CHAPTER 5

Microprocessors

In this chapter, you will learn how to

- Identify the core components of a CPU
- Describe the relationship of CPUs and memory
- Explain the varieties of modern CPUs
- Install and upgrade CPUs

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

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 creating graphics.

A good PC technician needs to understand some basic CPU functions to support PCs, so let's start with an analysis of how the CPU works. If you wanted to teach someone how an automobile engine works, you would use a relatively simple example engine, right? The same principle applies here. Let's begin our study of the CPU with the granddaddy of all PC CPUs: the famous Intel 8088, invented in the late 1970s. Although this CPU first appeared over 25 years ago, it defined the idea of the modern microprocessor

and contains the same basic parts used in even the most advanced CPUs today. Stick with me, my friend. Prepare to enter that little bit of magic called the CPU.

The Man in the Box

Let's begin by visualizing the CPU as a man in a box (Figure 5-1). This is one clever guy in this box. He can perform virtually any mathematical function, manipulate data, and give answers *very quickly*.

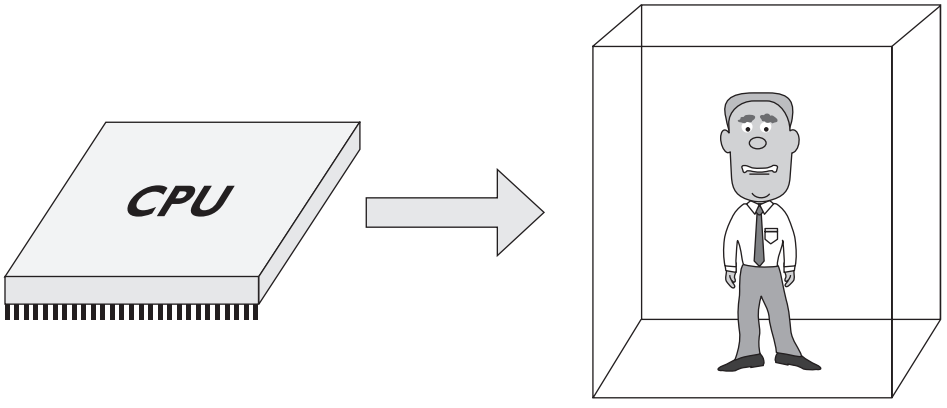


Figure 5-1 Imagine the CPU as a man in a box.

This guy is potentially very useful to us, but there's a catch—he lives closed up in a tiny box. Before he can work with us, we must come up with a way to exchange information with him (Figure 5-2).

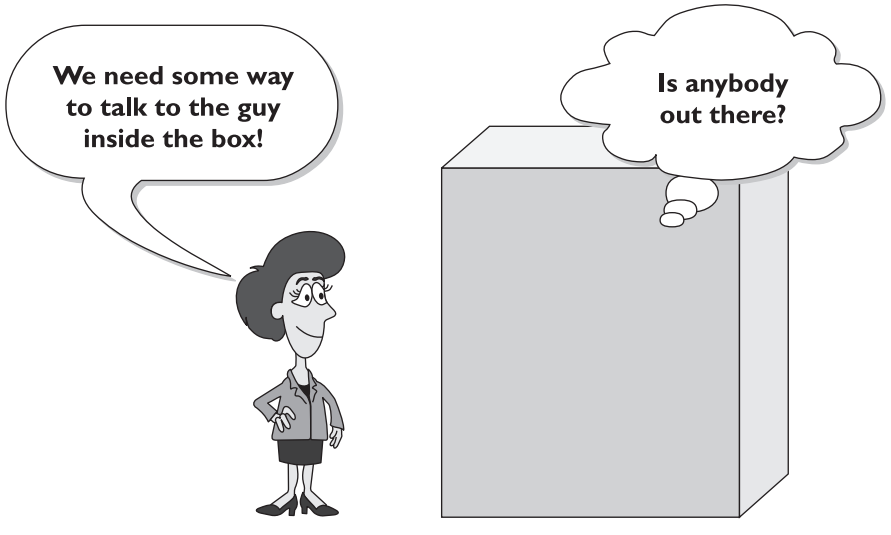


Figure 5-2 How do we talk to the Man in the Box?

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 5-3 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.

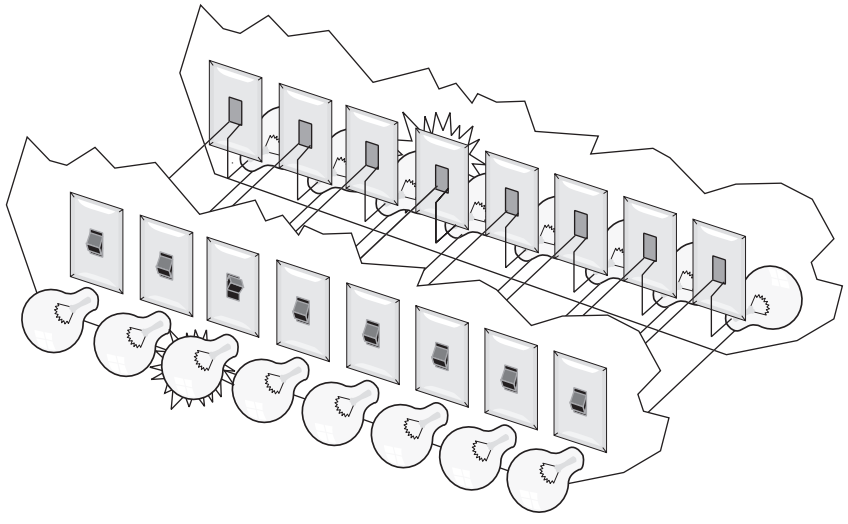


Figure 5-3 Cutaway of the external data bus—note that one light bulb pair is on

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 external data bus 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 external data bus, but you won't see any light bulbs or switches on the CPU. You can, however, see little wires sticking out of the CPU (Figure 5-4). 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 5-4 Close-up of the underside of a CPU

Now that the external data bus 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-off-off-on-on-off-off" when talking about which wires have voltage. Rather than saying that one of the external data bus wires is on or off, use the number 1 to represent on and the number 0 to represent off (Figure 5-5). That way, instead of describing the state of the lights as "on-off-off-off-on-on-off-off," I can instead describe them by writing "10101100."

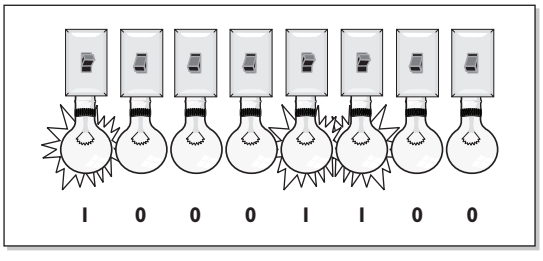


Figure 5-5 Here "1" means on, "0" means off.

In the world of computers, we constantly turn wires on and off. As a result, we can use this "1 and 0" or *binary* system to describe the state of these wires at any given moment. (See, and you just thought computer geeks spoke in binary to confuse normal people. Ha!) There's much more to binary numbering in the world of computing, but this is a great place to start. We will revisit the binary numbering system in greater detail in Chapter 8, "Expansion Bus."

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* (Figure 5-6).

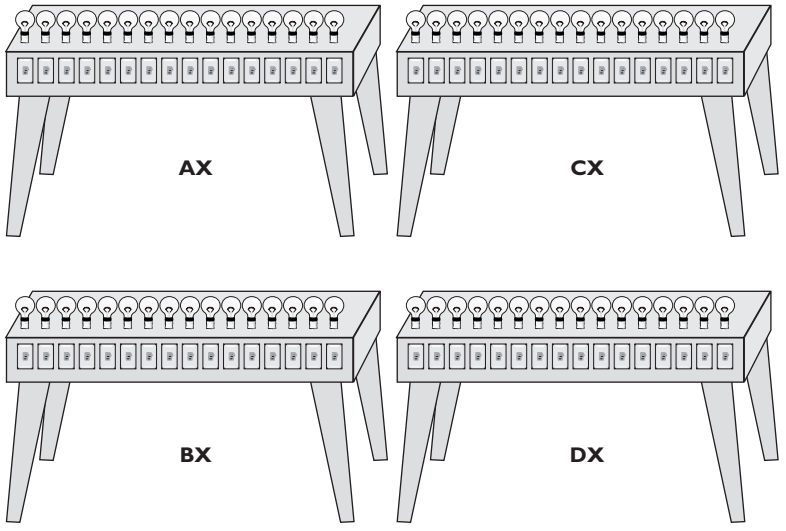


Figure 5-6
The four general-purpose registers

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.

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 we mentioned earlier? Let's make one to enable us to communicate with him. Figure 5-7 shows the codebook we'll use. We'll give one copy to him and make a second for us.

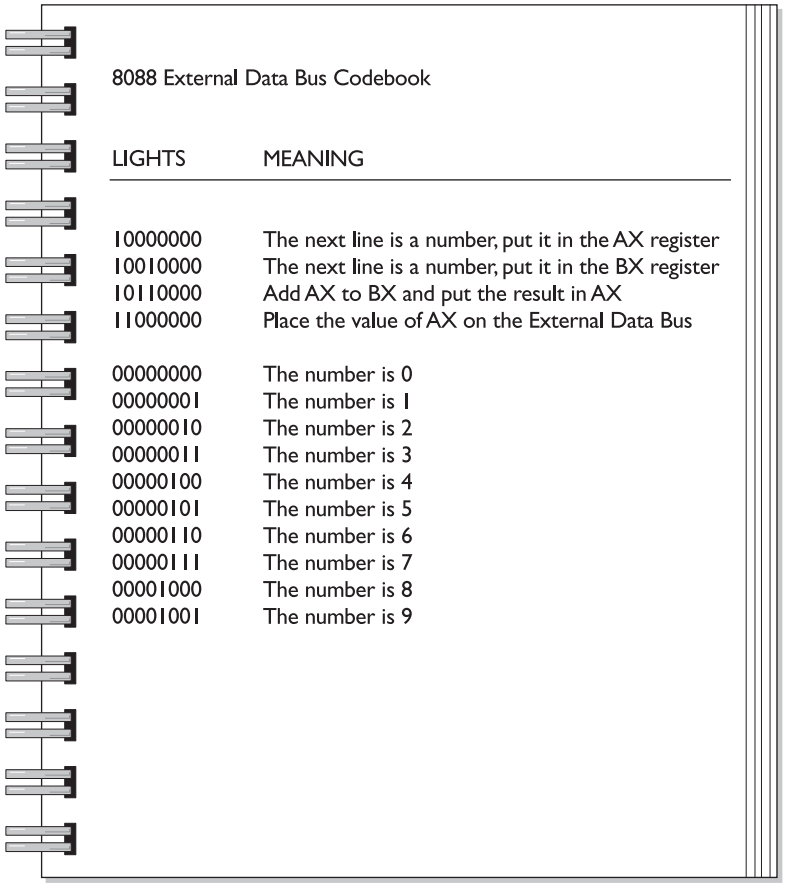
In this codebook, for example, 10000111 means *Move the number 7 into the AX register*. These commands are called the microprocessor's *machine language*. The commands listed in the figure are not actual commands; as you've probably guessed, I've simplified dramatically. The Intel 8088 CPU, invented in the late 1970s, 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 external data bus 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*.

Figure 5-7
CPU codebook

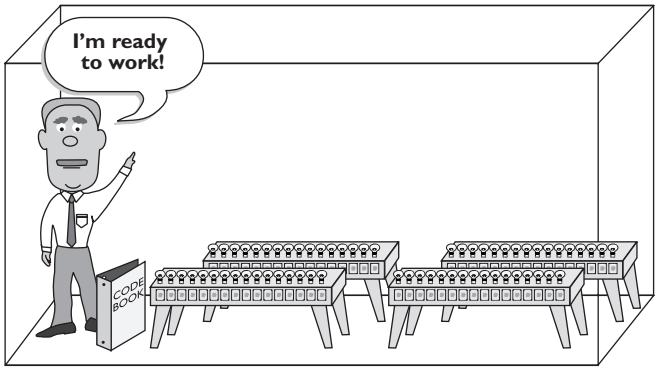


8088 External Data Bus Codebook

LIGHTS	MEANING
10000000	The next line is a number, put it in the AX register
10010000	The next line is a number, put it in the BX register
10110000	Add AX to BX and put the result in AX
11000000	Place the value of AX on the External Data Bus
00000000	The number is 0
00000001	The number is 1
00000010	The number is 2
00000011	The number is 3
00000100	The number is 4
00000101	The number is 5
00000110	The number is 6
00000111	The number is 7
00001000	The number is 8
00001001	The number is 9

So here is the CPU so far: the Man in the Box can communicate with the outside world via the external data bus; 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 external data bus (Figure 5-8).

Figure 5-8
The CPU so far



Clock

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

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

Well, a CPU also has a type of crank. To return to the Man in the Box, imagine there's a bell inside the box activated by a button on the outside of the box. Each time you press the button to sound the bell, the Man in the Box reads the next set of lights on the external data bus. 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 waiting to be processed (Figure 5-9).

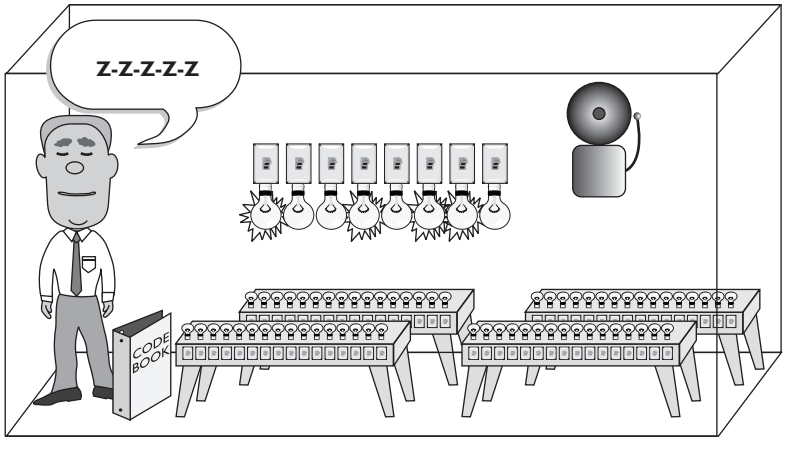


Figure 5-9 The CPU does nothing until activated **by** the clock.

For the CPU to process a command placed on the external data bus, a certain minimum voltage must be applied to the CLK wire. A single charge to the CLK wire is called a *clock cycle*. Actually, the CPU requires at least two clock cycles to act on a command, and usually more. Using the manual calculator analogy, you need to pull the crank at least twice before anything happens. In fact, a CPU may require hundreds of clock cycles to process some commands (Figure 5-10).

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

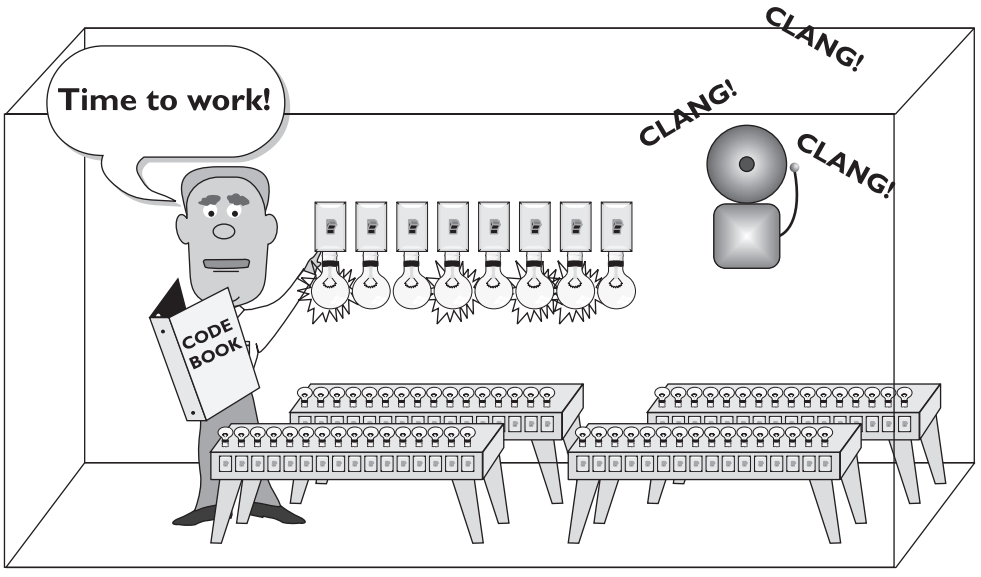


Figure 5-10 The CPU often needs more than one clock cycle to get a result.

standards, but still a pretty big number compared to using a pencil and paper. CPUs today run at speeds in excess of 3 GHz (3 billion cycles per second).

