brand
- 7 = market segment
- 2 = generation
- 7 = performance level
- 00 = model number
- X = power suffix (X indicates high-performance)

Try This! Processor Research

Both Intel and AMD maintain accessible Web sites with exhaustive information about their recent CPUs. All three Web sites listed here provide details you can use for client support and for recommendations when dealing with specific sockets for upgrades. Try this! Put one or more of the following links into a Web browser and explore the CPUs. Then tuck the URLs into your tech toolkit for later reference when needed.

- https://ark.intel.com
- https://www.amd.com/en/products/processors-desktop
- https://www.amd.com/en/products/processors-lapto

Installation Issues

When installing a CPU, you need to use caution with the tiny pins on the CPU or the socket. 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. Third, you must 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 Intel-

based motherboards, the sockets have hundreds of tiny pins that line up with contacts on the bottom of the CPU (see Figure 3-34). Intel CPUs use a *land grid array* (*LGA*) package for socketed CPUs, where the underside of the CPU has hundreds of contact points that line up with the socket pins.

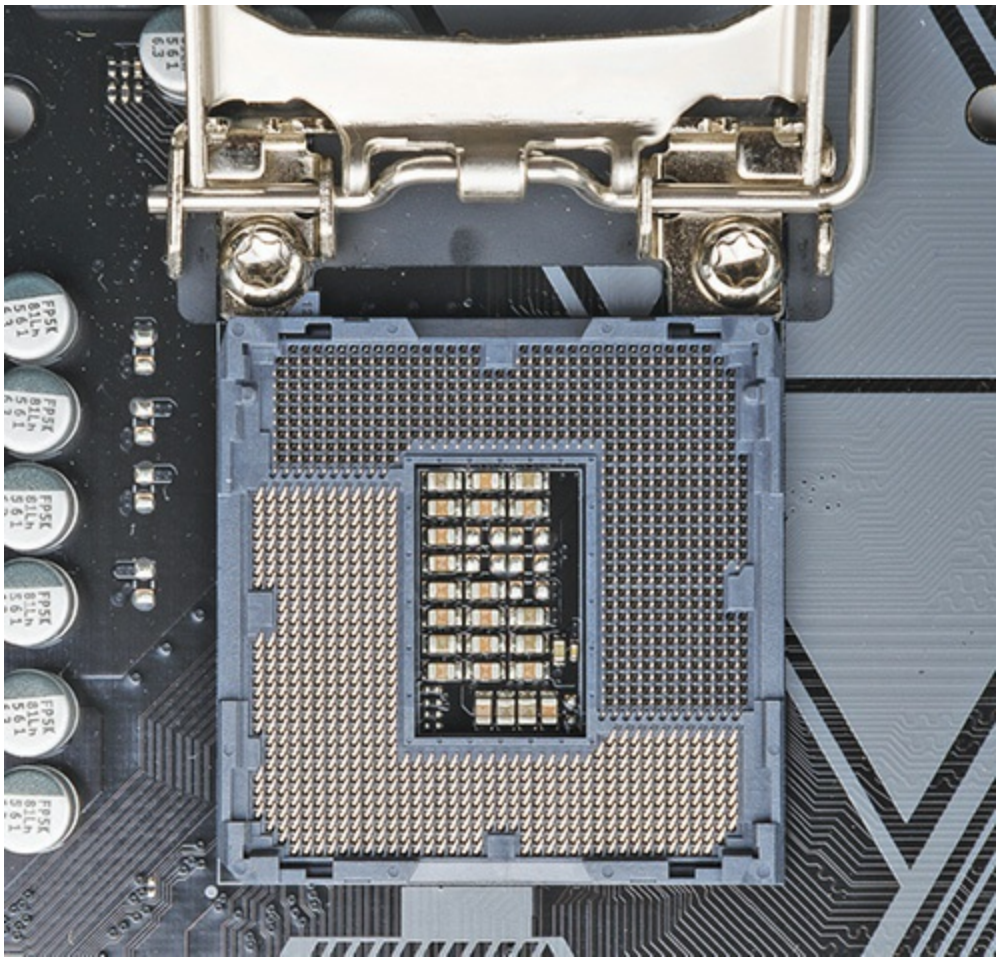

Figure 3-34 Intel-based socket with pins

AMD CPUs have the pins (see Figure 3-35); the sockets have holes. The pins on the AMD *pin grid array* (*PGA*) CPUs align with the holes in the sockets.

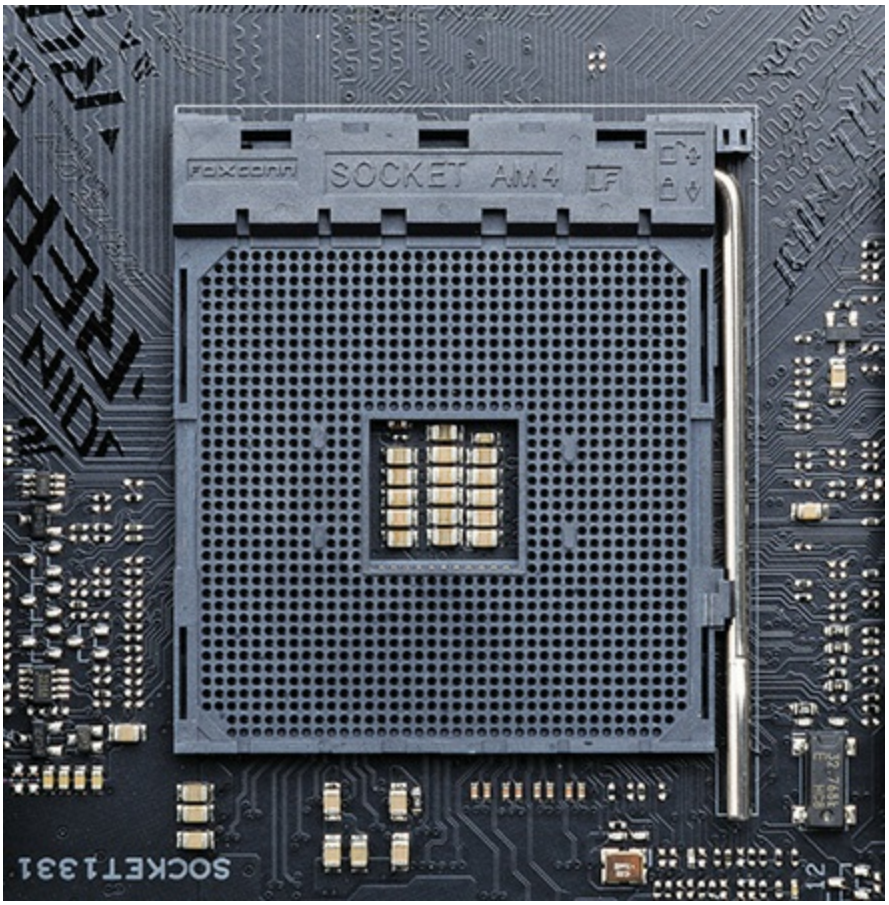

Figure 3-35 AMD-based socket without pins

All CPUs and sockets are keyed so you can't (easily) insert them incorrectly. Look at the underside of the CPU on the left side of Figure 3-36. Note that the pins do not make a perfect square, because a few are missing. Now look at the top of the CPU on the right in Figure 3-36. See the little mark at the corner? The socket also has tiny markings so you can line the CPU up properly with the socket.

