

INSIDE THE LAGACY CPU

By Mark E. Donaldson

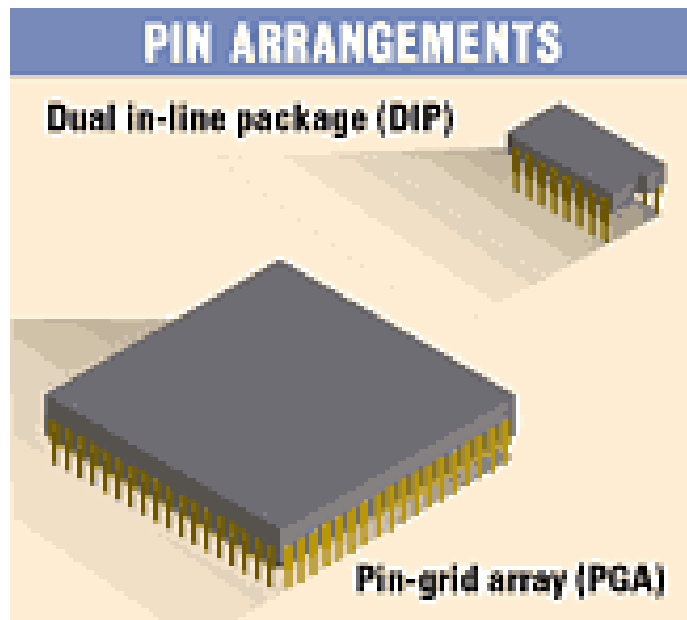
Introduction

The central processing unit (CPU) is the heart of your computer. This vital component, often referred to simply as the microprocessor (or even just processor), is in some way responsible for every single thing your computer does. It determines, at least in part, which operating systems you can use, which software packages are available to you, how much energy your PC uses, and how stable your system will be, among other things. The processor also dictates how much your system will cost: The newer and more powerful the processor, the more expensive the machine.

The Makeup of a Microprocessor

You may think a processor is the square or rectangular piece with many pins that fits into the processor slot on your motherboard, but actually that is just the packaging that contains the processor. The processor itself is a small, thin chip of silicon crystal, typically less than half a square inch in area. The packaging both protects the processor from contaminants (such as the air) and allows it, through the pins, to engage the motherboard's circuits and hence the system as a whole. The millions of electronic switches (the transistors) inside the processor need a carefully controlled environment in which to function.

Although most processors are made of silicon, any semiconductor material will do, as long as it can be fabricated into high-quality pieces of the necessary size. Silicon is widely available and inexpensive because of its ubiquitous use, and it is therefore the most popular material. Silicon works well, because it can form large crystals of uniformly high quality. Each crystal is about 8 inches across, which is important because manufacturers want to cut each crystal into as many chips as possible. Precision saws cut the crystal into slices less than a millimeter thick. These slices, called wafers, are chemically treated before being cut into individual chips. The process for physically applying the logical design of the processor to the chip is called photolithography. In this step, transistors and tiny wires are built onto the chip in a series of ten or more layers (called masks). Once this layering is complete, the chip is tested several times to ensure that the transistors and wires are in place and working properly, and then the chip is placed within its packaging.



The packaging not only protects the chip but also dissipates heat and allows the processor to connect to the motherboard. Over the years packaging has changed considerably, with new methods adopted for various processor designs. The first Intel chips used dual in-line

INSIDE THE LAGACY CPU

By Mark E. Donaldson

packages (DIPs), in which two parallel sets of 40 or more pins provided the connection to the motherboard (see Figure 1).

Because of the parallel design, upgrades to this package could not accommodate significant expansion of connectors: The package would simply get too long for the motherboard as pins were added, and signals from the end pins would require much more time to reach the processor chip than signals from closer pins. For these reasons, the 80286 processor introduced the pin-grid array (PGA) package. This package is typically square, with two, three, or even four rows of evenly spaced pins arranged around a central area. The pins fit into the corresponding holes of the socket module on the motherboard, and typically the package is locked in place by a levered arm.