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

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 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.

Understand that a CPU's clock speed is its *maximum* speed, not the speed at which it *must* run. A CPU can run at any speed, as long as that speed does not exceed its clock speed. Manufacturers used to print the CPU's clock speed directly onto the CPU, but for the past few years they've used cryptic codes (Figure 5-11). As the chapter progresses you'll see why they do this.

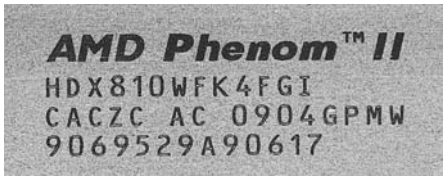


Figure 5-11 Where is the clock speed?

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 (Figure 5-12). The quartz oscillator sends out an electric pulse at a certain speed, many millions of times per second. This signal goes first to a clock chip that adjusts the pulse, usually increasing the pulse sent by the crystal by some large multiple. (The folks who make motherboards could connect the crystal directly to the CPU's clock wire, but then if you wanted to replace your CPU with a CPU with a different clock speed, you'd need to replace the crystal too.) As long as the PC is turned on, the quartz oscillator, through the clock chip, fires a charge on the CLK wire, in essence pushing the system along.

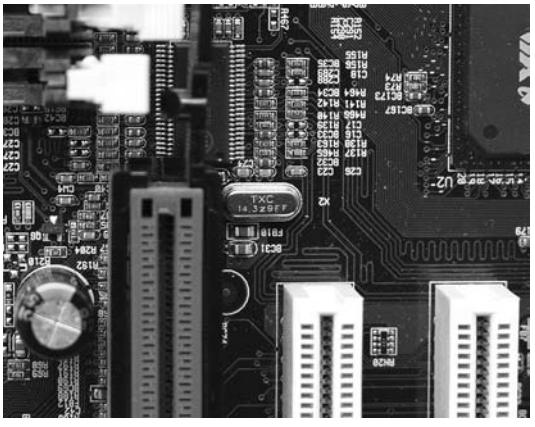


Figure 5-12 One of many types of system crystals

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.

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. This is a risky business that can destroy your CPU, but those willing to take that risk often do it. See the "Overclocking" section later in this chapter.

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 not-so-old days, this required very careful adjustments. With today's systems, the motherboard talks to the CPU (at a very slow speed), 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 Some motherboards enable you to override the default or automatic settings by changing a jumper or making a change in CMOS. A few enthusiasts' motherboards even enable you to make software changes to alter the speed of your CPU.

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, microscopic semiconductor circuits. When one of these circuits holds a charge, you can think of the light bulb as on; no charge, the light bulb is off.

Figure 5-13 is a diagram of a real 8088 CPU, showing the wires that comprise the external data bus and the single clock wire. Because the registers are inside the CPU, you can't see them in this figure.

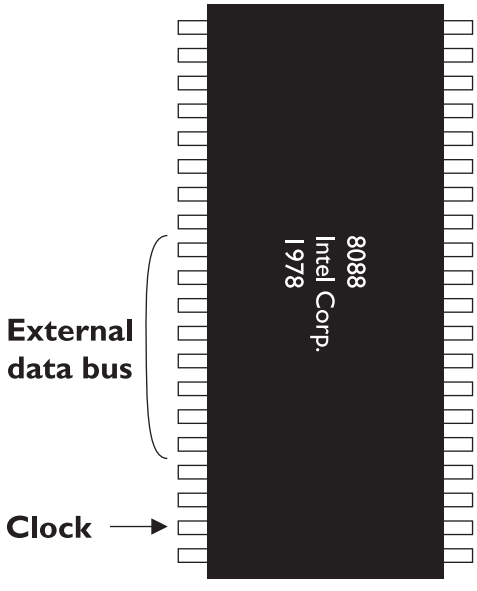


Figure 5-13 Diagram of an Intel 8088 showing the external data bus and clock wires

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 5-7 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 external data bus is a line of code. This program, therefore, has six lines of code.

Memory

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

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

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

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

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

on a hard drive. By addressing each line of memory, other programs will know where to find the data.

Memory and RAM

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

In Chapter 6, "RAM," the concept of RAM is developed in detail, so for now let's look at RAM as an electronic spreadsheet, like one you can generate in Microsoft Excel (Figure 5-14). Each cell in this spreadsheet can store only a one or a zero. Each cell is called a *bit*. Each row in the spreadsheet is eight bits across to match the external data bus of the 8088. Each row of eight bits is called a *byte*. In the PC world, RAM transfers and stores data to and from the CPU in byte-sized chunks. RAM is, therefore, arranged in byte-sized rows. Here are the terms used to talk about quantities of bits:

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

Figure 5-14 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 (Figure 5-15). In Chapter 6, "RAM," 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 5-15
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 the generations of processors 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. Chapter 11, "Hard Drive Technologies," Chapter 12, "Implementing Hard Drives," and Chapter 13, "Removable Media," 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 (Figure 5-16).

Wait a minute. This is not a matter of just plugging the RAM into the external data bus wires! RAM is a spreadsheet with thousands and thousands of discrete rows, and you only need to look at the contents of one row of the spreadsheet at a time, right? So

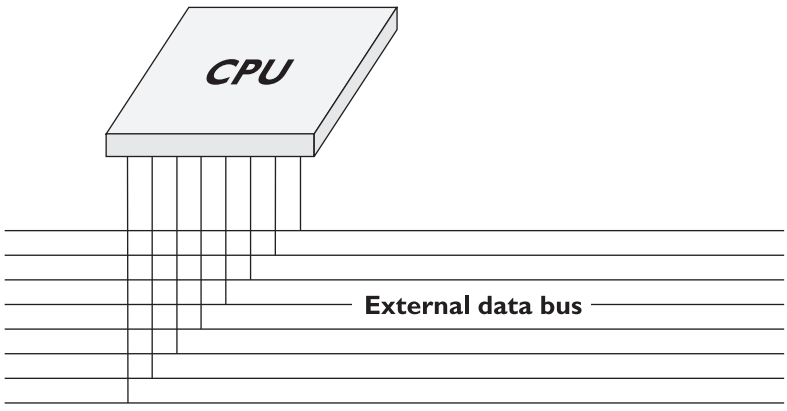


Figure 5-16 Extending the EDB

how do you connect the RAM to the external data bus 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 external data bus. Wouldn't you know I just happen to have such a chip? This chip comes with many names, but for right now just call it the *memory controller chip* (*MCC*),

The MCC contains special circuitry so it can grab the contents of any single line of RAM and place that data or command on the external data bus. This in turn enables the CPU to act on that code (Figure 5-17).

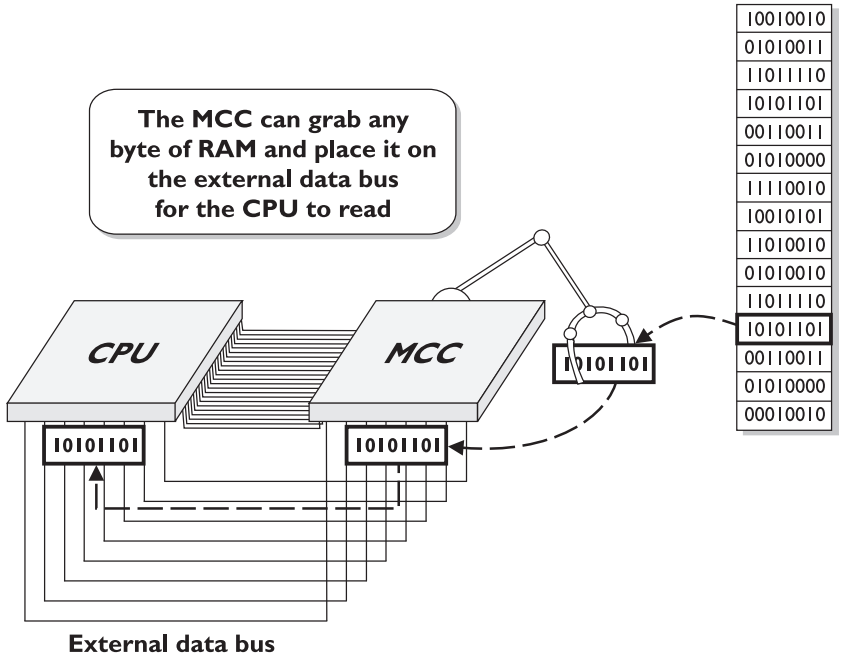


Figure 5-17 The MCC grabs a byte of RAM.

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 (Figure 5-18).

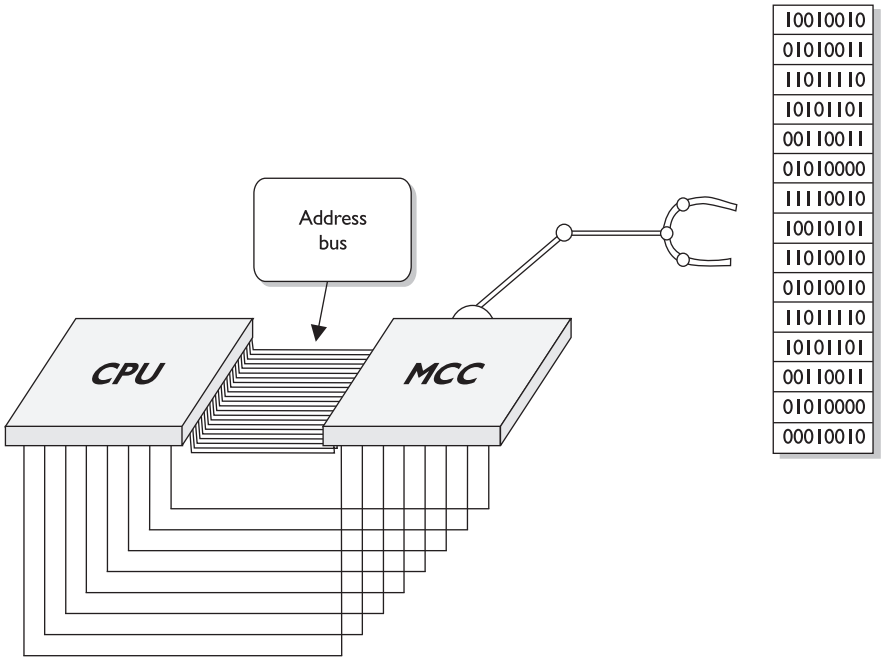


Figure 5-18 Address bus

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 64 kilobytes, but that was considered plenty back in the Dark Ages of Computing in the early 1980s.

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

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

1 kilo = 2^{10} = 1,024 (abbreviated as "K")

1 kilobyte = 1,024 bytes (abbreviated as "KB")

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

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

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

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

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

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

1 kilo is *not* equal to 1,000 (one thousand)

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

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

1 tera is *not* equal to 1,000,000,000,000 (one trillion)
(But they are pretty close!)

Of course, 1 kilo is equal to 1,000 when you talk in terms of the metric system. It also means 1,000 when you talk about the clock speed of a chip, so 1 KHz is equal to 1,000 Hz. When you talk storage capacity, though, the binary numbers kick in, making 1 KB = 1,024 bytes. Got it? This same bizarre dual meaning applies all the way up the food chain, so 1 MHz is 1,000,000 Hz, but 1 MB is 1,048,576 bytes; 1 GHz is 1 billion Hz, but 1 GB is 1,073,741,824 bytes; and so on.

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

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

Essentials

Modern CPUs

Modern CPUs retain the core structures of the Intel 8088, such as registers, instruction sets, and, of course, the *arithmetic logic unit* (*ALU*)—our friend, the Man in the Box. But in the decades of the personal computer, many manufacturers have risen to challenge Intel's dominance—some have even survived—and all processor makers have experimented

with various processor shapes, connectors, and more. The amazing variety of modern CPUs presents unique challenges to a new tech. Which processors go on which motherboards? Can a motherboard use processors from two or more manufacturers? Aren't processors all designed for PCs and thus interchangeable?

This section maps out the modern processor scene. It starts with a brief look at the manufacturers so you know who the players are. Once you know who's making the CPUs, we'll go through the generations of CPUs in wide use today, starting with the Intel Pentium. All modern processors share fundamental technology first introduced by Intel in the Pentium CPU. I use the Pentium, therefore, to discuss the details of the shared technology, and then add specific bonus features when discussing subsequent processors.

Manufacturers

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

Intel

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

NOTE As we go to print, Intel has announced a simplified naming scheme for the majority of its new processors, all under the Core brand name. The Core i3, Core i5, and Core i7 processor names quickly relay where the processor fits in relation to other Core processors. The Core i7 will offer more power than the Core i5, for example. My guess is that they'll keep the Atom processor line separate and distinct from the Core family.

AMD

You can't really talk about CPUs without mentioning Advanced Micro Devices—the Cogswell Cogs to Intel's Spacely Sprockets. AMD makes superb CPUs for the PC market and provides competition that keeps Intel on it toes. Like Intel, AMD doesn't just make CPUs, but their CPU business is certainly the part that the public notices. AMD has made CPUs that clone the function of Intel CPUs. If Intel invented the CPU used

in the original IBM PC, how could AMD make clone CPUs without getting sued? Well, chipmakers have a habit of exchanging technologies through cross-license agreements. Way back in 1976, AMD and Intel signed just such an agreement, giving AMD the right to copy certain types of CPUs.

The trouble started with the Intel 8088. Intel needed AMD to produce CPUs. The PC business was young back then, and providing multiple suppliers gave IBM confidence in their choice of CPUs. Life was good. 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 (Figure 5-19). You could yank an Intel CPU out of a system and snap in an AMD CPU—no problem!

Figure 5-19 Identical Intel and AMD 486 CPUs from the early 1990s

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

CPU Packages

One of the many features that make PCs attractive is the ability for users (okay, maybe advanced users) to replace one CPU with another. If you want a removable CPU, you need your CPUs to use a standardized package with a matching standardized socket on the motherboard. CPUs have gone through many packages, with manufacturers changing designs like snakes shedding skins. The fragile little DIP package of the 8088 (Figure 5-20)

Figure 5-20 The dual inline pin package of the Intel 8088

gave way to rugged slotted processors in the late 1990s (Figure 5-21), which have in turn given way to CPUs using the now prevalent grid array packaging.

Figure 5-21 An AMD Athlon Slot A processor

The grid array package has been popular since the mid-1980s. The most common form of grid array is the *pin grid array (PGA)*. PGA CPUs are distinguished by their square shape with many—usually hundreds—of tiny pins (Figure 5-22).

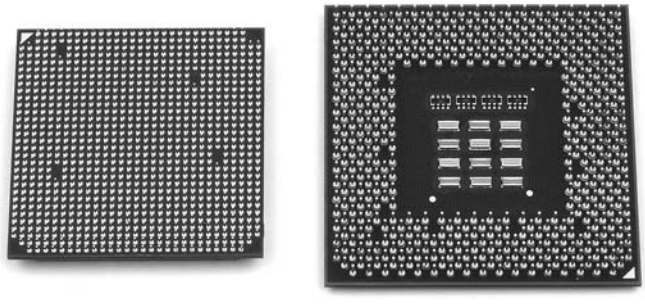


Figure 5-22 Samples of PGA packages

Collectively, Intel and AMD have used close to 100 variations of the PGA package over the years for hundreds of CPU models with such names as staggered-PGA, micro-PGA, ball grid array (which uses tiny balls instead of pins), and land grid array (which uses flat pads instead of pins). Other varieties of PGA CPUs are based on the number of pins sticking out of the CPU. The CPUs snap into special sockets on the motherboard, with each socket designed to match the pins (or balls or pads) on the CPU. To make CPU insertion and removal easier, these sockets—officially called *zero insertion force (ZIF) sockets*—use a small arm on the side of the socket (Figure 5-23) or a cage that fits over the socket (Figure 5-24) to hold the CPU in place. ZIF sockets are universal and easily identified by their squarish shape.

NOTE Although there are many types of PGA packages, most techs just call them all "PGA."