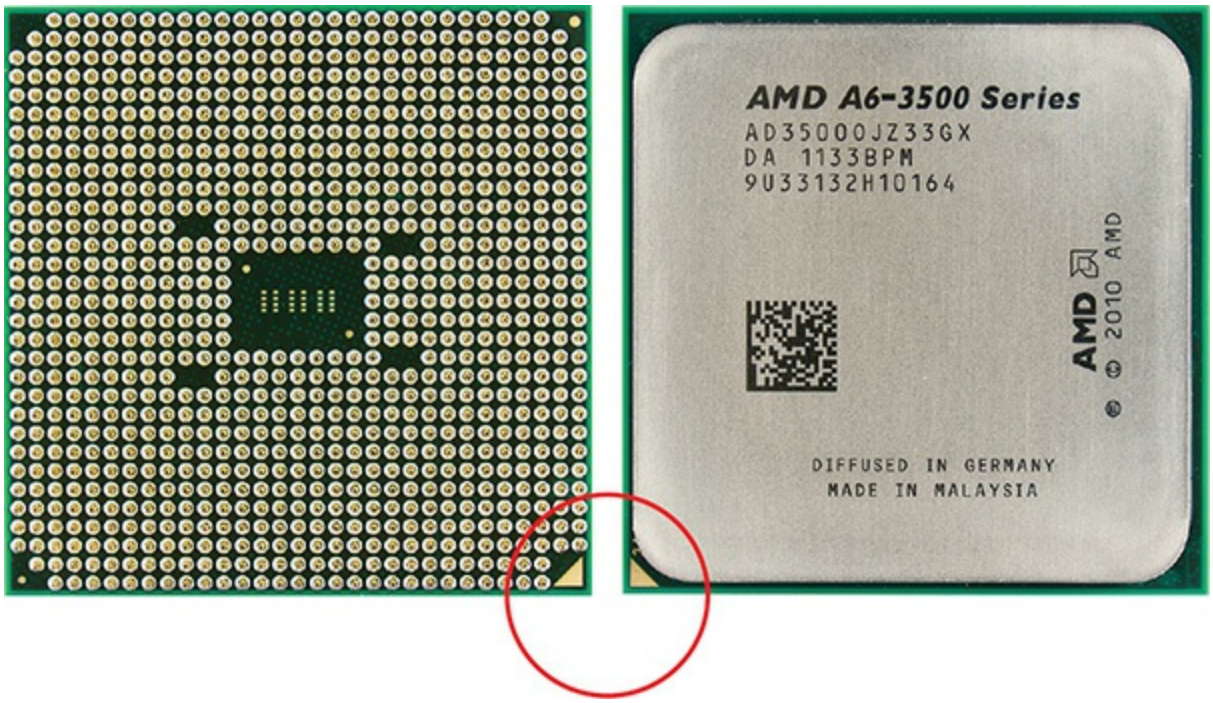

Figure 3-36 Underside and top of a CPU

In both socket styles, you release the retaining mechanism by pushing the little lever down slightly and then away from the socket (see Figure 3-37). You next raise the arm fully, and then move the retaining bracket (see Figure 3-38).