The square (or squarish) package design that we are most familiar with began with the 80286 and has remained dominant. As the quest for more capable processors grew, wider buses were needed and consequently more pins were required to fit these buses, and many alterations of the package began to appear. Pentium processors use the staggered pin-grid array (SPGA) design, which staggers the arrangement of the pins to allow them to fit closer together. The Pentium Pro, because it has separate chips for the CPU and the Level 2 cache, uses a design called the multichip module (MCM). An MCM is a package that contains more than one chip. Another recent package, the leadless chip carrier (LCC), uses tiny contact pads of gold instead of pins to make contact with the motherboard.

Other packages include the tape-carrier package (TCP), which is as thin as photographic film and is soldered to the motherboard, and the single-edge contact (SEC) cartridge, used for the Pentium II. This is actually a PGA package mounted on a small daughtercard that attaches to the motherboard through a single-edge connector. The SEC is a highly appealing design, because it takes up less space on the motherboard and has better electrical characteristics.

Inside the Processor

Fundamentally all processors do the same thing. They take signals in the form of 0s and 1s (thus binary signals), manipulate them according to a set of instructions, and produce output in the form of 0s and 1s. The voltage on the line at the time a signal is sent determines whether the signal is a 0 or a 1. On a 3.3-volt system, an application of 3.3 volts means that it's a 1, while an application of 0 volts means it's a 0.

Processors work by reacting to an input of 0s and 1s in specific ways and then returning an output based on the decision. The decision itself happens in a circuit called a logic gate, each of which requires at least one transistor, with the inputs and outputs arranged differently by different operations. The fact that today's processors contain millions of transistors offers a clue as to how complex the logic system is.

The processor's logic gates work together to make decisions using Boolean logic, which is based on the algebraic system established by mathematician George Boole. The main Boolean operators are AND, OR, NOT, and NAND (not AND). Many combinations of these