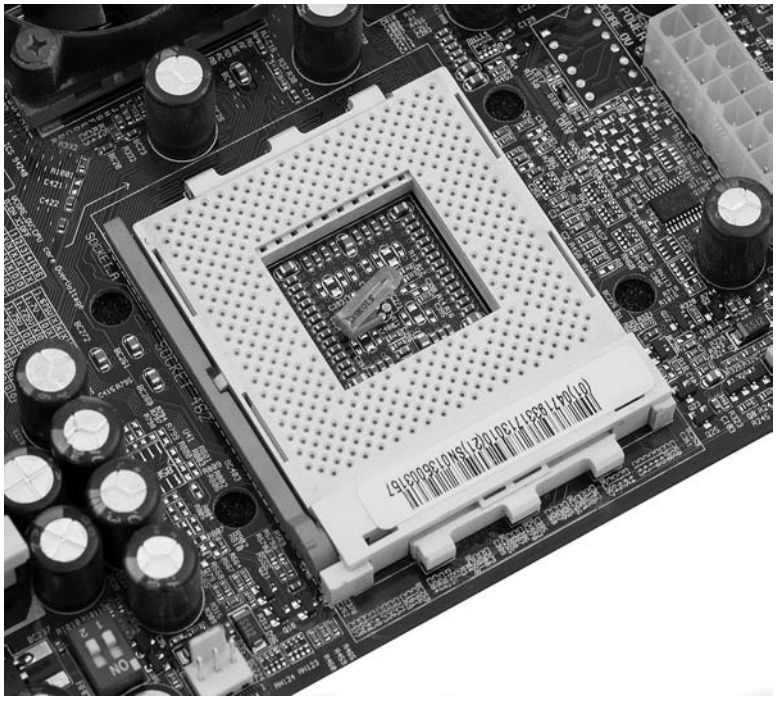


Figure 5-23
ZIF socket with arm on side

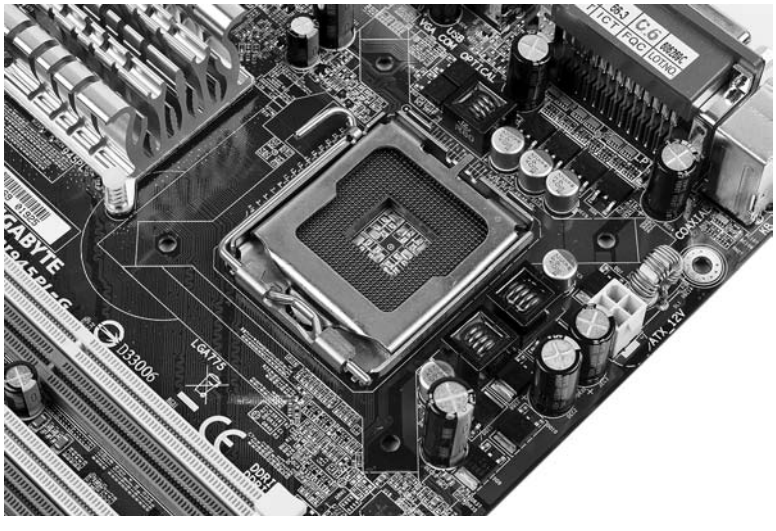


Figure 5-24
ZIF socket with cage over the top

The first generations of sockets used a numbering system that started with Socket 1 and went through Socket 8. Because of the hassle of trying to remember how many pins went with each type of socket, CPU makers started giving all sockets a name based on the number of pins. Most sockets today have names like Socket 1366 and Socket 775 to reflect the number of pins.

NOTE AMD CPUs and sockets have totally different numbering systems than Intel CPUs and sockets, so techs often use the name of the socket instead of AMD or Intel. For example: "Hey, did you see that Socket 1366 motherboard?"

It's very important to know the more common CPU/socket types. As you go through each type of CPU in this chapter, pay attention to the socket types used by those particular CPUs.

The Pentium CPU: The Early Years

Since the advent of the 8088 way back in the late 1970s, CPU makers have made many improvements. As technology has progressed from the 8088 to the most current CPUs, the sizes of the external data bus, address bus, and registers have grown dramatically. The clock speeds at which CPUs run have kept pace, getting faster and faster with each successive generation of processor. The 1980s were an exciting time for CPU technology. The 8088 CPU was supplanted by a series of improved processors with names such as 80286, 80386, and 80486 (Figure 5-25). These CPU families incorporated wider buses, increasingly higher clock speeds, and other improvements.

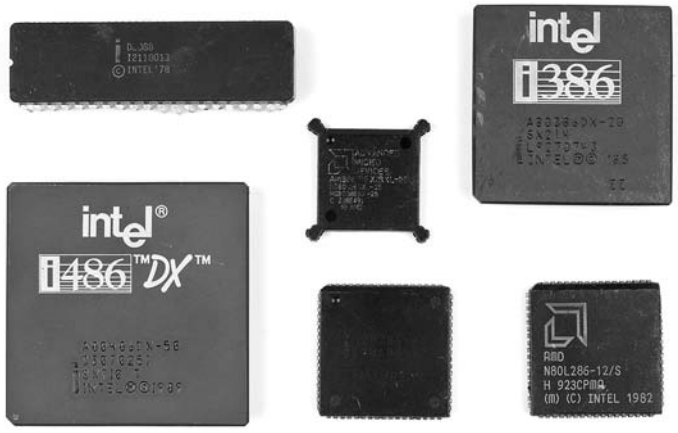


Figure 5-25 Old CPUs

In the early 1990s, Intel unveiled the Pentium CPU. Although no longer manufactured, the original Pentium CPU was the first Intel CPU to contain all of the core functions that define today's modern CPUs.

NOTE Many of the CPU features attributed here to the Pentium actually appeared earlier, but the Pentium was the first CPU to have *all* of these features.

The Pentium retained the core features of the 8088 and subsequent processors, although the clock was much faster, the address bus and external data bus were wider,

and the registers had more bits. You'll also see a number of other improvements that simply didn't exist on the original 8088.

The Rise of 32-bit Processing

The old 8088 had 16-bit registers, an 8-bit EDB, and a 20-bit address bus. Old operating systems (such as DOS and early versions of Windows) were written to work on the 8088. Over the years, later CPUs gradually increased their address buses and general-purpose register sizes to 32 bits, allowing much more powerful operating systems (such as Linux, Windows XP, and Windows Vista) to work with the Pentium to process larger numbers at a single time and to address up to 2^{32} = 4,294,967,296 = 4 gigabytes of RAM (see Figure 5-26). Running 32-bit operating systems on 32-bit hardware is called *32-bit processing*.

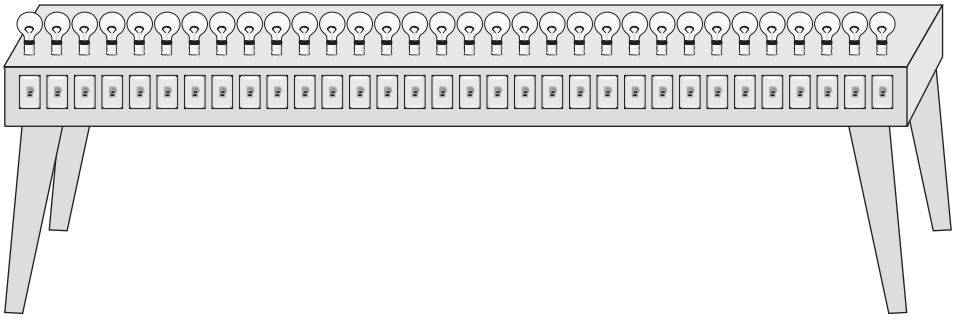


Figure 5-26 A 32-bit register

Both AMD and Intel now make 64-bit processors that address up to 2^{64} = 18,446,744,073,709,551,616 bytes of RAM. To take advantage of this larger address bus, a 64-bit version of the operating system must be used. You'll learn more about 64-bit processors later in this chapter.

Pipelining

Remember earlier when we talked about pulling the crank multiple times to get an answer out of the CPU? This is because to get a command from the EDB, do the calculation, and then get the answer back out on the EDB, the 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 done.
3. **Execute** Perform the calculation.
4. **Write** Send the data back onto the EDB.

Smart, discrete circuits inside your CPU handle each of these stages. In early CPUs, when a command was placed on the EDB, 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 (Figure 5-27)!

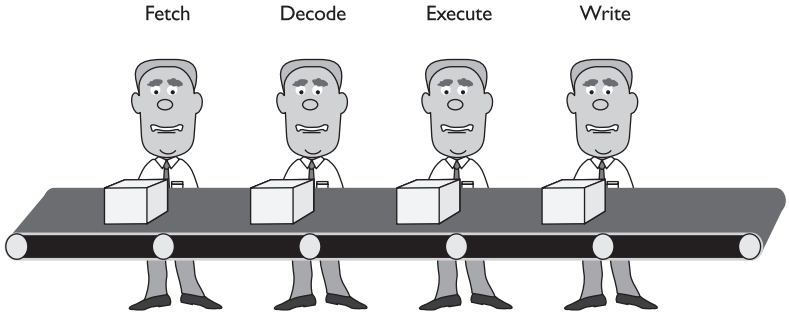


Figure 5-27
Simple pipeline

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 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, called *pipeline stalls*. The decode stage tends to cause the most pipeline stalls; certain commands are complex and therefore harder to decode than other commands. The Pentium used two decode stages to reduce the chance of pipeline stalls due to complex decoding.

NOTE After the Pentium, pipelines kept getting longer, reaching up to 20 stages in the Pentium 4. Since then, Intel and AMD have kept CPU pipelines around 12 stages (although this could change again).

Pipelining certainly helped the Pentium run more efficiently, but there's another issue: the execute stage. 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 *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 more than 90 percent of its work doing integer math. But the Pentium also had special circuitry to handle complex numbers, called the *floating point unit* (*FPU*). With a single pipeline, only the integer unit or the floating point unit 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 floating point finished executing the complex command (Figure 5-28).

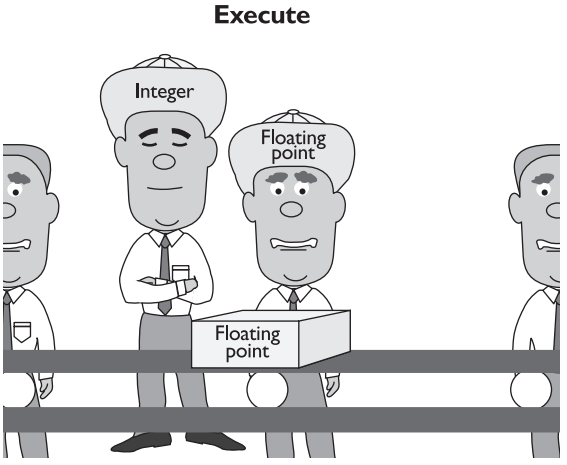


Figure 5-28
Bored integer unit

NOTE You'll see the integer unit referred to as the arithmetic logic unit (ALU) in many sources. Either term works.

To keep things moving, the folks at Intel gave the Pentium two pipelines: one main, do-everything pipeline and one that only handled integer math. Although this didn't *stop* pipeline stalls, a second pipeline kept running when the main one stalled (see Figure 5-29).

The two pipelines on the old Pentium were so successful that Intel and AMD added more and more pipelines to subsequent CPUs. Most CPUs today have around eight pipelines, although there's tremendous variance from CPU to CPU.

NOTE One of the biggest differences between equivalent AMD and Intel processors is the pipelines. AMD tends to go for lots of short pipelines, whereas Intel tends to go with just a few long pipelines.

CPU Cache

When you send a program to the CPU, you actually run lots of little programs all at the same time. Okay, let's be fair here: *you* didn't send all of these little programs—you just started your Web browser or some other program. The moment you double-clicked that icon, Windows started sending lots of 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.

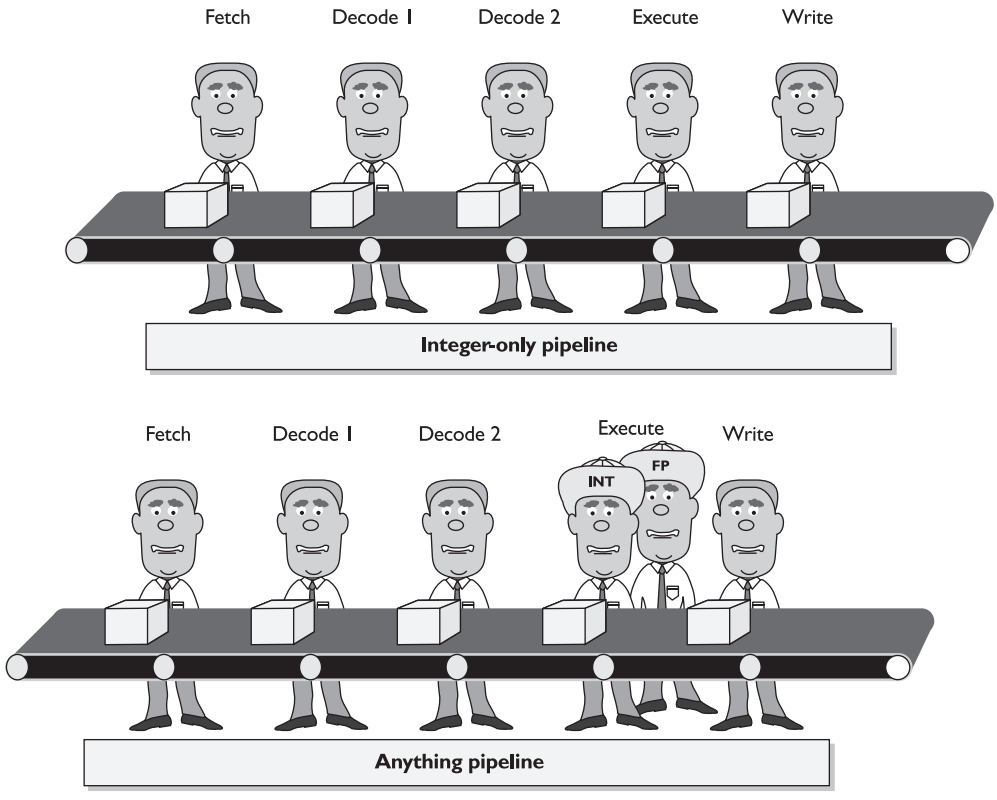


Figure 5-29 The Pentium dual pipeline

Take a look at Figure 5-30. It shows four programs in RAM: a Web browser, Solitaire, an image-editing program, and an e-mail program. Note they are not the same size. Some programs need more RAM than others.

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, the Pentium came with built-in, very high-speed RAM called *static RAM* (*SRAM*). This SRAM would preload as many instructions as possible and would also keep copies of already run instructions and data in the hope that the CPU would need to work on them again (see Figure 5-31). SRAM used in this fashion is called a cache.

The SRAM cache inside the CPU was tiny, only about 16 KB, but it improved performance tremendously. In fact, it helped so much that many motherboard makers began

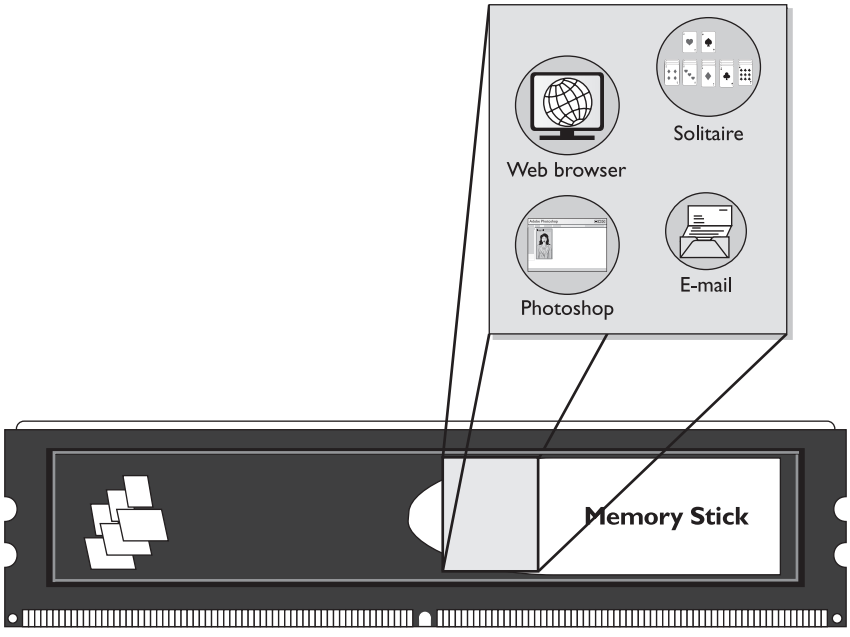


Figure 5-30 Four programs in RAM

adding a cache directly to the Pentium 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. Later, engineers took this cache concept even further and added the L2 cache onboard the CPU. Some CPUs even include three caches: an L1, an L2, and an L3 cache.