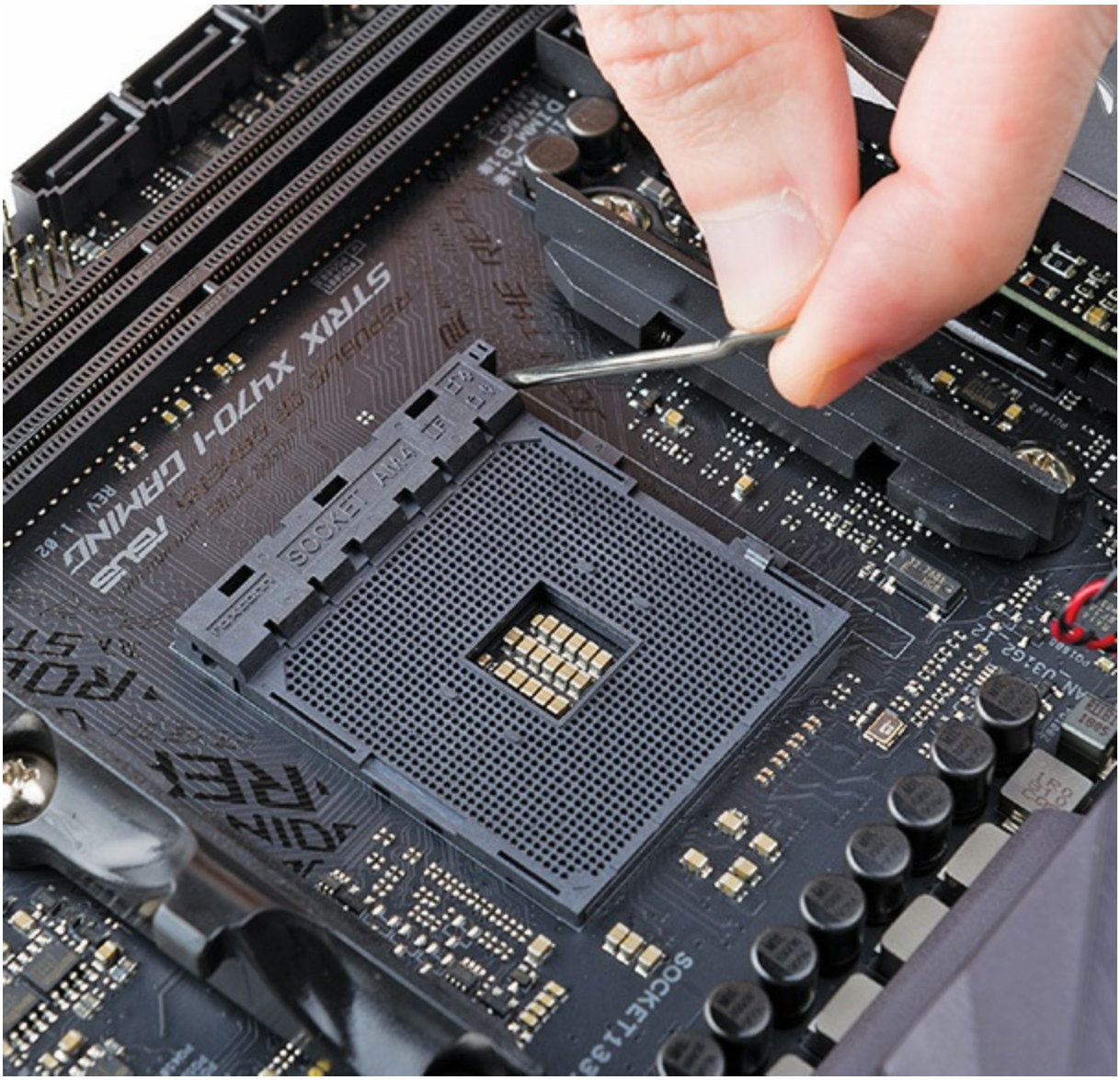

Figure 3-37 Moving the release arm

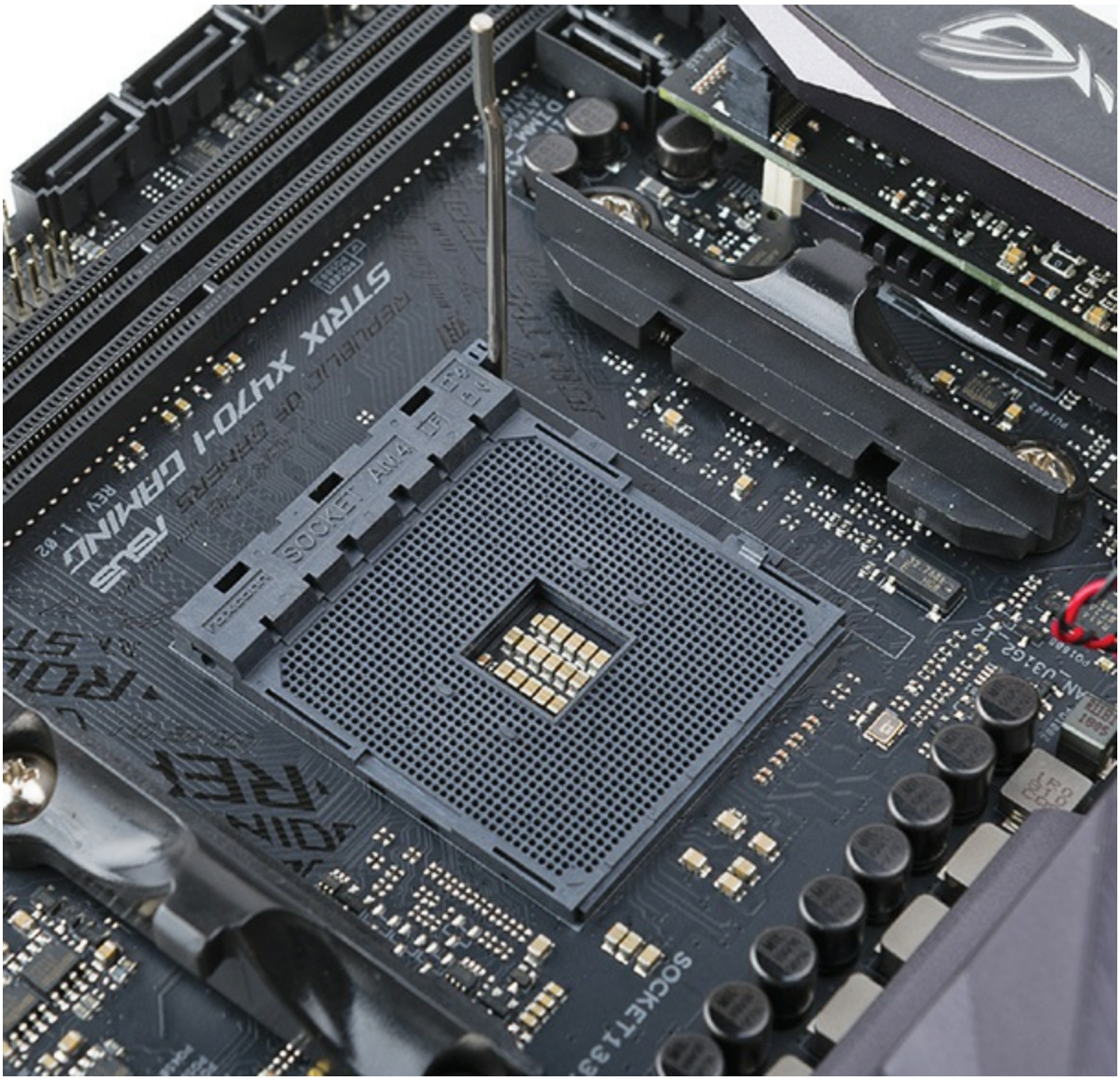

Figure 3-38 Fully opened socket

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.

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 7, "Power Supplies," 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.

EXAM TIP A heat sink by itself (no fan) on a chip provides *passive cooling*. A heat sink and fan combination provides *active cooling*. You'll sometimes hear the latter described as an *active heat sink*.

A *heat sink* is a copper or other metal device designed to dissipate heat from whatever it touches. Figure 3-39 shows the standard Intel heat sink and fan. Here are some cooling options:

Figure 3-39 Intel stock heat-sink and fan assembly

- **OEM CPU coolers** Original equipment manufacturer (OEM) heat-sink and fan assemblies are included with most Intel and AMD retail-boxed CPUs. OEM in this case means that Intel makes the heat-sink/fan assemblies. Rather confusingly, you'll see the term "OEM CPUs" used to mean CPUs you buy in bulk or not in the retail packaging. These are still made by Intel or AMD and are functionally identical to the retail versions. They don't 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 (see Figure 3-40)—some are even lighted.