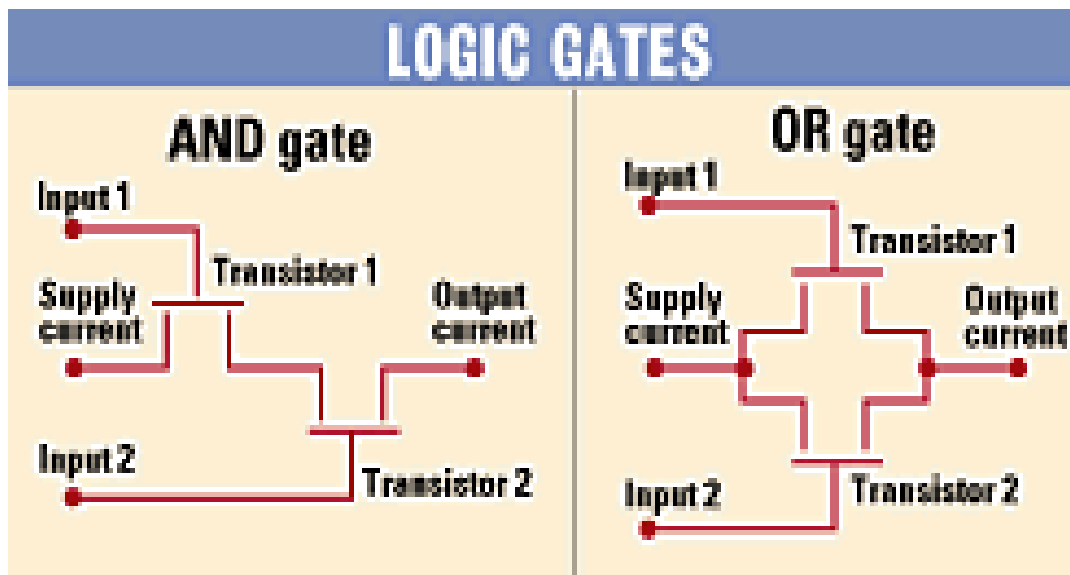
INSIDE THE LAGACY CPU

By Mark E. Donaldson

are possible as well. An AND gate outputs a 1 only if both its inputs were 1s. An OR gate outputs a 1 if at least one of the inputs was a 1. And a NOT gate takes a single input and reverses it, outputting 1 if the input was 0 and vice versa. NAND gates are very popular, because they use only two transistors instead of the three in an AND gate yet provide just as much functionality. In addition, the processor uses gates in combination to perform arithmetic functions; it can also use them to trigger the storage of data in memory.

Logic gates operate via hardware known as a switch, in particular, a digital switch. In the good old days of room-size computers (which looked lots more impressive in movies than today's machines), the switches were actually physical switches, but today nothing moves except the current itself. The most common type of switch in today's computers is a transistor known as a MOSFET (metal-oxide semiconductor field-effect transistor). This kind of transistor performs a simple but crucial function: When voltage is applied to it, it reacts by turning the circuit either on or off. Most PC microprocessors today operate at 3.3V, but earlier processors (up to and including some versions of the Pentium) operated at 5V. With one type of MOSFET, which will be the focus here, an incoming current at or near the high end of the voltage range switches the circuit on, while an incoming current near 0 switches the circuit off.

Millions of MOSFETs act together, according to the instructions from a program, to control the flow of electricity through the logic gates to produce the required result. Again, each logic gate contains one or more transistors, and each transistor must control the current so that the circuit itself will switch from off to on, switch from on to off, or stay in its current state.



AND and OR Logic-Gate Circuits

A quick look at the simple AND and OR logic-gate circuits shows how the circuitry works (see Figure 2). Each of these gates acts on two incoming signals to produce one outgoing signal. Logical AND means that both inputs must be 1 in order for the output to be 1. Logical OR

INSIDE THE LAGACY CPU

By Mark E. Donaldson

means that either input can be 1 to get a result of 1. In the AND gate, both incoming signals must be high-voltage (or a logical 1) for the gate to pass current through itself. Notice in Figure 2 how a high voltage must be applied to both of the transistors in this gate in order for the circuit to be completed. Otherwise the circuit will remain turned off, giving you a logical 0. In the OR gate, as long as either incoming current is high, the gate will allow the current through. Notice in Figure 2 how if a voltage is applied to either transistor, the circuit will be completed.

The flow of electricity through each gate is controlled by that gate's transistor. However, these transistors aren't individual and discrete units. Instead, large numbers of them are manufactured from a single piece of silicon (or other semiconductor material) and linked together without wires or other external materials. These units are called integrated circuits (ICs), and their development basically made the complexity of the microprocessor possible. The integration of circuits didn't stop with the first ICs.

Just as the first ICs connected multiple transistors, multiple ICs became similarly linked, in a process known as large-scale integration (LSI). Eventually such sets of ICs were connected, in a process called (using the industry's deeply creative naming techniques) very large-scale integration (VLSI). Intel's first claim to fame lay in its high-level integration of all the processor's logic gates into a single complex chip. The first processor to do this was the Intel 4004, the forerunner of all of today's Intel offerings. We'll look at the 4004 and its descendants next time.

Two of the most crucial components of the processor are the registers and the system clock. A register is an internal storage area, a unit of memory; and because it is part of the processor, it has the fastest type of memory in your system. Its function is to hold data used by instructions, in the form of bit patterns (sequences of 0s and 1s), in specific places where the processor can find them. The importance of the registers is demonstrated by the fact that processors are identified in one significant way by register size.

The term 16-bit processor refers to a processor with registers capable of holding 16 bits of data. Therefore, 32-bit processors have 32-bit register sizes, and 64-bit processors have double that. The greater the number of bits in a register, the more information the processor can process at once.