NOTE You'll hear caches referred to as Level 1, Level 2, and Level 3 for L1, L2, and L3 cache, respectively. Any of the terms are acceptable.

NOTE It's tempting to ask why processor manufacturers didn't just include bigger L1 caches instead of making onboard L1 and L2 caches. The answer is that a very small L1 and a larger L2 are much more efficient than a single fast L1.

The Pentium was capable of *branch prediction*, a process whereby the CPU attempted to anticipate program branches before they got to the CPU itself. An IF statement provides a nice example of this: "If the value in the AX register = 5, stop running this code and jump to another memory location." Such a jump would make all of the data in the

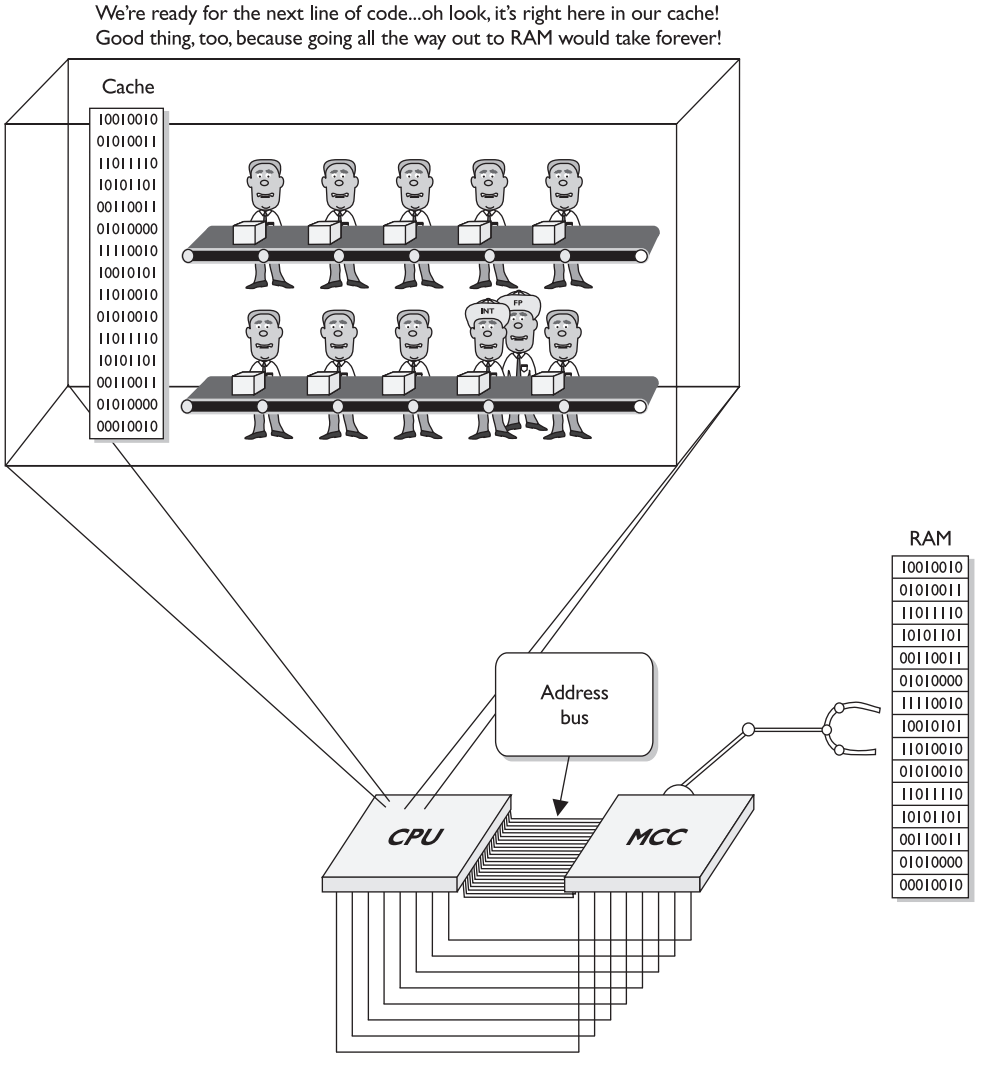


Figure 5-31 RAM cache

cache useless. The Pentium could recognize a branch statement. Using a counter that kept a record about the direction of the previous branch, the CPU would guess which way the branch was going to go and make sure that side of the branch was in its cache. The counter wasn't perfect, but it was right more often than it was wrong.

Clock Speed and Multipliers

In the earliest motherboards, the clock chip pushed every chip on the motherboard, not just the CPU. This setup worked great for a while until it became obvious that CPU

makers (really Intel) could make CPUs with a much higher clock speed than the rest of the chips on the motherboard. So Intel had a choice: Either stop making faster CPUs or come up with some way to make CPUs run faster than the rest of the computer.

To overcome this problem, Intel developed clock-multiplying CPUs. A *clock-multiplying CPU* takes the incoming clock signal and multiplies it inside the CPU to let the internal circuitry of the CPU run faster. The secret to making clock multiplying work is caching. CPUs with caches spend the majority of their clock cycles performing calculations and moving data back and forth within the caches, not sending any data on the external buses.

NOTE Clock multiplying first surfaced during the reign of the Intel 80486 CPUs. The first clock multipliers exactly doubled the clock speed, resulting in the term *clock doubling*. This term is used interchangeably with *clock multiplying*, even though modern CPUs multiply far more than just times two.

All modern CPUs are clock multipliers. So in reality, every CPU now has two clock speeds: the speed that it runs internally and the speed that it runs when talking on the address bus and the external data bus. Multipliers run from 2× up to almost 30×! Multipliers do not have to be whole numbers. You can find a CPU with a multiplier of 6.5× just as easily as you would find one with a multiplier of 7×. A late-generation Pentium would have an external speed of 66 MHz multiplied by 4.5× for an internal speed of 300 MHz. The Intel Pentium 4 3.06-GHz CPU runs at an external speed of 133 MHz with a 23× multiplier to make—yes, you've got it—3.06 GHz. Without the invention of multiplying, modern CPUs would be nowhere near their current blazing speeds.

The clock speed and the multiplier on Pentium CPU systems had to be manually configured via jumpers or DIP switches on the motherboard (Figure 5-32). Today's CPUs actually report to the motherboard through a function called CPUID (CPU identifier), and the speed and multiplier are set automatically.

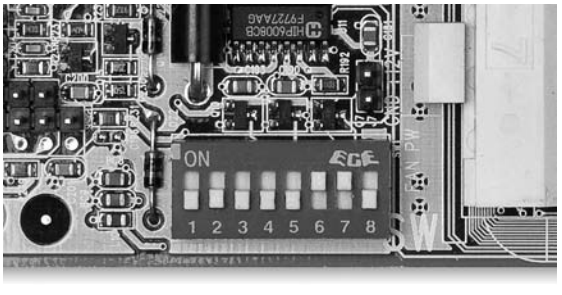


Figure 5-32 DIP switch on an old motherboard

For years, users pushed for faster and faster CPU clock speeds, because clock speed was considered the most important way to differentiate one CPU from another. By 2003, advancements in caching, pipelining, and many other internal aspects of the CPU made clock speed an inaccurate way to compare one CPU to another. CPU makers give their

processors model numbers—nothing more than marketing names—to tell one processor from another. The Intel Core Duo T2300, for example, actually runs 1.66 GHz (166 MHz external speed with a 10× multiplier). If you want to know the speed of a particular processor, you must go to the CPU maker's Web site or other source.

CPU Voltages

In the simplest sense, a CPU is a collection of *transistors*, tiny electrical switches that enable the CPU to handle the binary code that makes up programs. Transistors, like other electrical devices, require a set voltage to run properly. Give a transistor too much and you fry it, too little and it doesn't work. For the first ten years of the personal computer, CPUs ran on 5 volts of electricity, just like every other circuit on the motherboard. To increase the complexity and capability with new generations of CPUs, microprocessor developers simply increased the number of transistors. But eventually they altered this strategy to increase the efficiency of the CPUs and keep the size down to something reasonable.

Intel and AMD discovered that by reducing the amount of voltage used, you could reduce the size of the transistors and cram more of them into the same space. Intel released the Pentium, for example, that required only 3.3 volts. AMD responded with its versions of the Pentium-class CPUs with even lower voltages.

Motherboard manufacturers had to scramble to adapt to the changing CPU landscape by creating motherboards that could handle multiple voltages of CPUs. All of the logic circuits still ran at 5 volts, so manufacturers started installing a *voltage regulator module* (*VRM*) that damped down voltages specifically for the CPUs.

Because the new and improved motherboards handled many CPU voltages, initially techs had to install a VRM specific to the CPU. As manufacturers got better at the game and built VRMs into the motherboards, techs just had to change jumpers or flip switches rather than install a VRM (Figure 5-33).

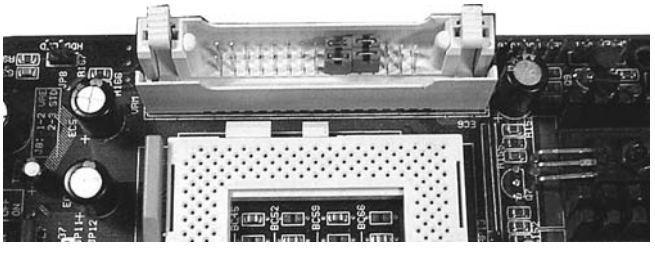


Figure 5-33 Volt regulator module

Getting the voltage right on today's CPUs is no longer a concern. Just as for clock speed and multipliers, today's CPUs tell the motherboard the voltage they need automatically. The integrated VRMs take care of the rest (Figure 5-34).

The feature set of the early Pentium CPUs beats at the heart of every subsequent processor. Newer processors have a 64-bit data bus, 32-bit or larger address bus, 32-bit or larger registers, multiple pipelines, and two or three levels of cache. All run at some multiple of the system clock. So, now that you have the scoop on the Pentium, you're ready to check out subsequent CPU models.

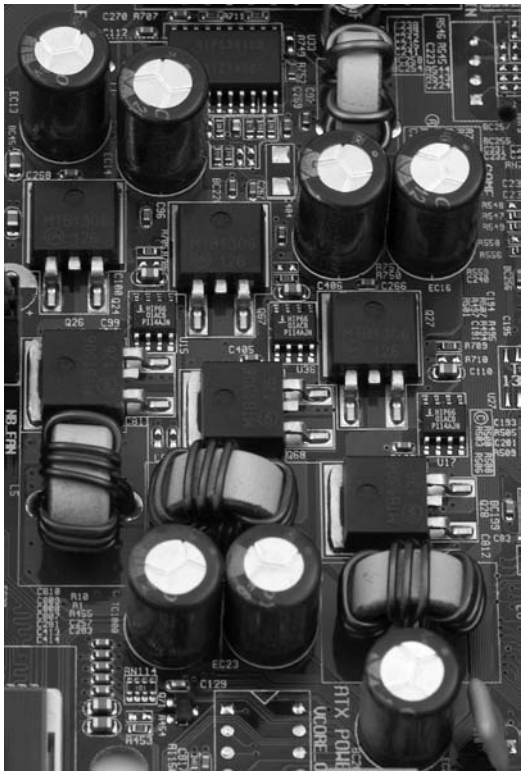


Figure 5-34
Typical motherboard voltage regulators

Original Pentium

The Pentium is not a new chip; it's been around since 1990, and the last versions of the Pentium chip were discontinued in 1995. The original Pentium was the springboard for the Pentium Pro, probably the most important CPU ever made, and thus it makes sense to start there. The rest of this chapter looks at all of the popular CPUs developed since the Pentium and describes how they've built on this legacy CPU (Figure 5-35).

Figure 5-35
An early Pentium

AMD made a competitor to the Pentium called the AMD K5. The AMD K5 was pin-compatible with the Pentium, but to keep Intel from suing them, AMD made the K5 very different on the inside, using a totally new (at least for Intel) method of processing. The AMD K5 had some success, but was rather quickly upstaged by better AMD CPUs.

Pentium Pro

In 1995, Intel released the next generation of CPU, the Pentium Pro, often called the P6. The Pentium Pro was a huge CPU with a distinctive, rectangular PGA package (Figure 5-36). The P6 had the same bus and register sizes as the Pentium, but three new items made the P6 more powerful than its predecessor: quad pipelining, dynamic processing, and an on-chip L2 cache. These features carried on into every CPU version that followed, so many people consider the Pentium Pro to be the true "Father of the Modern CPU."

Figure 5-36 Pentium Pro

Superscalar Execution

The P6 had four pipelines, twice as many as the Pentium. These pipelines were deeper and faster. With this many pipelines, the P6 was guaranteed to always, no matter what, run at least two instructions at the same time. The ability to execute more than one instruction in any one clock cycle is called *superscalar execution*.

Out-of-Order Processing/Speculative Execution

From time to time, a CPU must go to system memory to access code, no matter how good its cache. When a RAM access takes place, the CPU must wait a few clock cycles before processing. Sometimes the wait can be 10 or 20 clock cycles. System memory is dynamic RAM and needs to be refreshed (charged up) periodically, causing further delays. When the P6 was forced into wait states, it took advantage of the wait to look at the code in the pipeline to see if it could run any commands while the wait states were active. If it found commands it could process that were not dependent on the data being fetched from DRAM, it ran these commands out of order, a feature called *out-of-order processing*. After the DRAM returned with the code, it rearranged the commands and continued processing.

The P6 improved on the Pentium's branch prediction by adding a far more complex counter that would predict branches with a better than 90-percent success rate. With the combination of out-of-order processing and the chance of a branch prediction so high, the CPU could grab the predicted side of the branch out of the cache and run it out of order in one pipeline, even before running the branch itself. This was called *speculative execution*.

On-Chip L2 Cache

The P6 had both an L1 and an L2 cache on the CPU. Because the L2 cache was on the chip, it ran almost as fast as the L1 cache (Figure 5-37). Be careful with the term "on-chip." Just because the L2 cache was on the chip, that doesn't mean it was built into

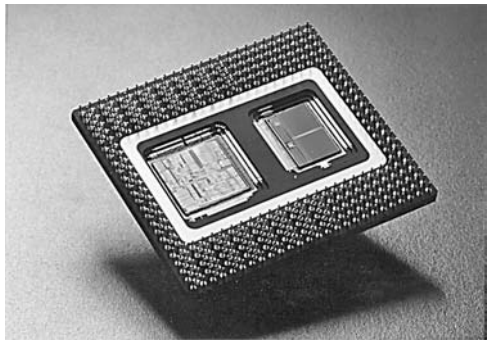


Figure 5-37 A P6 opened to show separate CPU and L2 cache (photo courtesy of Intel)

the CPU. The CPU and the L2 cache shared the same package, but physically they were separate.

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*.

Figure 5-38 shows a more modern configuration, labeling the important buses. Note that the external data bus and address bus are there, but the chipset provides separate address buses and external data buses—one set just for the CPU and another set for the rest of the devices in the PC. No official name has been given to the interface between the RAM and the chipset. On the rare occasions when they discuss it, most techs simply call it the *RAM interface*.

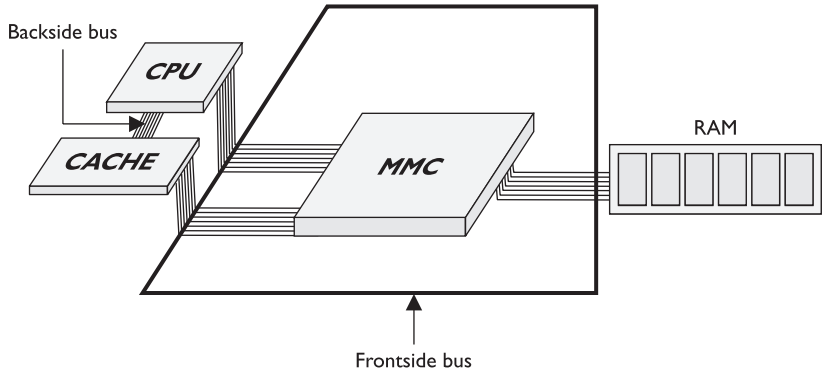


Figure 5-38 Frontside and backside buses

The Pentium Pro had a unique PGA case that fit into a special socket, called Socket 8. No other CPU used this type of socket. The Pentium Pro made strong inroads in the high-end server market, but its high cost made it unacceptable for most people's desktop computers.

Although the Pentium Pro never saw a large volume of sales compared with the Pentium, many people in the industry consider it the most important chip ever created by Intel. Its feature set was the prototype for all CPUs designed ever since.