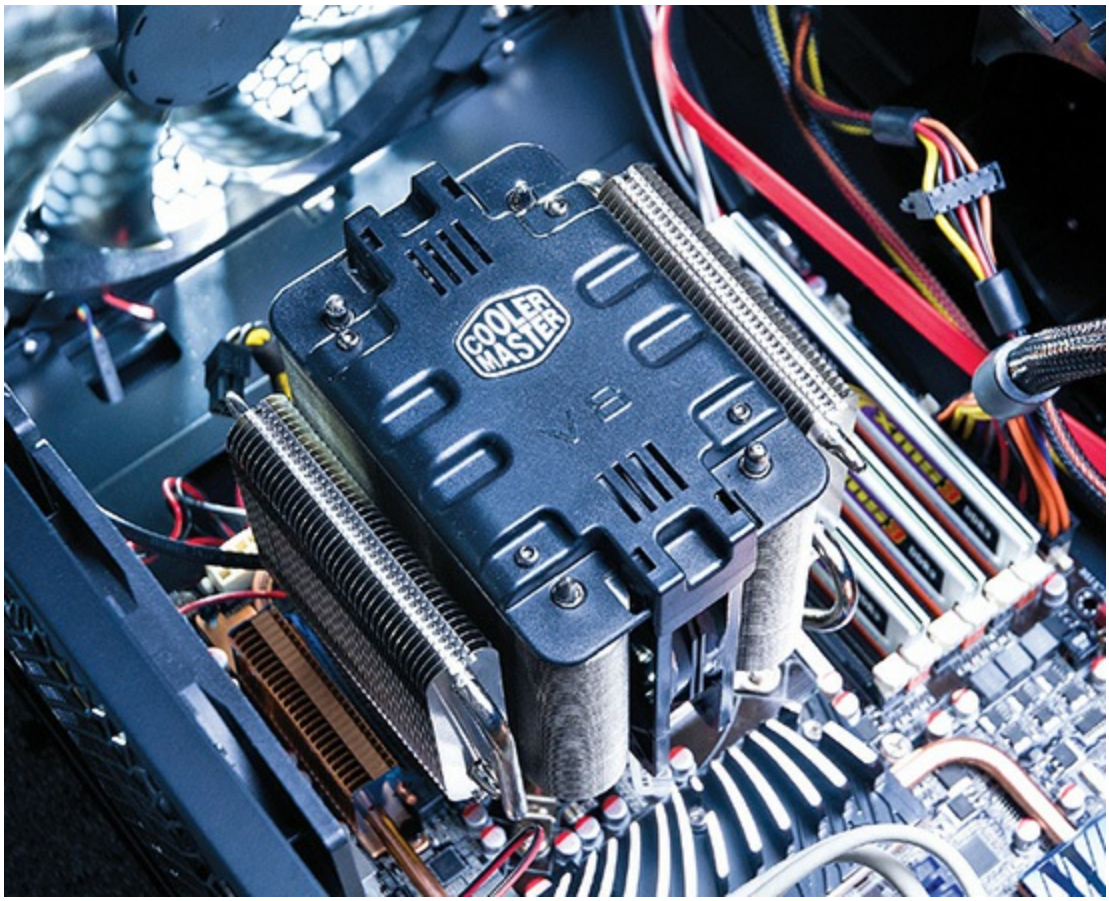

Figure 3-40 Cool retail heat sink

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

And of course, you need plenty of hosing to hook them all together. Figure 3-41 shows a typical liquid-cooled CPU.

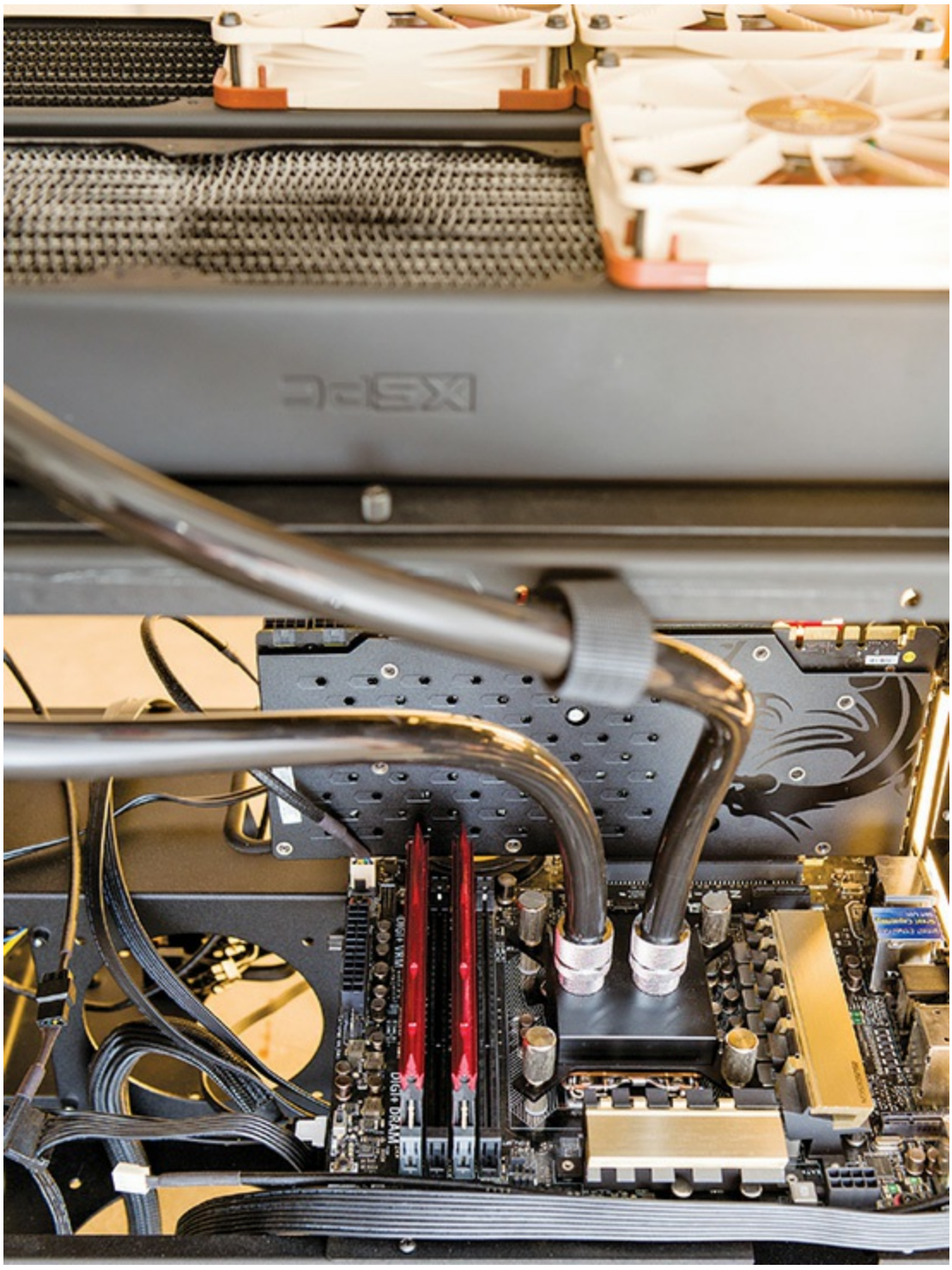

Figure 3-41 Liquid-cooled CPU

Several companies sell liquid-based cooling systems. Although they look impressive and certainly cool your CPU, unless you're overclocking or want

a silent system, a good fan will more than suffice.