The processor spends its time reacting to signals, but it can't react to all of them at the same time or they would become jumbled. Instead, the processor waits until it is given the go-ahead to receive a signal. How long it waits is determined by the system clock. At precise intervals, the system clock sends electrical pulses as a means of polling the system for waiting instructions. If an instruction is waiting and the processor is not already busy with previous instructions, the processor brings the instruction in and works on it. The number of instructions the processor can handle in a single clock cycle (one pulse of the system clock) depends on the design of the processor itself.

INSIDE THE LAGACY CPU

By Mark E. Donaldson

The first microprocessors were able to handle only one instruction per cycle, but today's processors speed this up considerably through two processes, called pipelining and superscalar execution. Pipelining allows the processor to read a new instruction from memory before it is finished processing the current instruction. In some processors, several instructions can be worked on simultaneously. The extent to which pipelined data can flow into the processor is called the pipeline depth. Up through the 80286, Intel processors had a

pipeline depth of only 1 (in effect, there was no pipeline at all), but with the 80486 family, the pipeline depth jumped to 4; up to four instructions could be in different pipeline stages. Pentiums have a pipeline depth of 5, and MMX technology enables even more.

A superscalar processor has more than one pipeline, meaning it can execute more than one set of instructions at the same time. Theoretically this can double performance, but usually one of the pipelines ends up waiting for an instruction to finish in another pipeline.

Instructions

Computers run on low-level commands called instructions. By low-level, we mean that these commands work directly with the processor, in effect communicating with the processor's most basic capabilities. Each type of processor has a specific group of these commands on which it can act. This group is called the processor's instruction set.

The processor's instructions are accessible to human programmers through various programming languages. The instructions themselves are written in machine language, the lowest-level language of all, which consists solely of numbers and thus is rarely used by programmers. To get around this difficulty, programmers turn either to assembly language, which uses the same instructions but gives them names (such as add), or to a high-level language (HLL), in which the machine instructions are encompassed within larger-scale commands. Figure 1 illustrates these language levels.

Revised December 28, 2008

LANGUAGE LEVELS

High-level language

```
PRINT "A"
```

Assembly language

```
mov dx,41h•  
mov ah,2  
int 21h
```

Machine code in hexadecimal•

```
BA4100•  
B402•  
CD21•
```

Machine code in binary

```
101110100100000100000000•  
10110100000000010•  
1100110100100001•
```



Microprocessor



INSIDE THE LAGACY CPU

By Mark E. Donaldson

HLLs don't dispense with machine instructions in any way; they simply make them easier to work with. A program written in an HLL must be compiled, usually first into an internal intermediate language, then to machine language. The two stages, called a compiler front end and back end, allow the compiler writer to separate the parts of the process that are architecture-neutral from those that are architecture-specific. Ultimately, the processor must receive numerical instructions to do anything at all.

Typical instructions for the x86 instruction set, which has formed the basis of the PC environment for years, include commands for such activities as arithmetic functions, data movement, logical instructions, and input/output instructions. Arithmetic instructions include add, which adds the contents of different registers together, and inc (increment), which adds 1 to the value in the register. Data movement instructions include mov, which moves data from one register or memory address to another register or memory address, and xchng (exchange), which swaps the values in two different registers or memory addresses. All programs consist of combinations of the wide variety of instructions available to the processor.

Superscalar Designs

Last time we mentioned pipelining, the technique that allows a processor to start the execution of a new instruction before completing the current one. Pipelining saves time by ensuring that the microprocessor doesn't have to wait for instructions; however, processors can still complete just one instruction per clock cycle. To increase efficiency and thereby save processing time, today's processors (Compaq/Digital's Alpha, IBM/Motorola's PowerPC, Intel's Pentium line, and Sun's SPARC) feature superscalar architecture. The main benefit of superscalar technology is that it allows processors to execute more than one instruction per clock cycle with multiple pipelines.