Later Pentium-Class CPUs

Intel's usual game plan in the rough-and-tumble business of chip making is to introduce a new CPU and simultaneously declare all previous CPUs obsolete. That did not happen with the Pentium Pro, however, because Intel never really developed the P6 for most users. It was to be the CPU for powerful, higher-end systems. This kept the Pentium as the CPU of choice for all but the most power-hungry systems.

Figure 5-39 Later-generation Pentium

While the Pentium Pro languished on the high end for several years, Intel and AMD developed new Pentium-class CPUs that incorporated a series of powerful improvements, some of which were taken from the Pentium Pro. These improvements required that they be regarded as a new family of CPUs, which I call the "later Pentium-class CPUs" (Figure 5-39). Although certainly some profound differences exist between these CPUs, they all have three groups of similar improvements: *multimedia extensions (MMX)*, increased multipliers/clocks, and improved processing.

Later-generation Pentiums were pin-compatible with earlier Pentiums, but included a large number of improvements. The most important improvement was increases in multipliers, and therefore clock speeds, but other improvements also took place—some borrowed from the P6 and some developed just for this new breed of Pentium.

MMX

In 1996, Intel added a new enhancement to its Pentium CPU, called *multimedia extensions* (*MMX*), in response to the large number of programs with heavy graphics needs coming out at this time. MMX was designed to work with large graphics by calculating on large chunks of data and performing vector math (vector math is needed to handle graphical issues such as spinning a 3D object). MMX was not heavily supported by the folks who wrote graphics programs, but MMX did start the idea that CPUs should have special circuitry just for such programs. Over time, the graphics community began to work with Intel to improve MMX, eventually replacing it with better solutions.

Increased Clocks and Multipliers

Later Pentiums all have vastly increased multipliers, resulting in higher speeds. Most early Pentiums used 2.5× multipliers at best, but later Pentium-class processors had up to 4.5× multipliers.

Pentium II

Intel's next major CPU was the Pentium II. Although highly touted as the next generation of CPUs, the Pentium II was little more than a faster Pentium Pro with MMX and

a refined instruction set. The Pentium II came in a distinctive *single-edge cartridge* (*SEC*) that gave more space for the L2 cache and made CPU cooling easier while freeing up more room on the motherboard (Figure 5-40). Aggressive advertising and pricing made the Pentium II extremely popular.

Figure 5-40
Pentium II

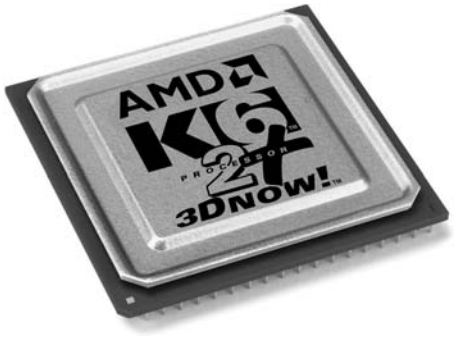
The Pentium II initially achieved the higher clock speeds by using high multiples of a 66-MHz external speed. During this time, however, AMD began to sell CPUs designed to run on 100-MHz motherboards. Although the final Pentium II models also ran on 100-MHz motherboards, Intel's slow adoption of 100-MHz external-speed CPUs lost market share for Intel.

The SEC cartridge also created another problem: it was not free to copy. This prevented other CPU manufacturers from making CPUs that fit in the SEC's special Slot 1 connection and forced AMD to create its own SEC packages that were incompatible with Intel's. From the Pentium II to today, AMD and Intel CPUs are no longer interchangeable. We live in a world where AMD CPUs have motherboards designed for AMD, while Intel CPUs must have motherboards designed for Intel.

AMD K6 Series

From 1997 to 2000, AMD produced a series of processors called the K6 that matched—and in many people's view, surpassed—the Pentium II, propelling AMD into serious competition with Intel (Figure 5-41). Four models were included in the K6 series: the

Figure 5-41
AMD K6
(*photo courtesy of AMD*)



K6, K6-2, K6-2+, and K6-III, each incorporating more advanced features than the previous model. The K6 processors incorporated a number of improvements, including a 64-KB L1 cache, extremely advanced pipelining, and support for motherboard speeds of up to 100 MHz (on later models). The K6-2 added AMD's proprietary 3DNow! instruction set—a direct competitor to Intel's MMX and a significant advancement in graphics-handling capabilities—and increased clock speeds. The K6-III included even more advancements in pipelining and added a 256-KB L2 cache, all on a standard Socket 7 PGA package.

Pentium III

The Pentium III improved on the Pentium II by incorporating *Streaming SIMD Extensions* (*SSE*), Intel's direct competitor to AMD's 3DNow!; a number of internal processing/pipelining improvements; full support for 100-MHz and 133-MHz motherboard speeds; and a high-speed L2 cache. The Pentium III was first produced by using an SEC package (Figure 5-42), but improvements in die technology enabled Intel to produce PGA versions later, ending the short reign of the SEC-package CPUs.

Figure 5-42
Intel Pentium III

Practical Application

Processing and Wattage

To make smarter CPUs, Intel and AMD need to increase the number of microscopic transistor circuits in the CPU. The more circuits you add, the more power they need. CPUs measure their power use in units called watts, just like a common light bulb. Higher wattage also means higher heat, forcing modern CPUs to use powerful cooling methods. Good techs know how many watts a CPU needs, because this tells them how hot the CPU will get inside a PC. Known hot CPUs are often avoided for general-purpose PCs because these CPUs require more aggressive cooling.

NOTE As you read the wattages for the various CPUs, imagine a light bulb with that wattage inside your system unit.

CPU makers really hate heat, but they still want to add more circuits; they constantly try to reduce the size of the circuits, because smaller circuits use less power. CPUs are made from silicon wafers. The electrical circuitry is etched onto the wafers with a process called photo lithography. Photo lithography is an amazingly complex process, but basically requires placing a thin layer of chemicals on the wafer. These chemicals are sensitive to ultraviolet light; if a part of this mask is exposed to UV light, it gets hard and resistant. If it isn't exposed, it's easy to remove. To make the circuitry, a mask of the circuits is placed over the wafer, and then the mask and wafer are exposed to UV light. The mask is removed and the wafer is washed in chemicals, leaving the circuits. If you want microscopic circuits, you need a mask with the pattern of the microscopic circuits. This is done though a photographic process. The old 8088 used a 3-micrometer (one millionth of a meter) process to make the mask. Most of today's CPUs are created with a 45-nanometer process, and a 32-nanometer process is appearing in some chips. The same CPU created with a smaller process is usually cooler.

NOTE A nanometer is one billionth of a meter.

CPU Codenames

Intel and AMD fight to bring out new CPUs with an almost alarming frequency, making the job of documenting all of these CPUs challenging. Luckily for us, the CPU makers use special CPU codenames for new CPUs, such as Bloomfield and Deneb to describe the first version of the Core i7 and the Phenom II X4, respectively. These codenames are in common use, and a good tech should recognize these names—plus they make a dandy way to learn about what's taking place in the CPU business.

AMD Athlon

Athlon is the brand name for a series of CPUs released by AMD. A number of different CPUs have been launched under this brand to compete head to head against the latest Intel chips. The original Athlon, now referred to as *Athlon Classic*, was the first AMD CPU to drop any attempt at pin compatibility with Intel chips. Instead, AMD decided to make its own AMD-only slots and sockets. The first of these sockets used an SEC package and was called Slot A (Figure 5-43).

AMD then shifted back to a PGA package with the release of an Athlon CPU codenamed Thunderbird (Figure 5-44). This CPU, along with several Athlon CPUs to come, used a proprietary 462-pin socket called Socket A. Thunderbird (like its predecessor) had an interesting double-pumped frontside bus that doubled the data rate without

Figure 5-43
Early Athlon CPU

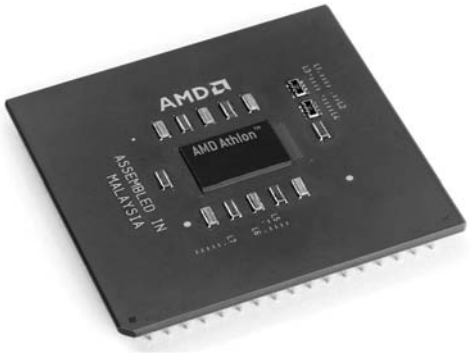


Figure 5-44
Athlon Thunderbird (*photo courtesy of AMD*)

increasing the clock speed. The Athlon Thunderbird CPUs have a smaller but far more powerful L2 cache, as well as a number of other minor improvements when compared to the Athlon Classic.

Next, AMD launched the Athlon XP, first under the codename Palomino. AMD incorporated a number of performance enhancements to the Athlon core, including support for Intel's SSE instructions. AMD released an update to Palomino, codenamed Thoroughbred, which featured increased external bus speeds and was manufactured by using a 130-nm process that reduced the required CPU wattage. The last 32-bit CPUs to wear the Athlon badge were codenamed Barton and Thorton. Barton featured double the L2 cache of Thoroughbred and had an even faster external bus. Thorton was a cheaper version that did not increase the size of the cache.

One interesting aspect of the Athlon XP was AMD's attempt to ignore clock speeds and instead market the CPUs by using a performance rating (PR) number that matched the equivalent power of an Intel Pentium 4 processor. For example, the Athlon XP 1800+ actually ran at 1.6 GHz, but AMD claimed it processed as fast as or faster than a Pentium 4 1.8 GHz—ergo "1800+."

AMD Duron

Duron is the generic name given to lower-end CPUs based on the Athlon processor. A Duron is basically an Athlon with a smaller cache. Because the Duron supported the

same frontside bus as the Athlon, it had a slight performance edge over its low-end rival from Intel at the time. The Duron connected to the same 462-pin Socket A as the later Athlon CPUs (Figure 5-45). AMD discontinued the Duron brand in 2004 and replaced it with the Sempron brand, discussed later in this chapter.

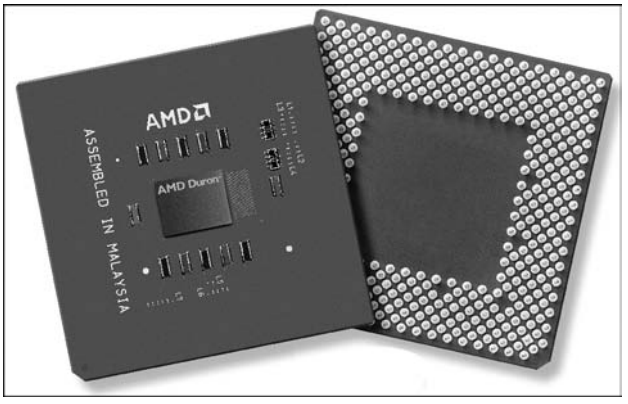


Figure 5-45
AMD Duron
(*photo courtesy of AMD*)

Intel Pentium 4

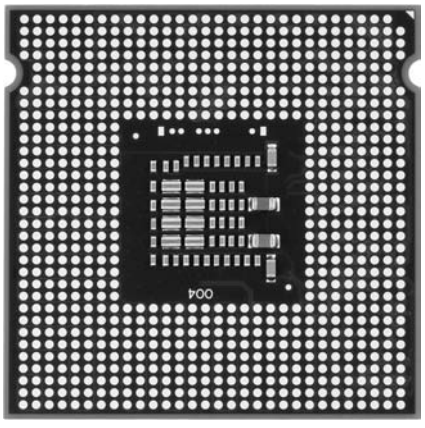
Although the Pentium II and III were little more than improvements on the Intel Pentium Pro, the Pentium 4 introduced a completely redesigned core, called NetBurst. NetBurst centered around a totally new 20-stage pipeline combined with features to support this huge pipeline. Each stage of the pipeline performed fewer operations than typical pipeline stages in earlier processors, allowing Intel to crank up the clock speed for the Pentium 4 CPUs. The first Pentium 4s, codenamed Willamette, included a new version of SSE called SSE2, and later versions introduced SSE3.

The Pentium 4 featured a *quad-pumped frontside bus* where the external data bus was sampled four times per clock cycle. In the early going, there were two packages for the Pentium 4 CPUs. The first Pentium 4 CPUs came in a 423-pin PGA package. This was followed with a 478-pin PGA package (Figure 5-46). Even though the new package has more pins, it is considerably smaller than the earlier package.

Figure 5-46
Pentium 4 (478- and 423-pin)

Intel switched to the Land Grid Array (LGA) 775 package with the release of a Pentium 4 CPU codenamed Prescott (Figure 5-47). Again, even though the LGA 775 package has more pins than a Socket 478 package, it is smaller. With the Pentium 4 CPUs codenamed Northwood and Prescott, Intel unveiled an interesting advancement in superscalar architecture called hyperthreading.

Figure 5-47
Pentium 4 LGA



NOTE P4 Prescotts and Northwoods came in hyperthreaded and non-hyperthreaded versions.

With *hyperthreading*, each individual pipeline can run more than one thread at a time—a tricky act to achieve. A single Intel P4 with hyperthreading looks like two CPUs to the operating system. Figure 5-48 shows the Task Manager in Windows XP on a system running a hyperthreaded Pentium 4. Note how the CPU box is broken into two groups—Windows thinks this one CPU is two CPUs.

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

Starting with the LGA 775 Prescotts, Intel dumped the convention of naming CPUs by their clock speed and adopted a cryptic three-digit model-numbering system. All Prescott Pentium 4s received a three-digit number starting with a 5 or a 6. One of the 2.8-GHz Pentium 4 CPUs is a 521, for example, and one of the 3-GHz processors is called the 630.

A late version of the Pentium 4 CPU was released, called the Pentium 4 Extreme Edition. The Extreme Edition CPUs incorporated a large L3 cache and other architectural details borrowed from Intel's Xeon line of server CPUs. The Pentium 4 Extreme Edition also had some of the highest wattages ever recorded on any Intel desktop CPU—over

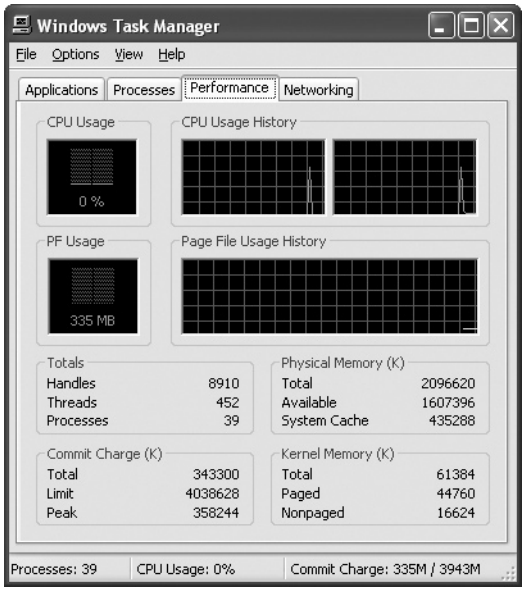


Figure 5-48 Windows Task Manager with the Performance tab displayed for a system running a hyperthreaded Pentium 4

110 watts! Extreme Edition CPUs ran incredibly fast, but their high price kept them from making any significant impact on the market.

These Pentiums reached the apex of clock speeds, approaching 4 GHz. After this, Intel (and AMD) stopped the CPU clock-speed race and instead began to concentrate on parallel and 64-bit processing (both to be discussed later in this chapter).

Mobile Processors

The inside of a laptop PC is a cramped, hot environment where no self-respecting CPU should ever need to operate. Since the mid-1980s, CPU manufacturers have endeavored to make specialized versions of their processors to function in the rugged world

Deciphering the Numbers

Give it up. Intel must have some scheme for their CPU numbering, but it doesn't match the processor speed. They call a 2.66-GHz CPU a 506, for example, which might lead you to believe that the "6" reflects the "66" in the speed. But the 2.8-GHz CPU that followed was named the 511. Go figure!

Here's what Intel says on www.intel.com/products/processor_number/: "Intel processor numbers are based on a variety of features that may include the processor's underlying architecture, cache, Front Side Bus, clock speed, power and other Intel technologies. A processor number represents a broad set of features that can influence overall computing experience, but is not a measurement of performance."

of laptops. Over the years, a number of CPU laptop solutions have appeared. Virtually every CPU made by Intel or AMD has come in a mobile version. You can usually tell a mobile version by the word "mobile" or the letter "M" in its name. Here are a few examples:

- Mobile Intel Pentium III
- Intel Pentium M
- Mobile AMD Athlon 64
- AMD Turion 64 (All Turions are mobile processors but don't have "mobile" or "M" in their name. AMD usually adds "mobile technology" as part of the Turion description.)
- Intel Core Duo (see the "Intel Core" section later in the chapter.)