EXAM TIP In some instances, you can create a system that has no fan for the CPU, what's called *fanless cooling*. Aside from mobile devices (like an Apple iPad) that have no fans, the term can be very misleading. The Xeon CPUs powering the servers in my office, for example, only have heat sinks with no fans. On the other hand, they have ducts directly to the case fans, which serve the same function as an active CPU fan. So, go figure.

See also the "Beyond A+" section at the end of this chapter for interesting passive cooling developments.

Once you have a heat-sink and fan assembly sorted out, you need to connect it 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 3-42). 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.)

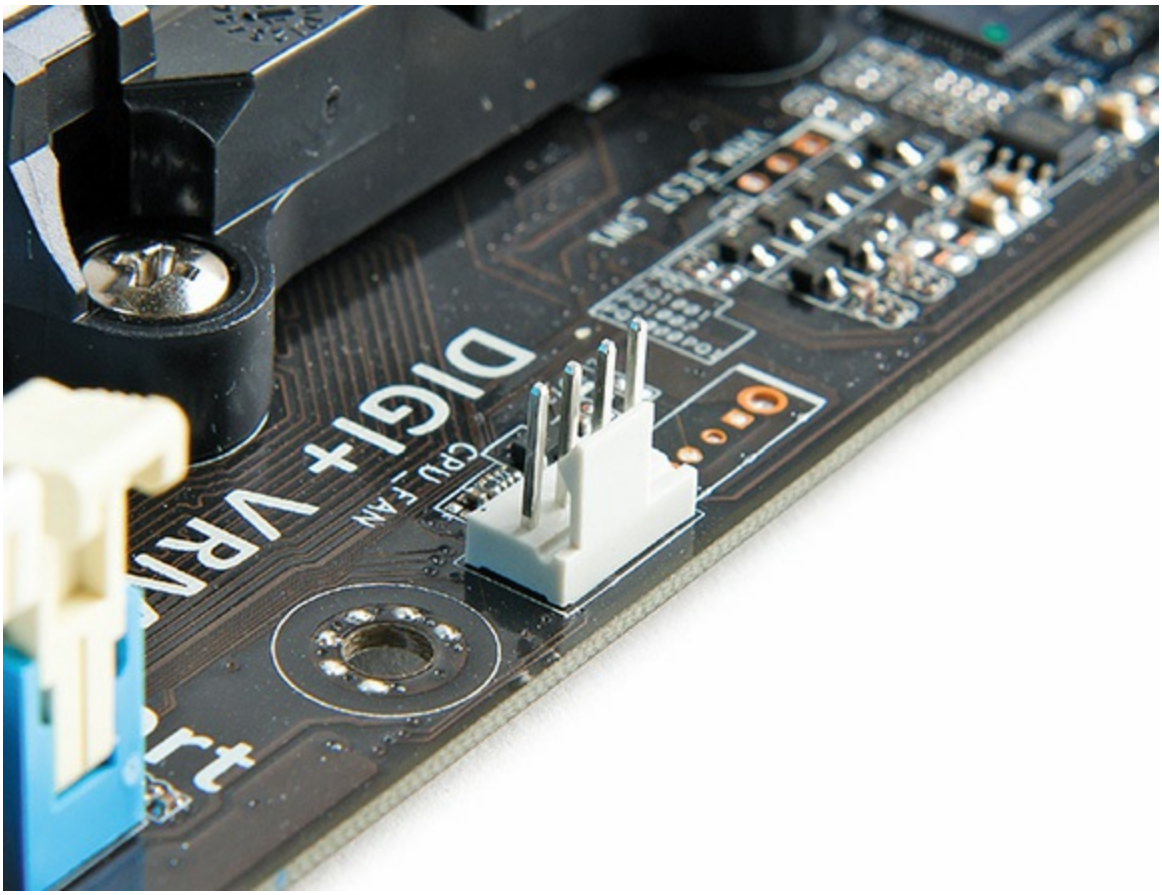

Figure 3-42 CPU fan power standout on motherboard

Next, before inserting the heat sink, you need to add a small amount of *thermal paste* (also called *thermal compound*, *heat dope*, or *nasty silver goo*). Many heat sinks come with some thermal paste already on them; the thermal paste 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 thermal paste on from a tube, know that you need to use only a tiny amount of this compound (see Figure 3-43). Spread it on as thinly, completely, and evenly as you can. Unlike so many other things in life, you *can* have too much thermal paste!