In a superscalar design, the processor looks for instructions that can be handled within the same clock cycle and processes these together. In the Pentium processor, for example, simple instructions such as mov, or, and add can be processed in this way, although only under specific circumstances (one instruction cannot require the results of a second). But more complex instructions such as those involving floating-point operations can't be handled this way at all.

Parallel processing offers obvious speed benefits, but superscalar technology has critics. Some argue that it wastes many opportunities for parallel execution, because combining individual instructions takes too much time and because individual instructions are often delayed while waiting for resources. For example, say instruction A is being executed from one pipeline and instruction B from another. Instruction C waits in the first pipeline for instruction A to finish. When instruction A finishes, the obvious thing to do is to replace it with instruction C, the next one in that pipeline. But if instruction C needs the results of instruction B, currently being executed from the other pipeline, it has to sit and wait. This defeats the

INSIDE THE LAGACY CPU

By Mark E. Donaldson

attempt at parallel execution and ruins any chance of increased speed. Your expensive new processor is basically twiddling its thumbs.

Even in a well-designed program, one that attempts to make full use of both pipelining and parallel execution, pipelines can get clogged. To help combat this gridlock, engineers have designed superscalar processors to perform out-of-order execution. If a free pipeline has nothing to do because instruction C needs the results of instruction B, the processor can look for the first instruction in the program that doesn't depend on instruction B (instruction H, let's say). It then starts working on instruction H and related instructions until instruction B is finished, at which point it goes back to instruction C. Instead of sending the results of the out-of-order instruction into the registers (where the processor directly deals with data), the processor sends them to a buffer for storage, then sorts everything into the proper order before releasing them.

One possible problem with out-of-order execution is that two instructions may need to use the same register. To compensate for this, today's processors can change the names of registers on the fly, in a process known as register renaming. Clearly, out-of-order execution requires extremely careful processor design, because programs will possibly fail if instructions are not processed in the proper order.

Instruction Sets: RISC and CISC

An instruction set is the specific group of instructions that a particular processor can recognize and execute. Over the years, a debate has raged over two processor-design philosophies surrounding the implementation of instruction sets. The first approach, initially known as microcode, is CISC (complex instruction-set computing). On the other end is RISC (reduced instruction-set computing). Your Nintendo 64 game machine, PowerPC Macintosh, and Silicon Graphics workstation all use RISC technology, while your Intel PC and your 680x0 Macintosh use CISC designs. The actual difference between the two is lessening all the time, but the debate continues.

The first processors used hardware to execute each instruction. These hard-wired processors were extremely fast, since there was no software for the instructions to work through. As you might imagine, though, this approach caused a major problem. Any change to the hardware required a change to the (software) instructions as well, and vice versa. Simple programs were possible, but adding complexity was nearly impossible. To get around this problem, IBM devised microcode, simple software stored on the chip from which the processor obtained its instructions.

One advantage of CISC's microcode was that the instruction set could be modified much more easily than before, and thus increasingly complex instructions were possible. Also, because each instruction replaced several simple hard-wired instructions, programs could be written with a smaller number of instructions overall. Another advantage was that CISC programs took less memory space, and memory was expensive back in the sixties and seventies. However, handling these complex instructions was substantially more work for the

INSIDE THE LAGACY CPU

By Mark E. Donaldson

processor. And different complex instructions required different numbers of clock cycles. Thus microcode had disadvantages, too.

The solution to complexity, of course, is simplicity, and that's precisely what the RISC initiative attempted in the mid-1970s. The basis for RISC was that even with microcode designs, the vast majority of a processor's time was spent executing simple instructions, not complex ones; researchers came to realize that a sequence of simple instructions could often outperform a single complex instruction. In a RISC design, there are fewer instructions, and each is the same length (32 bits) and can be executed in one clock cycle. Microcode disappears for most frequently executed instructions, with RISC processors returning to the premicrocode system of hardware logic. Replacing microcode is the processor's high-speed memory cache, in which sequences of instructions are stored. In a RISC design, programs must be carefully compiled to optimize use of the processor's instruction set, and pipelining must be used to its greatest effect.