A mobile processor uses less power than an equivalent desktop model. This provides two advantages. First, the battery in the laptop lasts longer. Second, the CPU runs cooler, and the cooler the CPU, the fewer cooling devices you need.

Almost every mobile processor today runs at a lower voltage than the desktop version of the same CPU. As a result, most mobile CPUs also run at lower speeds—it takes juice if you want the speed! Mobile CPUs usually top out at about 75 percent of the speed of the same CPU's desktop version.

TIP Intel uses the marketing term *Centrino* in the laptop market to define complete mobile solutions, including a mobile processor, support chips, and wireless networking. There is no Centrino CPU, only Centrino solutions that include some type of Intel mobile CPU.

Reducing voltage is a good first step, but making a smart CPU that can use less power in low-demand situations reduces power usage even more. The first manifestation of this was the classic *System Management Mode* (*SMM*). Introduced back in the times of the Intel 80386 processor, SMM provided the CPU with the capability to turn off devices that use a lot of power, such as the monitor or the hard drives. Although originally designed just for laptops, SMM has been replaced with more advanced power-management functions that are now built into all AMD and Intel CPUs.

CPU makers have taken power reduction one step further with *throttling*—the capability of modern CPUs to slow themselves down during low demand times or if the CPU detects that it is getting too hot. Intel's version of throttling is called *SpeedStep*, and AMD's version is known as *PowerNow!*

Early 64-Bit CPUs

Both AMD and Intel now produce 64-bit CPUs. A 64-bit CPU has general-purpose, floating point, and address registers that are 64 bits wide, meaning they can handle 64-bit-wide code in one pass—twice as wide as a 32-bit processor. And they can address much, much more memory.

With the 32-bit address bus of the Pentium and later CPUs, the maximum amount of memory the CPU can address is 2^{32} or 4,294,967,296 bytes. 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}). A 64-bit address bus can address 16 exabytes of RAM.

No 64-bit CPU uses an actual 64-bit address bus. Every 64-bit processor gets its address bus clipped down to something reasonable. The Intel Itanium, for example, only has a 44-bit address bus, for a maximum address space of 2^{44} or 17,592,186,044,416 bytes. AMD's Phenom II, on the other hand, can allow for a 48-bit physical address space for 2^{48} or 281,474,976,710,656 bytes of memory.

Initially, both AMD and Intel raced ahead with competing 64-bit processors. Interestingly, they took very different paths. Let's look at the two CPUs that made the first wave of 64-bit processing: the Intel Itanium and the AMD Opteron.

Intel Itanium (Original and Itanium 2)

Intel made the first strike into the 64-bit world for PCs with the Itanium CPU. The Itanium was more of a proof-of-concept product than one that was going to make Intel any money, but it paved the way for subsequent 64-bit processors. The Itanium had a unique 418-pin *pin array cartridge* (*PAC*) to help house its 2- or 4-MB Level 3 cache (Figure 5-49).

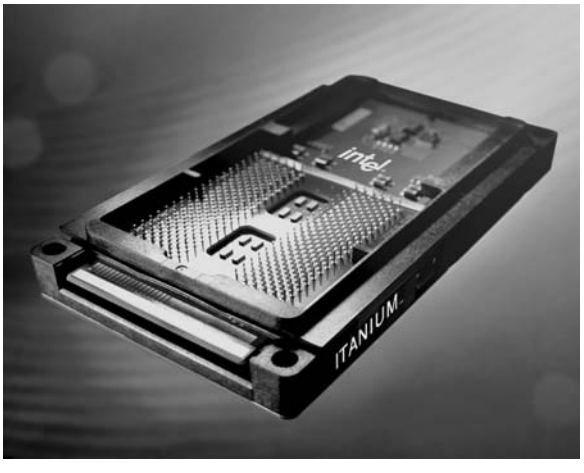


Figure 5-49 Intel Itanium (*photo courtesy of Intel*)

The Intel Itanium 2 was Intel's first serious foray into the 64-bit world. To describe the Itanium 2 in terms of bus sizes and clock speeds is unfair. The power of this processor goes far deeper. Massive pipelines, high-speed caching, and literally hundreds of other improvements make the Itanium 2 a powerful CPU for high-end PCs. The Itanium 2 uses a unique form of PGA that Intel calls *organic land grid array* (*OLGA*) (see Figure 5-50).

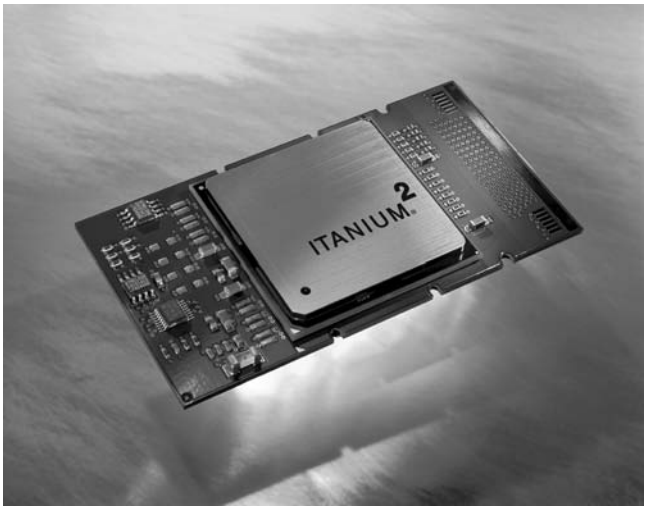


Figure 5-50 Intel Itanium 2 (*photo courtesy of Intel*)

Intel made a bold move with the Itanium and the Itanium 2 by not making them backward-compatible to 32-bit programming. In other words, every OS, every application, and every driver of every device has to be rewritten to work on the Itanium and Itanium 2. In theory, developers would create excellent new applications and devices that dump all of the old stuff (and problems) and thus would be more efficient and streamlined. If a company has a lot invested in 32-bit applications and can't make the jump to 64-bit, Intel continues to offer the Pentium 4 or Pentium Xeon. If you need 64-bit, get an Itanium 2. AMD didn't agree with Intel and made 64-bit processors that also ran 32-bit when needed. Intel would eventually follow AMD in this decision.

AMD Opteron

Coming in after the Itanium, AMD's Opteron doesn't try to take on the Itanium head to head. Instead, AMD presents the Opteron as the lower-end 64-bit CPU. But don't let the moniker "lower-end" fool you. Although the Opteron borrowed heavily from the Athlon, it included an I/O data path known as HyperTransport. Think of HyperTransport as a very high speed link, providing direct connection to other parts of the PC—and to other CPUs for multiprocessing—at a blistering speed of over 6 GB per second! The Opteron comes in a micro-PGA package, looking remarkably like a Pentium 4 (Figure 5-51).

Unlike the Itanium, the Opteron runs both 32-bit and 64-bit code. AMD gives customers the choice to move slowly into 64-bit without purchasing new equipment. This was the crucial difference between AMD and Intel in the early days of 64-bit processing.

Intel and AMD pitch the Itanium 2 and Opteron CPUs at the server market. This means that as a CompTIA A+ tech, you won't see them unless you go to work for a company that has massive computer needs. Newer CPUs from both companies fight for the desktop dollar.

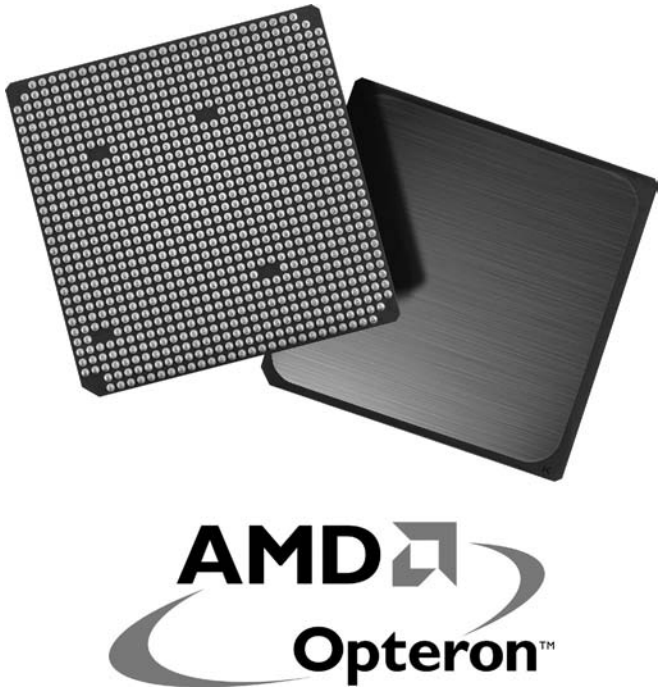


Figure 5-51
AMD Opteron
(*photo courtesy of AMD*)

Athlon 64

To place the Athlon 64 with the early generation CPUs is hardly fair. The Athlon 64 was the first for-the-desktop 64-bit processor, so in that aspect it is an early 64-bit CPU (Figure 5-52). AMD made two lines of Athlons: the "regular" Athlon 64 and the Athlon FX series. The FX series runs faster than the regular Athlon 64s, uses more wattage, and

Figure 5-52
Athlon 64

is marketed to power users who are willing to pay a premium. Underneath those two lines, AMD has almost twenty sub-lines of Athlon 64s in different codenames, making listing all of them here unwieldy.

The AMD 64s have a number of enhancements beyond simply moving into the 64-bit world. The most fascinating is the inclusion of a memory controller into the CPU, eliminating the need for an external MCC and for all intents also eliminating the idea of the frontside bus. The RAM directly connects to the Athlon 64. AMD 64s support Intel's SSE and SSE2 graphics extensions (later versions support SSE3).

While regular Athlon 64s use the same AMD PR numbers to describe CPUs, Athlon 64 FXs use a two-digit model number that's just as cryptic as Intel's current three-digit numbers.

AMD Sempron CPUs

AMD produces various Sempron CPUs for the low end of the market. Semprons come in two socket sizes and have less cache than the Athlon 64, but they offer a reasonable trade-off between price and performance.

Multicore CPUs

CPU clock speeds hit a practical limit of roughly 4 GHz around 2002–2003, motivating the CPU makers to find new ways to get more processing power for CPUs. Although Intel and AMD had different opinions about 64-bit CPUs, both decided at virtually the same time to combine two CPUs into a single chip, creating a *dual-core* architecture. Dual core isn't just two CPUs on the same chip. A dual-core CPU has two execution units—two sets of pipelines—but the two sets of pipelines share caches (how they share caches differs between Intel and AMD) and RAM.

NOTE Putting two or more execution cores onto a single chip is called *multicore*.

A multicore CPU can process more than one thread at a time; this is called *parallel processing*. Through parallel processing, the CPU can more readily juggle the demands of both applications and Windows, making the overall computing experience better. With multithreaded applications (programs written to take advantage of multiple CPUs or CPUs with multiple cores), this parallel processing can dramatically improve the performance of those applications.

Pentium D

Intel won the race for first dual-core processor with the Pentium D line of processors (Figure 5-53). The Pentium D is simply two late-generation Pentium 4s molded onto the same chip, with each CPU using its own cache—although they do share the same front-side bus. One very interesting aspect to the Pentium D is the licensing of AMD's *AMD64*

extensions—the "smarts" inside AMD CPUs that enable AMD CPUs to run either 64- or 32-bit code. Intel named their version *EM64T*. There are two codenames for Pentium D processors: the Smithfield (model numbers 8*xx*), using a 90-nm process, and the Presler (model numbers 9*xx*), using a 65-nm process. Pentium Ds use the same LGA 775 package seen on the later Pentium 4s.

Figure 5-53 Pentium D (*photo courtesy of Intel*)

Athlon Dual Cores

AMD's introduction to dual core came with the Athlon 64 X2 CPUs. The X2s are truly two separate cores that share L1 caches, unlike the Intel Pentium D. Athlon 64 X2s initially came in both "regular" and FX versions packaged in the well-known AMD Socket 939. To upgrade from a regular Athlon 64 to an Athlon 64 X2, assuming you have a Socket 939 motherboard, is often as easy as simply doing a minor motherboard update, called *flashing the BIOS*. Chapter 7, "BIOS and CMOS," goes through this process in detail, or you can simply check your motherboard manufacturer's Web site for the information on the process. In 2006, AMD announced the *Socket AM2*, designed to replace the Socket 939 across the Athlon line.

Intel Core

Intel introduced the *Intel Core* CPUs in 2006. Intel then followed up with the Core 2 processors, the first generation of CPUs to use the Intel Core architecture. Are you confused yet? Let's look a little closer at the Core and Core 2 CPUs.

Intel Core

Intel based the first generation of core processors, simply called Core, on the 32-bit-only Pentium M platform. Like the Pentium M, Core processors don't use the NetBurst architecture, instead falling back to a more Pentium Pro–style architecture (codenamed Yonah) with a 12-stage pipeline. Core CPUs come in single- (Solo) and dual-core (Duo) versions, but they all use the same 478-pin FCPGA package. Core also dispenses with the three-digit Pentium numbering system, using instead a letter followed by four numbers, such as T2300.

Intel Core 2

With the Core 2 line of processors (Figure 5-54), Intel released a radically revised processor architecture called Core. Redesigned to maximize efficiency, the Core 2 processors offer up to 40 percent in energy savings at the same performance level compared to the Pentium D processors. To achieve the efficiency, Intel cranked up the cache size

Figure 5-54
Intel Core 2 CPU

(to 2 or 4 MB) and went with a wide, short pipeline. The CPU can perform multiple actions in a single clock cycle and, in the process, run circles around the competition.

NOTE Intel's naming conventions can leave a lot to be desired. Note that the Core Solo and Core Duo processors were based on the Pentium M architecture. The Core 2 processors are based on the Core architecture.

Intel released three Core 2 versions for the desktop: the Core 2 Solo, Core 2 Duo, and Core 2 Quad. Intel has also released an enthusiast version named the Core 2 Extreme that comes in both Duo and Quad configurations. The Core 2 line also includes mobile versions. All versions incorporate AMD's 64-bit technology, rebranding it as Intel 64, so they can run 64-bit versions of Windows natively.

AMD Phenom

To achieve a quad-core processor, AMD took a different approach than Intel. Intel's Core 2 Quad series of processors combine two dual-core processors, each with their own caches, on the same physical die. These two on-die chips share a frontside bus to communicate with each other and memory.

For AMD's first quad-core desktop processor, called the Phenom, AMD decided to have each CPU core possess its own L1 and L2 caches, but have all four cores share an L3 cache to facilitate communication between cores. This is why AMD refers to the Phenom as a native quad-core processor. The Phenom series of processors are all 64-bit CPUs that feature the AMD64 technology also found in the Athlon 64 CPUs.

The Phenom series processors have an integrated memory controller that supports two channels of DDR2 memory. With the inclusion of an integrated memory controller, the Phenom processors do not possess a traditional frontside bus. Instead, they use the same HyperTransport bus that the Athlon 64 / Opteron series of processors have. Phenom processors are supported by AMD's Socket AM2+ or Socket AM3. The CPU may also use Socket AM2, but this may incur a performance penalty.

AMD refers to a Phenom processor with four cores as a Phenom X4. A Phenom X3 also exists and, as the name suggests, it possesses only three cores. In reality, the Phenom X3 is a quad-core processor with one of the cores shut off due to a defect. As such, AMD sells the Phenom X3 processors at a discount compared to the Phenom X4.

Similar to Intel's Extreme Edition CPUs for enthusiasts, AMD offers versions of the Phenom dubbed Black Edition. Not only are the Black Edition CPUs in the higher-end of their CPU range, but they also feature an unlocked clock multiplier allowing for fine-tuned overclocking.

AMD Phenom II

The Phenom II is a revision of the Phenom with a few improvements (see Figure 5-55). It includes triple the amount of L3 cache as the original Phenom, support for Intel's SSE4a instructions, increased HyperTransport bus speeds, and an enhanced memory controller that can support two channels of DDR2 or DDR3 memory.

Figure 5-55
AMD Phenom II

The Phenom II is built using a 45-nm process instead of the 65-nm process used with the original Phenom. The Phenom II is supported by Socket AM3 and AM2+; however, the CPU only supports DDR3 memory when using Socket AM3.

Like the Phenom before it, the Phenom II is available in both an X4 quad-core version and an X3 triple-core version. But unlike the Phenom, the Phenom II is also available in an X2 dual-core version. Enthusiast Black Editions of all Phenom II configurations are also available.