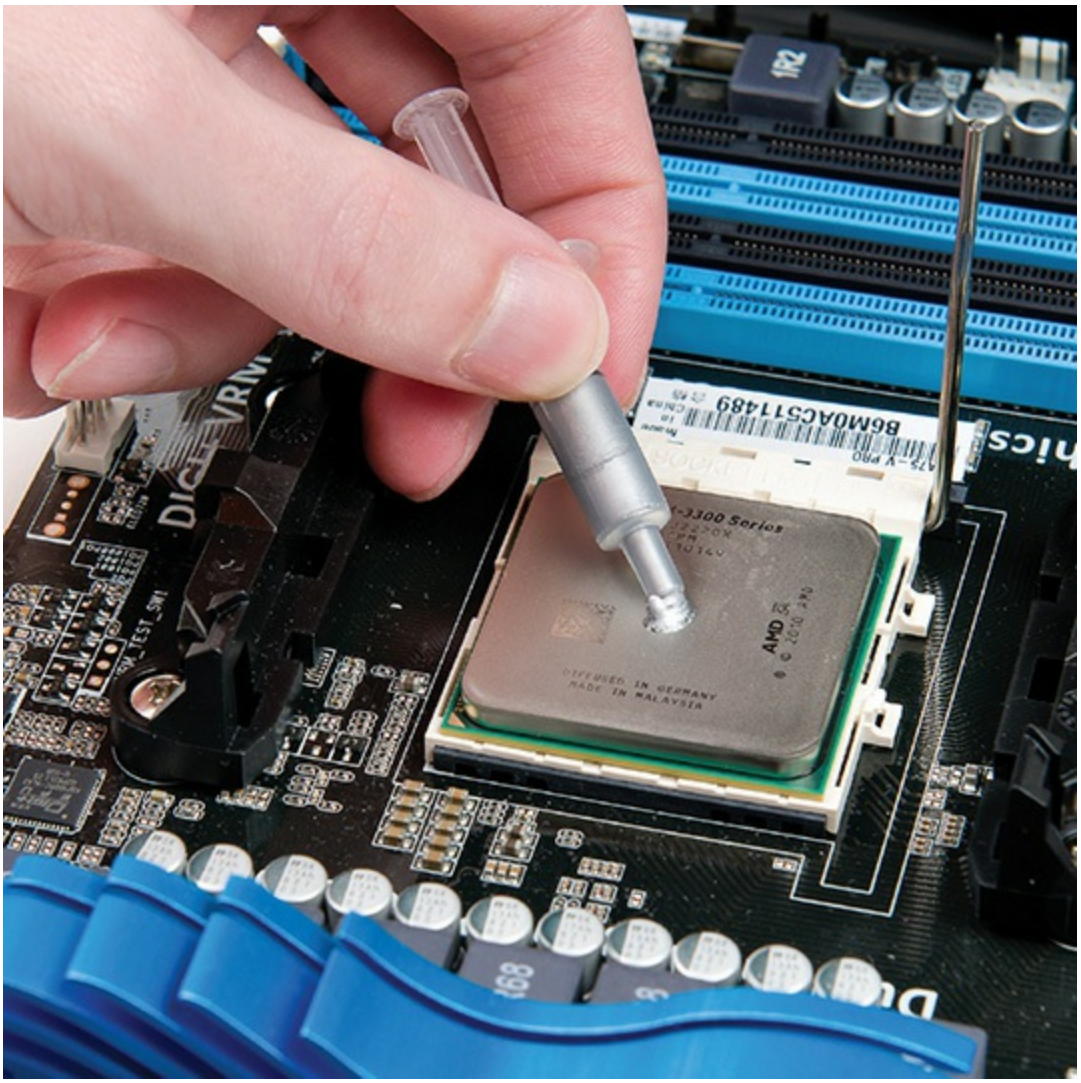

Figure 3-43 Applying thermal paste

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 3-44).

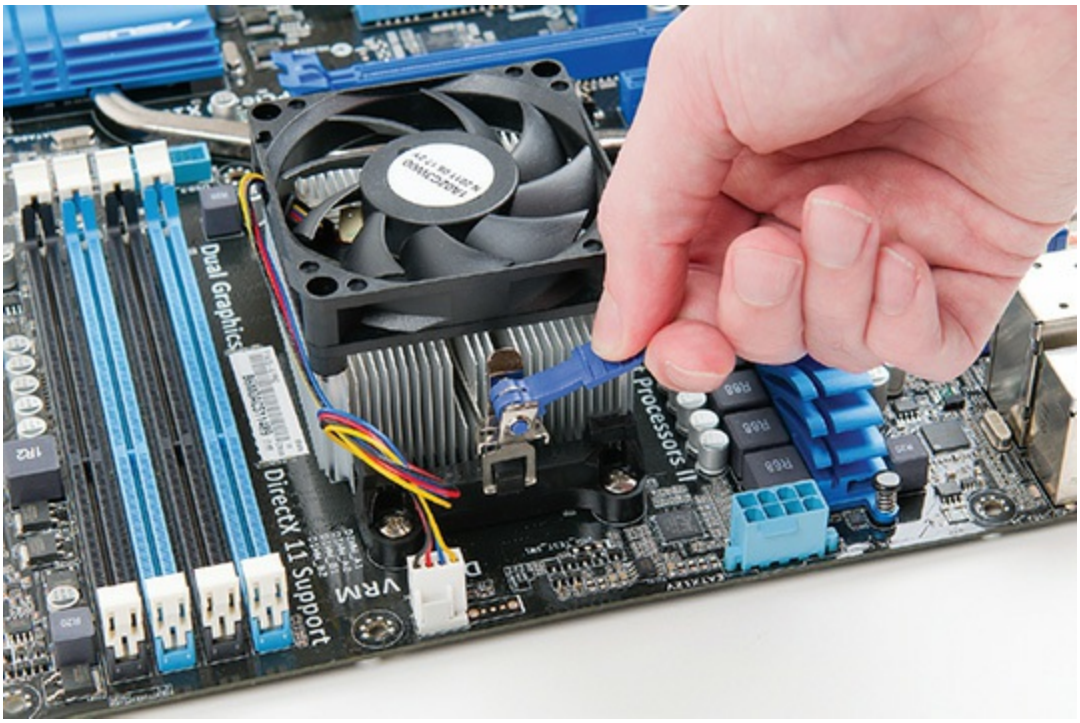

Figure 3-44 AMD stock heat-sink and fan assembly

Finally, you can secure many aftermarket heat-sink and fan assemblies by screwing them down from the underside of the motherboard (see Figure 3-45). You have to remove the motherboard from the case or install the heat sink before you put the motherboard in the case.

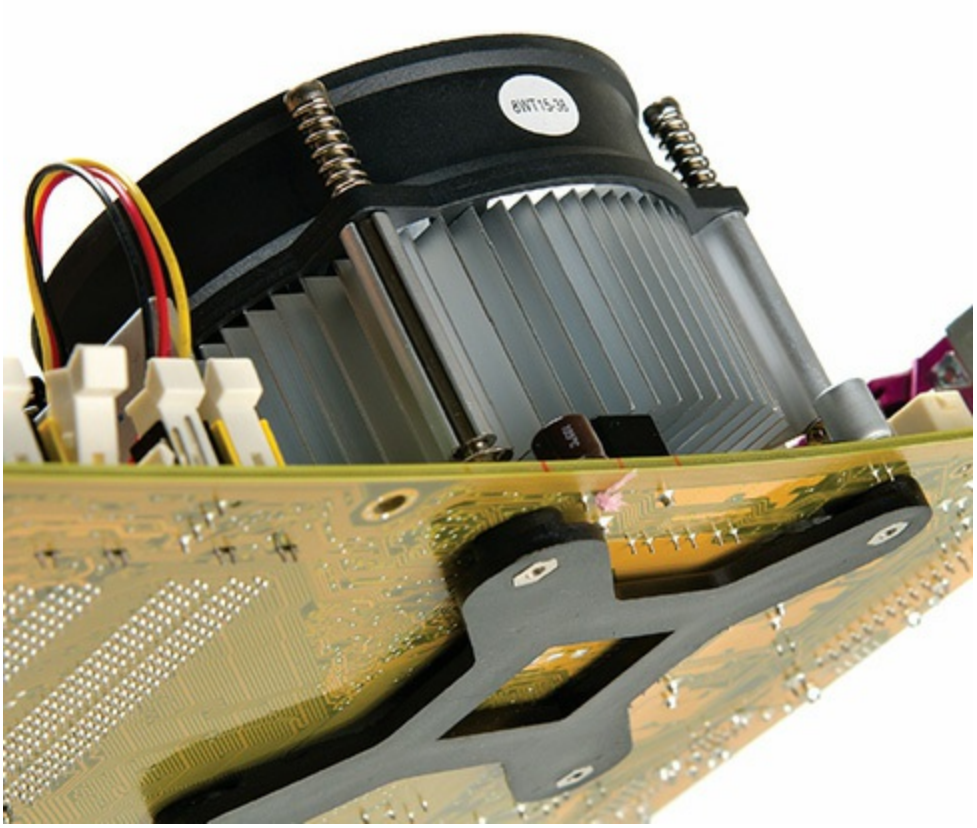

Figure 3-45 Heat-sink and fan assembly mounted to motherboard with screws

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.