As soon as the RISC design appeared, the microcode design was renamed CISC. The most widespread CISC designs, of course, are the Intel family of processors, IBM mainframe processors, and the Motorola 680x0 processors. The most popular instruction set is the x86 instruction set, initially designed for the Intel 8086 processor and still fundamentally in place in today's Pentiums. The most recent addition to the x86 instruction set has been MMX, a set of 57 new instructions that apply primarily to multimedia programming. The x86 instruction set may disappear with Intel's upcoming IA-64 instruction-set architecture. The first implementation of the architecture will be the upcoming 64-bit Merced (P7) processor, incorporating an essentially new instruction set for the first time in the PC's history. Intel has publicly proclaimed that x86 instruction-set support will be included in Merced, but it has not yet specified how the compatibility will be achieved.

The Intel Processor Family

Although the term PC means personal computer, today it's used almost exclusively to mean a machine running an Intel or Intel-compatible processor and a Microsoft operating system (DOS, Windows 95, or Windows NT). This wasn't always the case: Apple II computers used to be called PCs, as did Commodore 64s and any other computer that was smaller than a minicomputer.

But when IBM entered the market back in 1981, it called its machine the IBM-PC, and non-IBM machines capable of running applications written for that machine became known as IBM-PC-compatibles. The term was shortened to IBM-compatible or PC-compatible and soon to just PC. It wasn't always necessary to run a Microsoft operating system, since OS/2-powered machines were also called PCs. These machines shared one thing: the Intel processor. Today you can buy PCs with Intel-compatible processors (and we'll look at those next time), but most PCs still sport Intel chips, which is pretty much the way it's been since the early days.

INSIDE THE LAGACY CPU

By Mark E. Donaldson

4004 through 80286

The first Intel design was the 4004, way back in 1971. The 4004 was a 4-bit processor and could do little but basic arithmetic calculations. A year later, Intel upped the ante with an 8-bit version of the 4004, the 8008, and it ran at roughly 0.2 MHz. A couple years after that, the 8080 came along, offering a more powerful instruction set. The 8080 formed the basis of Zilog's (not Intel's) Z80 processor, for which CP/M--the first personal-computer operating system--was developed. The first serious processor to come from today's hardware giant was the 8086, released in 1978 with a 16-bit data bus. But the 16-bit bus was too expensive at the time, and very little 16-bit hardware existed, including motherboards. So Intel reengineered it with an 8-bit external design and in 1979 released it as the 8088. The registers were still 16 bits in size (like the 8086's), but the 8-bit data bus brought it back into compatibility with the hardware of the late seventies. The 8088 became the heart of the IBM-PC, while the more capable 8086 was relegated to a secondary role in clones. The 8088 was capable of 4.77-MHz and 8-MHz speeds, and the 8086 handled these two speeds and also 10 MHz.

In 1982, Intel released an extremely important processor: the 80286. When this found its way into the spanking-new IBM PC-AT in 1984, we began to see just what PCs might be capable of. It offered a 16-bit data bus, 24-bit addressing, and a 16-bit register size, and its processing efficiency was so advanced over the 8088's that even at its introductory speed of 6 MHz, it performed at well over four times the speed of the 4.77-MHz 8088. The 286, as it came to be known, was soon introduced in speeds of 8 MHz, 10 MHz, and 12 MHz. It was capable of addressing 16MB of memory, which was a real advance over the 8088's 1MB. It also offered full compatibility with its predecessor by including two modes of operation, real and protected mode, the first emulating the 8088 (with that chip's 1MB restriction) and the second allowing protected memory ranges to be isolated from one another to avoid program conflicts. Unfortunately, DOS didn't handle protected mode well, so that option remained largely unused until DOS extenders and then OS/2 were introduced.