Intel Core i7

Intel's Core i7 family of processors is based off of a new microarchitecture called Nehalem, which succeeds Intel's Core microarchitecture. Like AMD's Phenom series of processors, the Core i7 is a native quad-core processor and all four cores share an L3 cache. This processor is Intel's first to feature an integrated memory controller. The memory controller supports up to three channels of DDR3 memory.

The Core i7 also marks the return of hyperthreading to Intel's CPUs (see Figure 5-56). With hyperthreading, each individual core can support two simultaneous threads, which, in aggregate, allows a single Core i7 to support up to eight simultaneous threads.

The processor features a large 8-MB L3 cache and was first manufactured using a 45-nm process. Intel designed a new 1366-pin socket for the Core i7 family of processors, called LGA 1366. Like the Core 2 Quad before it, the Core i7 is a 64-bit CPU conforming to the Intel-64 standard, meaning that it can also run 32-bit code.

Figure 5-56
Intel Core i7

With Nehalem, Intel has done away with the traditional frontside bus and replaced it with a technology they named *QuickPath Interconnect (QPI)*. QPI is very similar to the HyperTransport bus found on the Opteron and Phenom series of processors.

Intel Celeron

Intel uses the brand Celeron for its entire family of lower-end CPUs. There are Celerons based on the Pentium II, Pentium III, Pentium 4, Pentium-M, Core and Core 2 Duo. The first Celerons were SEC, but lacked the protective covering of the Pentium II. Intel calls this the *single edge processor* (*SEP*) package (Figure 5-57). The Pentium III–based Celerons were PGA and used Socket 370 (Figure 5-58).

Celeron processors based on the Pentium 4 appeared first using Socket 478, but LGA 775 versions were eventually released. Over time, these Pentium 4-based Celerons borrowed more and more advanced features from their desktop counterparts, including support for SSE3 extensions and Intel-64 addressing.

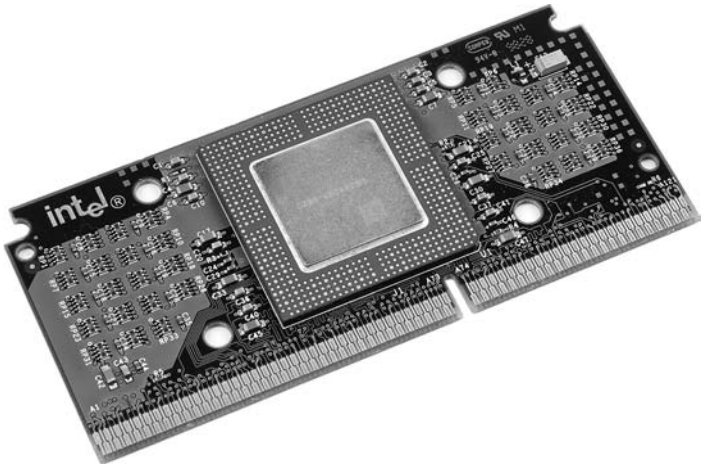


Figure 5-57
Pentium II Celeron

Processors designed to rival AMD's Sempron series chips were also released under the Celeron brand. At first, these chips were based on Intel's Yonah architecture, but Intel released Celerons based on the Core microarchitecture shortly thereafter. Many processors from the latest generation of Celerons are available in dual-core models.

Figure 5-58
Intel Celeron

An entire line of Celerons for mobile computers also exists. Traditionally these chips went by the names Mobile Celeron or Celeron-M. However, Intel dropped that nomenclature with the latest mobile releases, simply referring to them as Celeron.

Intel Pentium Dual-Core

Intel resurrected the Pentium brand in 2006 with the release of the Pentium Dual-Core. Although similar in name, the Pentium Dual-Core is not the same processor as the Pentium D. The Pentium Dual-Core was originally a 32-bit processor based off the Yonah core, but Intel quickly followed that processor up with a 64-bit Pentium Dual-Core based off of the Core microarchitecture.

The Pentium Dual-Core line contains both mobile and desktop processors, and recently, just as with the Celeron brand, Intel has started referring to them simply as Pentium. These Pentiums constitute a middle-of-the-road series of processors for Intel, more powerful than the Celeron line but less powerful than the Core 2 line.

Intel Xeon Processors

Just as the term Celeron describes a series of lower-end processors, the term Xeon (pronounced "Zee-on") defines a series of high-end processors Intel built around the P6, NetBurst, Core, and Nehalem microarchitectures. Both the Pentium II Xeon and the Pentium III Xeon used a unique SEC package that snapped into a Xeon-only slot called Slot 2 (Figure 5-59). With the release of the Xeon based off of the Pentium 4, however, Intel moved to PGA packaging, such as the Xeon-only 603-pin package depicted in Figure 5-60.

Figure 5-59
Intel Pentium III Xeon

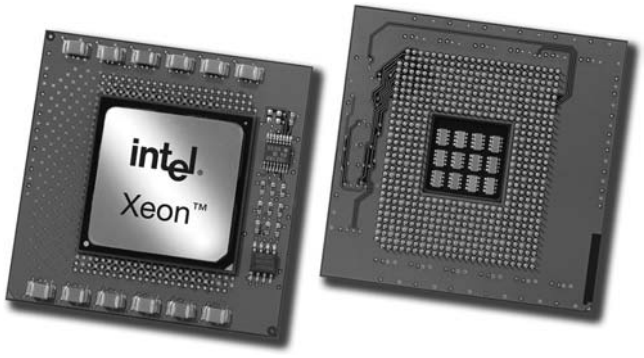


Figure 5-60 Intel Pentium 4-based Xeon (*photo courtesy of Intel*)

Xeon processors add large L2 caches and, especially with more recent Xeons, L3 caches as well. Although a few Xeon processors can only work alone, most are carefully designed to work together in sets of two, four, or more. Today's Xeon processors themselves can contain 2, 4, and even 8 CPU cores per package. Put those two elements together, and modern Xeon systems can contain 32 CPU cores, each with hyperthreading, to support 64 simultaneous threads.

Intel and their partners are working on delivering configurations that will allow for even more CPUs, facilitating even greater processing power. Although expensive, the Xeon's immense power lets them enjoy broad popularity in the high-horsepower world of server systems.

Installing CPUs

Installing or upgrading a CPU is a remarkably straightforward process. You take off the fan and heat-sink assembly, remove the CPU, put a new CPU in, and snap the fan and heat-sink assembly back on. The trick to installing or replacing a CPU begins with two important questions: Do you need to replace your CPU? What CPU can you put in the computer?

Why Replace a CPU?

The CPU is the brain of your system, so it seems a natural assumption that taking out an old, slow CPU and replacing it with some new, fast CPU will make your computer run faster. No doubt it will, but first you need to consider a few issues, such as cost, cooling, and performance.

Cost

If you have an older CPU, there's a better than average chance that a faster version of your CPU is no longer available for retail purchase. In that case, replacing your CPU with a new one would require you to replace the motherboard and probably the RAM too. This is doable, but does it make sense in terms of cost? How much would this upgrade compare to a whole new system?

Cooling

Faster CPUs run hotter than slower ones. If you get a new CPU, you will almost certainly need a new CPU cooler to dissipate the heat generated by the more powerful processor. In addition, you may discover that your case fans are not sufficient, causing the CPU to overheat and the system to lock up. You can add improved cooling, but it might require a new case.

Performance

A faster CPU will make your computer run faster, but by how much? The results are often disappointing. As you go through this book, you will discover many other areas where upgrading might make a much stronger impact on your system's performance.

Determining the Right CPU

So you go through all of the decision-making and decide to go for a new CPU. Perhaps you're building a brand new system or maybe you're ready to go for that CPU upgrade. The single most important bit of documentation is called the motherboard book (Figure 5-61). Every computer should come with this important book that contains all of the details about what CPUs you can use as well as any special considerations for installing a CPU. Usually in the first few pages, the motherboard book will tell you exactly which CPUs your system can handle (as shown in Figure 5-62).

Figure 5-61
Sample motherboard books

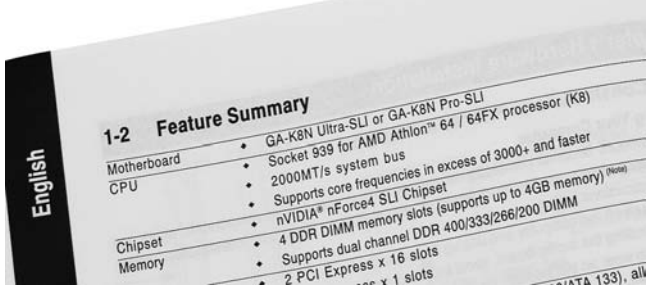


Figure 5-62
Allowed CPUs

If you don't have a motherboard book, call the place where you bought the PC and ask for it. If they don't have it, get online and find it—I'll show you where to look in later chapters.

Your first concern is the socket. You can't install an Athlon 64 X2 into a Pentium D's Socket 775—it won't fit! If your motherboard book lists the CPU you want to install, you're ready to start shopping.

Buying a CPU

Buying a CPU is a tricky game because most stores will not accept returns unless the CPU is bad. If you're not careful, you could get stuck with a useless CPU. Here are a few tricks.

CPUs come packaged two ways, as retail-boxed CPUs or OEM CPUs. Retail-boxed CPUs have two advantages. First, they are the genuine article. There are a surprising number of illegal CPUs on the market. Second, they come with a fan and heat sink that is rated to work with that CPU.

Most stores have an installation deal and will install a new CPU for very cheap. I take advantage of this sometimes, even though it may mean I don't have my PC for a few days. Why does your humble author, the Alpha Geek, have others do work he can do himself? Well, that way I'm not out of luck if there is a problem. Heck, I can change my own oil in my car, but I let others do that, too.

If you buy an OEM CPU, you will need the right fan and heat-sink assembly. See "The Art of Cooling" section later in this chapter.

Preparing to Install

Once you're comfortable that your new CPU will work with your motherboard, get back to that motherboard book and see if you must adjust any tiny jumpers or switches for your CPU. These jumpers might adjust the motherboard speed, the multiplier, or the voltage. Take your time, read the motherboard book, and set those jumpers or switches properly. Locate the fan power connector, usually called the CPU fan, as shown in Figure 5-63.

NOTE Many motherboards have no jumpers or switches.

Most CPUs use some form of mounting bracket for the CPU cooler. Some of these brackets require mounting underneath the motherboard, which means removing the motherboard from the system case.

CAUTION Before attempting to do anything inside the system unit, make sure you have adequate ESD protection. Make sure the power is off and the system is unplugged.

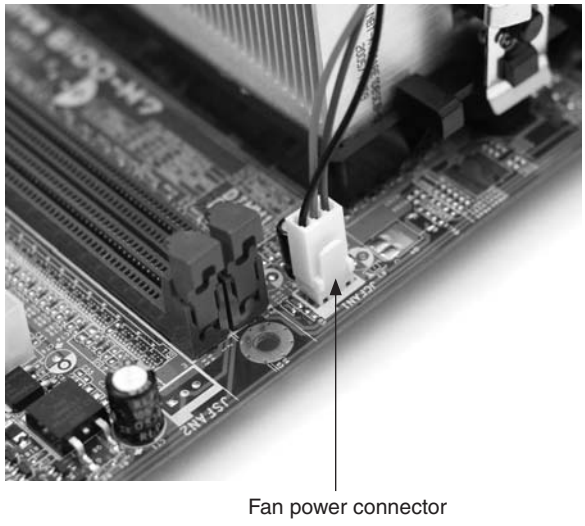


Figure 5-63
Fan connection

If you're removing an old CPU, you'll need to take off the old CPU cooler. Removing CPU coolers scares me more than any other physical act I do on a PC. Many (not all) CPU fans use a metal clamp on both sides of the socket. These clamps usually require you to pry them off to remove them, using a flat-head screwdriver (Figure 5-64). You need a lot of force—usually far more than you think you should use, so take your time

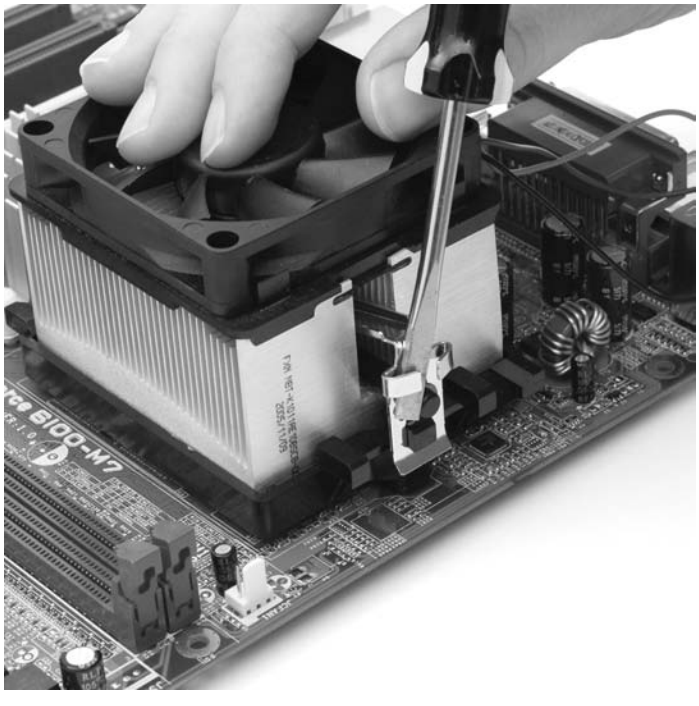


Figure 5-64
Removing an old fan

to pry that old fan off. Don't let the screwdriver slip; you could damage some fragile components on the motherboard, rendering the motherboard inoperable.

CAUTION Using a new fan when you replace a CPU is a good idea—even if the old fan works with your new CPU. Fans get old and die too.

Inserting a PGA-Type CPU

Inserting and removing PGA CPUs is a relatively simple process; just *don't touch the pins* or you might destroy the CPU. Figure 5-65 shows a technician installing a Sempron into a Socket 754. Note that the pins on the CPU only fit in one orientation. These *orientation markers* are designed to help you align the CPU correctly. Although the orientation markers make it difficult to install a CPU improperly, be careful: Incorrectly installing your CPU will almost certainly destroy the CPU or the motherboard, or both!

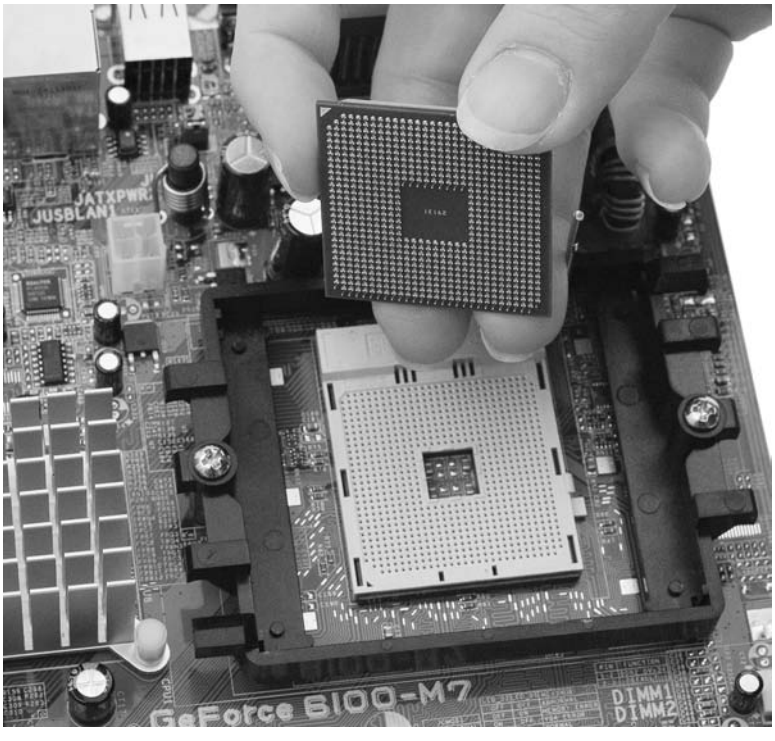


Figure 5-65 Orienting the CPU

To install, first lift the arm or open the metal cover. Align the CPU, and it should drop right in (Figure 5-66). If it doesn't, verify your alignment and check for bent pins on the CPU. If you encounter a slightly bent pin, try a mechanical pencil that takes

Figure 5-66
CPU inserted

thick (0.9mm) lead. Take the lead out of the mechanical pencil, slide the pencil tip over the bent pin, and straighten it out. Be careful! A broken CPU pin ruins the CPU. Make sure the CPU is all the way in (no visible pins), and snap down the arm or drop over the metal cover.