NOTE Chapter 5, "Firmware," 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 5!

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 specific 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 *system lockups*, *reboots*, and *unexpected shutdowns*. 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. If a client wants to overclock, explain the potential consequences.

CPU makers do not encourage overclocking. Why would you pay more for a faster processor when you can take a cheaper, slower CPU and just make it run faster? Bowing to enthusiast market pressure, however, both Intel and AMD make utilities that help you overclock their respective CPUs.

- **Intel Extreme Tuning Utility (Intel XTU)** Don't skip the additional Performance Tuning Protection Plan if you go this route.
- **AMD Overdrive Utility** No extra warranty is provided here; you're on your own.

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 3-46).

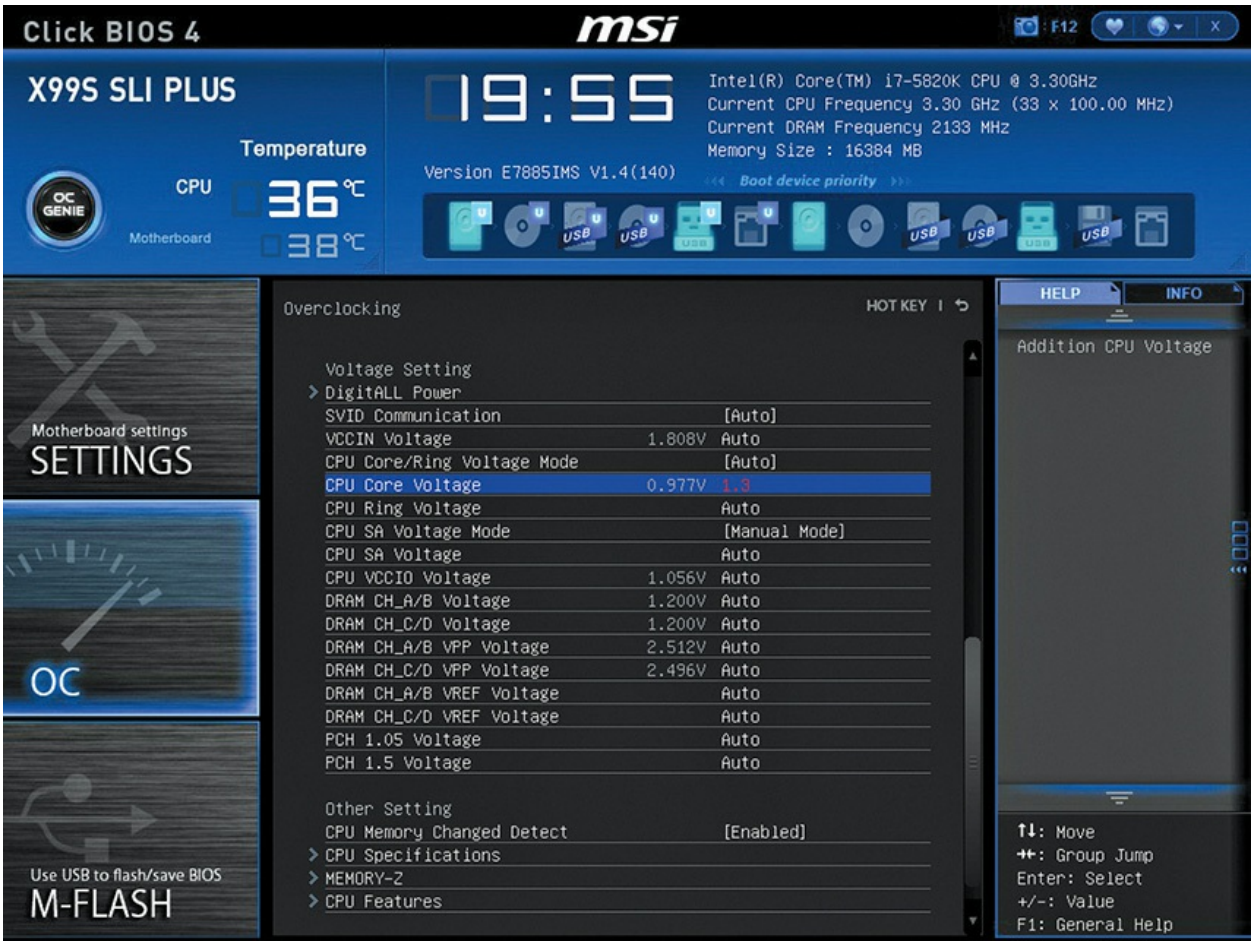

Figure 3-46 Manually overriding CPU settings in the system setup utility

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 or a button called *CMOS clear* or *CLRTC* (see Figure 3-47) that makes the CMOS go back to default settings. Before you try overclocking on a modern system, find the CMOS-clear jumper or button and make sure you know how to use it! Hint: Look in the motherboard manual.

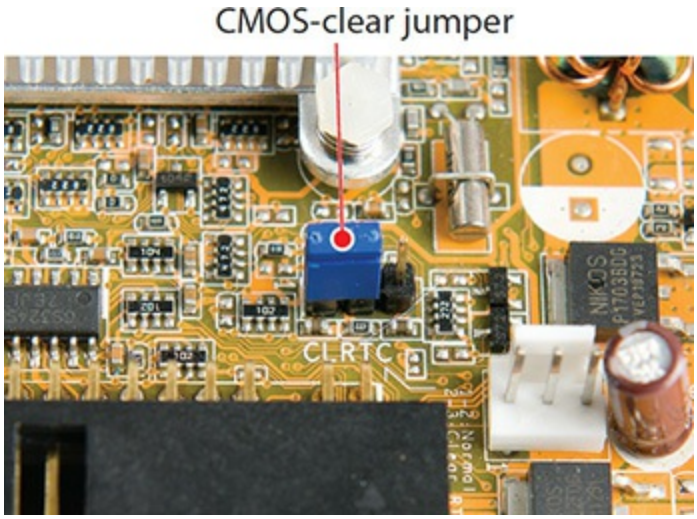

Figure 3-47 CMOS-clear jumper

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.

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 vintage 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. 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 3-48). On macOS, by comparison, you might get a spinning pinwheel that doesn't stop or a kernel panic (i.e., automatic restart). (CompTIA calls the BSoD and pinwheel *proprietary crash screens* and adds a space in the latter, so *pin wheel*.)

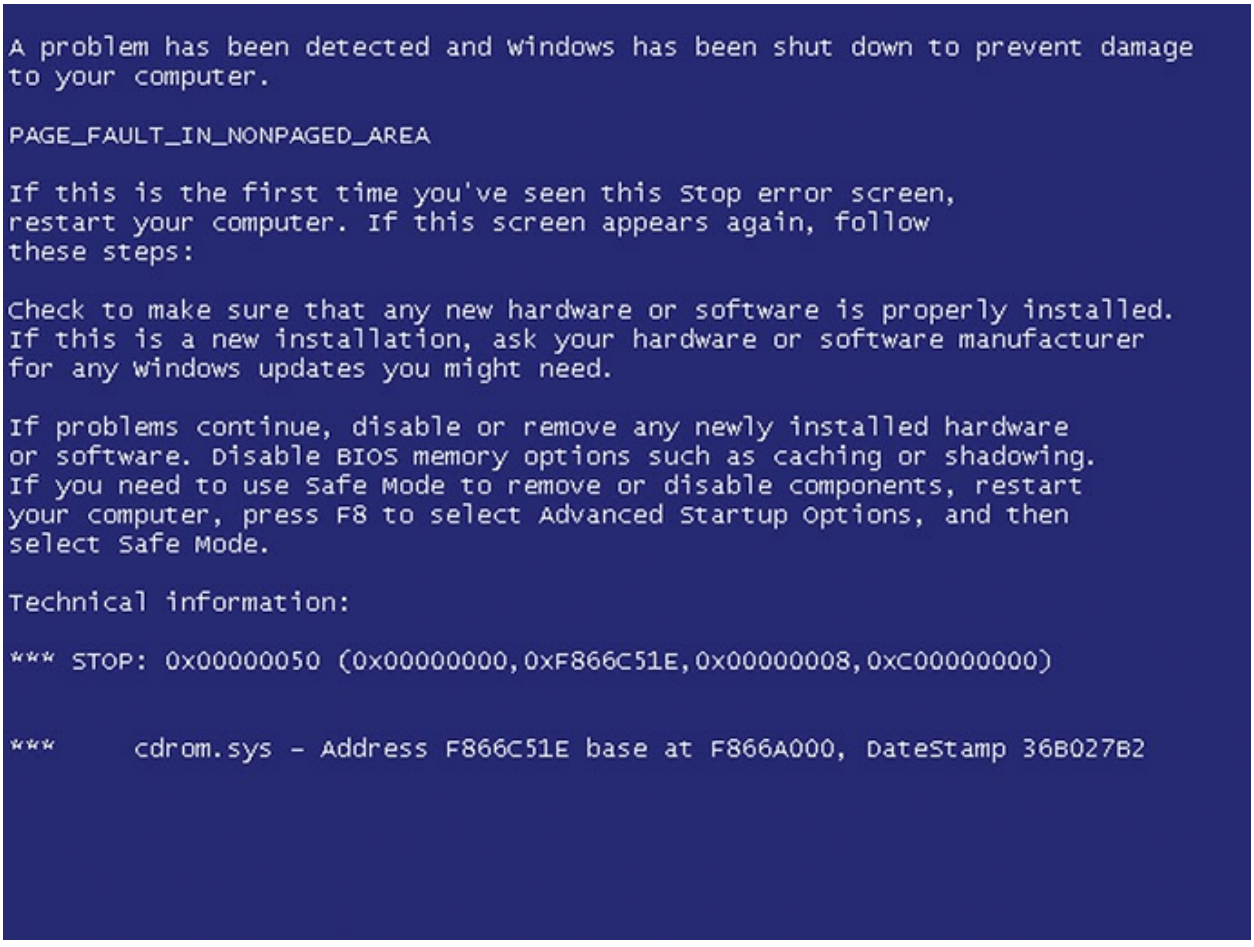

Figure 3-48 Blue Screen of Death

Or the entire computer 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.

Beyond A+

Intel Core M

The Intel Core M runs cool and sips juice for incredibly long battery life in mobile devices. The official TDP is just 4.5 watts—compared to a mobile version of a Core i7 that demands 57 watts. The trade-off Intel makes with the Core M is in raw processing power. It's a little less powerful than a mobile Core i3—enough to get the job done, but not enough to run a serious game or other demanding application. On the other hand, the incredibly low electricity use means manufacturers can skip the fan and make super-skinny devices.

Chapter Review

Questions

1. What do registers provide for the CPU?
   - **A.** Registers determine the clock speed.
   - **B.** The CPU uses registers for temporary storage of internal commands and data.
   - **C.** Registers enable the CPU to address RAM.
   - **D.** Registers enable the CPU to control the address bus.
2. What function does the external data bus have in the PC?
   - **A.** The external data bus determines the clock speed for the CPU.
   - **B.** The CPU uses the external data bus to address RAM.

C. The external data bus provides a channel for the flow of data and commands between the CPU and RAM.

D. The CPU uses the external data bus to access registers.

3. What is the function of the address bus in the PC?

A. The address bus enables the CPU to communicate with the memory controller chip.

B. The address bus enables the memory controller chip to communicate with the RAM.

C. The address bus provides a channel for the flow of data and commands between the CPU and RAM.

D. The address bus enables the CPU to access registers.

4. 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

5. 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

6. Into which socket could you place an Intel Core i5?

A. Socket LGA 2011

B. Socket LGA 1151

C. Socket C

D. Socket AM3+

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

A. Hyper-Threading

B. SpeedStep

C. Virtualization

D. *x*64

8. What steps do you need to take to install a Core i3 CPU into an FM2+ 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 thermal paste; 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.

9. 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 paste 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.

10. Darren 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 paste 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.

Answers

1. **B.** The CPU uses registers for temporary storage of internal commands and data.
2. **C.** The external data bus provides a channel for the flow of data and commands between the CPU and RAM.
3. **A.** The address bus enables the CPU to communicate with the memory controller chip.
4. **A.** The terms megahertz (MHz) and gigahertz (GHz) describe how many million or billion (respectively) cycles per second a CPU can run.
5. **D.** Intel and AMD CPUs come with virtualization support, enabling more efficient implementation of virtual machines.
6. **B.** You'll find Core i5 processors in several socket types, notably LGA 1150 and LGA 1151.
7. **A.** Intel loves its Hyper-Threading, where a single-core CPU can function like a dual-core CPU as long as it has operating system support.
8. **D.** Intel and AMD processors are not compatible at all.
9. **B.** Most likely, the nephew disconnected the CPU fan to get at the RAM slots and simply forgot to plug it back in.
10. **B.** The best answer here is that he forgot the thermal paste, though you can also make an argument for a disconnected fan.