Now it's time for the CPU cooler. Before inserting the heat sink, you need to add a small amount of *thermal compound* (also called *heat dope*). Many coolers come with heat-sink compound already on them; the heat-sink compound on these pre-doped coolers is covered by a small square of tape—take the tape off before you snap down the fan. If you need to put heat dope on from a tube, know that it only takes a tiny amount of this compound (see Figure 5-67). Spread it on as thinly, completely, and evenly as you can. Unlike so many other things in life, you *can* have too much heat dope!

Securing heat sinks makes even the most jaded PC technician a little nervous (Figure 5-68). In most cases, you must apply a fairly strong amount of force to snap the heat sink into place—far more than you might think. Also, make certain that the CPU cooler you install works with your CPU package.

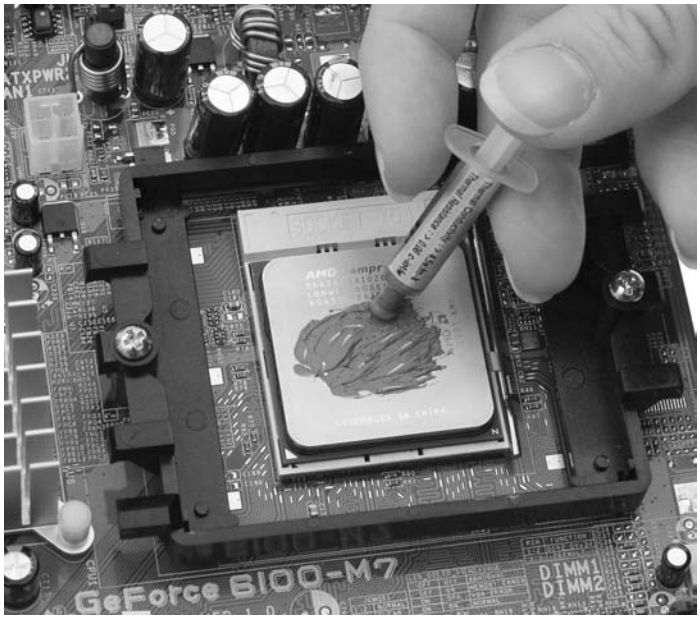


Figure 5-67
Applying thermal compound

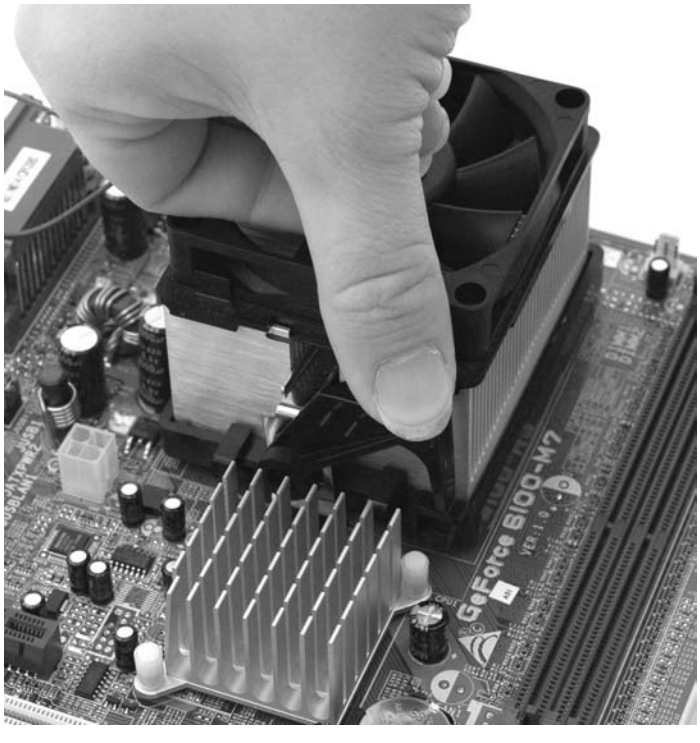


Figure 5-68
Installing the fan

Testing Your New CPU

The next step is to turn on the PC and see if the system boots up. If life were perfect, every CPU installation would end right here as you watch the system happily boot up. Unfortunately, the reality is that sometimes nothing happens when you press the power button. Here's what to do if this happens.

First, make sure the system has power—we'll be going through lots of power issues throughout the book. Second, make sure the CPU is firmly pressed down into the socket. Get your head down and look at the mounted CPU from the side—do you see any of the CPU's wires showing? Does the CPU look unlevel in its mount? If so, reinstall the CPU. If the system still does not boot, double-check any jumper settings—messing them up is very easy.

As the computer starts, make sure the CPU fan is spinning within a few seconds. If it doesn't spin up instantly, that's okay, but it must start within about 30 seconds at the least.

The Art of Cooling

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

- **OEM CPU Coolers** OEM heat-sink and fan assemblies are included with a retail-boxed CPU. OEM CPUs, on the other hand, don't normally come bundled with CPU coolers. Crazy, isn't it? OEM CPU coolers have one big advantage: you know absolutely they will work with your CPU.
- **Specialized CPU Coolers** Lots of companies sell third-party heat sinks and fans for a variety of CPUs. These usually exceed the OEM heat sinks in the amount of heat they dissipate. These CPU coolers invariably come with eye-catching designs to look really cool inside your system—some are even lighted (see Figure 5-69).

Figure 5-69 Cool retail heat sink

The last choice is the most impressive of all: liquid cooling! That's right, you can put a little liquid-cooling system right inside your PC case. 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 5-70 shows a typical liquid-cooled CPU.

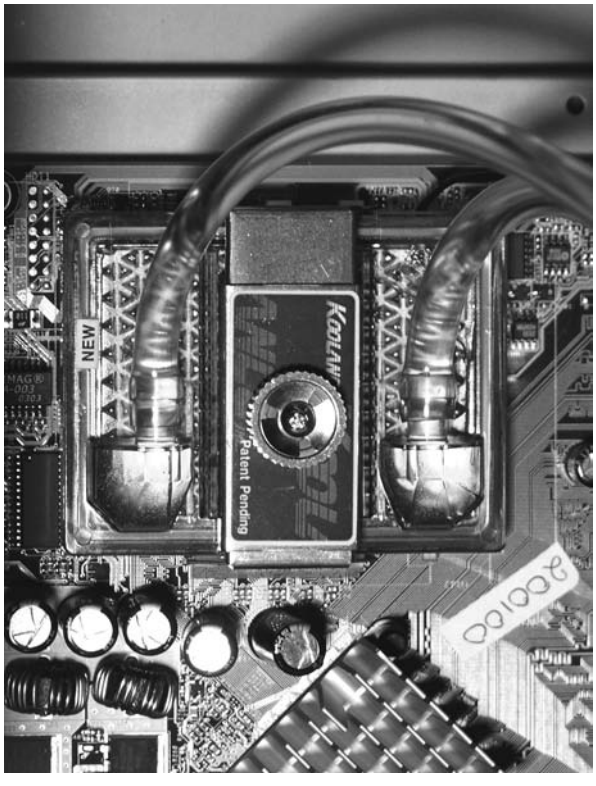


Figure 5-70 Liquid-cooled CPU

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

Whether you have a silent or noisy cooling system for your CPU, always remember to keep everything clean. Once a month or so, take a can of compressed air and clean dust off the fan or radiator. CPUs are very susceptible to heat; a poorly working fan can create all sorts of problems, such as system lockups, spontaneous reboots, and more.

EXAM TIP CPUs are *thermally sensitive devices*—keep those fans clean!

Know Your CPUs

In this chapter, you have seen the basic components and functions of a PC's CPU. A historical view has been provided to help you better understand the amazing evolution of CPUs in the more than 20-year life span of the personal computer.

The information in this chapter will be referred to again and again throughout the book. Take the time to memorize certain facts, such as the size of the various caches, CPU speeds, and clock-doubling features. Good technicians can spout off these facts without having to refer to a book.

Beyond A+

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.

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

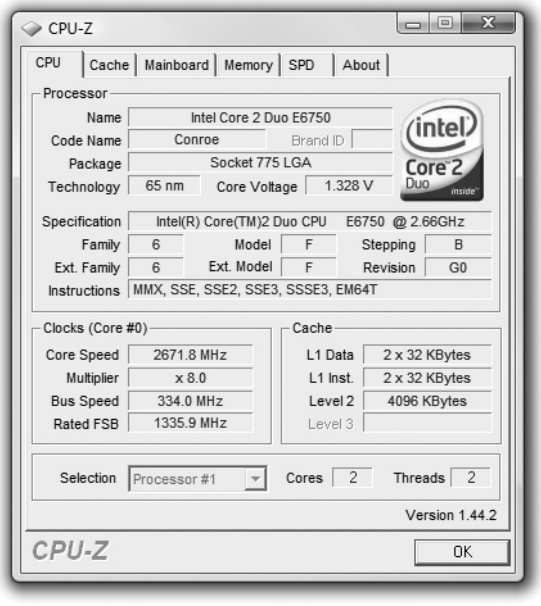
Before I say anything else, I must warn you that intentional overclocking of a CPU immediately voids any warranty. Overclocking has been known to destroy CPUs. Overclocking might make your system unstable and prone to lockups and reboots. I neither applaud nor decry the practice of overclocking. My goal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

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

I don't think Intel or AMD really care too much what *end users* do with their CPUs. You own it; you take the risks. A number of criminals, however, learned to make a good business of re-marking CPUs with higher than rated speeds and selling them as legitimate CPUs. These counterfeit CPUs have created a nightmare where unsuspecting retailers and end users have been given overclocked CPUs. When they run into trouble, they innocently ask for warranty support, only to discover that their CPU is counterfeit and the warranty is void.

If you want to know exactly what type of CPU you're running, download a copy of the very popular and free CPU-Z utility from www.cpuid.com. CPU-Z gives you every piece of information you'll ever want to know about your CPU (Figure 5-71).

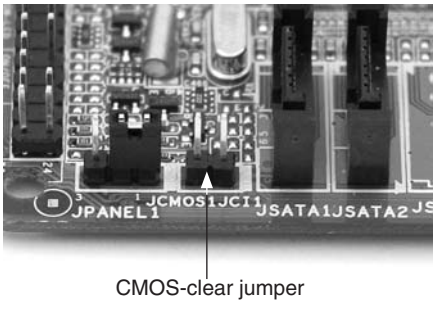
Figure 5-71 CPU-Z in action



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.

Overriding the defaults can completely lock up your system, to the point where even removing and reinstalling the CPU doesn't bring the motherboard back to life. (There's also a slight risk of toasting the processor, although all modern processors have circuitry that shuts them down quickly before they overheat.) Most motherboards have a jumper setting called *CMOS clear* (Figure 5-72) that makes the CMOS go back to default settings.

Figure 5-72 CMOS-clear jumper



Before you try overclocking on a modern system, find the CMOS-clear jumper and make sure you know how to use it! Hint: Look in the motherboard manual.

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

Intel Atom

Intel's Atom processors are very power-efficient processors designed for applications such as ultra mobile PCs, mobile Internet devices, netbooks, and low-power desktops. The Atom range of CPUs consists of both 32-bit and 64-bit models; however, only the models aimed at the low-power desktop segment support 64-bit so far. Many Atom processors also support hyperthreading, and there are now several dual-core models. Figure 5-73 shows an atom processor.

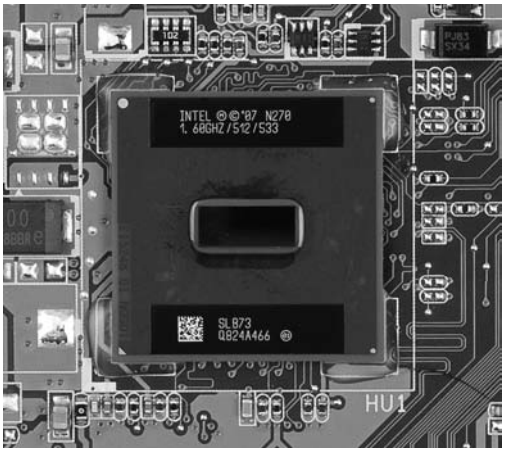


Figure 5-73 Intel Atom processors

Atom processors support Intel's SSE3 instructions, but they do not support SSE4. As of this publication, Intel Atom processors have only been released in a package that is soldered directly to the motherboard. Atom processors are manufactured using a 45nm-process, and many feature Intel's SpeedStep Technology to further reduce their power consumption. The Atom line of processors has become extremely popular for use in netbooks, where heat and power consumption are a primary concern.

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. What is the size of the data bus and the L1 cache on a Core 2 Duo CPU?
   - **A.** 32-bit data bus, 32-KB L1 cache
   - **B.** 64-bit data bus, two 32-KB L1 caches
   - **C.** 64-bit data bus, two 64-KB L1 caches
   - **D.** 128-bit data bus, two 128-KB L1 caches
5. When a tech adds a new Athlon 64 X2 processor to a motherboard, which of the following should the tech check? (Choose the *best* answer.)
   - **A.** Clock speed of the CPU, clock multiplier for the CPU
   - **B.** Clock speed of the CPU, clock multiplier for the CPU, voltage settings on the motherboard
   - **C.** Clock speed of the CPU, clock multiplier for the CPU, voltage settings on the motherboard, system clock speed
   - **D.** Voltage settings on the motherboard, system clock speed

6. The Intel Core Duo has which of the following advantages over the Pentium M?
   - **A.** Level 1 cache
   - **B.** 64-bit data bus
   - **C.** Quad pipelining
   - **D.** Dual processors
7. The Intel Core Duo processor has two 32-KB Level 1 caches, whereas the Athlon 64 X2 has one 128-KB Level 1 cache. What do caches provide for the processors?
   - **A.** They enable the CPU to continue working during pipeline stalls.
   - **B.** They enable the CPU to continue working during hard drive refreshes.
   - **C.** They enable the CPU to access RAM.
   - **D.** They enable the CPU to access the chipset memory controller.
8. What distinguishes the Athlon 64 from the original Athlon? (Select two.)
   - **A.** Athlon 64 uses an external memory controller.
   - **B.** Athlon 64 uses an internal memory controller.
   - **C.** Athlon 64 runs on a traditional frontside bus.
   - **D.** Athlon 64 runs on a HyperTransport.
9. Jane, the hardware technician for a nonprofit corporation, has 10 systems that she needs to upgrade with new microprocessors. Each system currently has a Socket 939 motherboard with an Athlon 64 3200+ CPU installed. To keep the upgrade costs low, her boss has told her to use the existing motherboards if possible.
   - **Primary objective** Upgrade the systems with faster CPUs.
   - **Optional objectives** Use existing motherboards and avoid adding any hardware aside from the CPU, fan, and heat-sink assembly.
   - **Proposed solution** Jane places an order for 10 PPGA Core 2 Duo processors and 10 PPGA to Socket 939 converters.

   The proposed solution:
   - **A.** Meets only the primary objective
   - **B.** Meets the primary objective and one of the optional objectives
   - **C.** Meets the primary objective and both of the optional objectives
   - **D.** Meets none of the objectives
10. A donor gives five Socket 939 Athlon X2 processors to the nonprofit corporation for which Jane works as a technician. Jane has several systems with Socket 939 motherboards and Athlon 64 CPUs installed. Her boss wants her to upgrade five systems with the new processors, but to keep the upgrade costs low, if possible, by using the existing motherboards.

- **Primary objective** Upgrade the systems with faster CPUs.
- **Optional objective** Use existing motherboards.
- **Proposed solution** Remove the Athlon 64 CPUs on five systems and replace them with the new Athlon 64 X2 CPUs. Update the motherboards, following the motherboard manufacturer's guidelines.

The proposed solution:

A. Meets only the primary objective

B. Meets only the optional objective

C. Meets the primary and optional objectives

D. Meets neither objective

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. **C.** The Core 2 Duo CPU has a 64-bit data bus and two 64-KB L1 caches.
5. **C.** Even with the CPUID built into modern processors, a good tech should check the motherboard settings for speed, clock multiple, and voltage and should also know the speed of the CPU.
6. **D.** The Intel Core Duo has dual processors, a definite advantage over the Pentium M.
7. **A.** Caches enable the CPU to continue working during pipeline stalls.
8. **B, D.** The Athlon 64 used an internal memory controller and ran on HyperTransport.
9. **D.** What was Jane thinking? You can't install Intel processors on AMD-based motherboards. Her proposed solution accomplishes none of the objectives.
10. **C.** That's a little more like it. With a minor motherboard update, Jane can put those Athlon 64 X2 processors to work.