The 286 had a significant drawback that shortened its life, however: 286-based PCs booted into real mode and could be switched via software to protected mode, but there was no way to switch back without rebooting. This, of course, limited the 286's usefulness for backward compatibility with real-mode programs, which constituted the vast majority of commercial and noncommercial programs available at the time.

The 80386

The 80286 only hinted at what was possible with the x86 instruction set and the established CISC architecture. When the 80386 was introduced in 1985 (in a Compaq rather than an IBM machine), the personal computing phenomenon began to take shape. The 386 was the first 32-bit Intel processor, after all, with a 32-bit data path and register size. It could address up to 4GB (gigabytes) of memory, and its speeds, ranging from 16 MHz to 33 MHz (AMD and Cyrix 386 clones went all the way to 40 MHz)--made it the fastest Intel processor so far. The first chip to hit the market was simply called the 80386, but soon after that Intel released the 80386SX (more on this later). For a further distinction between the two models, the original

INSIDE THE LAGACY CPU

By Mark E. Donaldson

chip began to be referred to as the 80386DX. The SX and DX distinction carried through to the 80486 family, but not into the Pentium family.

Not only could the 386 address much more memory than the 286, it also allowed memory to be addressed as one large chunk. The 286 handled memory in segments of only 64K, despite having a potential full 16MB to work with. The benefit of addressing memory as a single large chunk is that applications and their data can be loaded into all the memory available on the individual machine, and once loaded they can be accessed much more quickly. The 386 also featured a small chunk (16 bytes) of a built-in instruction queue and the ability to load the data that the processor would require next.

While the 286 operated in two possible modes, the 386 introduced a third. Real mode still allowed full compatibility with programs designed for the 8086, and like the 286, the 386 entered real mode at boot time. Protected mode was also still a feature, and it worked like the 286 chip in this mode. The new mode was called virtual 8086, in which the 386 could emulate multiple 8086 chips, each in an isolated memory space. In this way, the 386 became the first practical multitasking chip in the Intel family. The 286 was able to multitask protected-mode programs, but the operating systems and programs of the time offered little support for that feature.

Early 386DX chips were expensive, and they offered few improvements when used with existing DOS-based software. That's why Intel released the 386SX, a different physical design from the 386DX, able to use simpler motherboard circuitry. The 386SX used a 16-bit rather than a 32-bit data bus, although the processor's registers were 32-bit. The 16-bit bus meant a slowdown, since data couldn't be supplied to the processor to keep up with its capacity.

For users, the major advantage of the 80386 family was that its use of memory and multitasking finally made the development of a PC-based graphical user interface (GUI) possible. Microsoft Windows 3.0 ran on a 286, but slowly and inefficiently. Only with the 386 family--with the ability to address a single large memory chunk, the strong use of virtual memory, and the protected and virtual 8086 modes, did Windows begin to acquire the power it needed.

The Intel 80486 Family

Despite the importance of the 80386 family to the development of PCs, the PC platform as we know it today truly emerged with the introduction of the 486 in 1989. Windows 3.x could run reasonably well on the 80386DX and slowly but decently on the 80386SX, but the 486 brought into the mix a number of new technologies that resulted in faster and smoother performance.

The 486 processor was the first in the Intel family to include a Level 1 cache (8K in size), which meant fewer RAM accesses, and since L1 was incorporated directly into the processor itself, access to the cached data was faster than ever before. The 486 was also the first Intel

INSIDE THE LAGACY CPU

By Mark E. Donaldson

processor to offer burst mode, allowing increased data-transfer speeds between RAM and CPU. The 486 was the first truly pipelined Intel x86 processor, and thus it provided greater instruction throughput. It contained about 1.25 million transistors, five times as many as the 80386, which were dedicated to more advanced functional units, to an on-board floating-point unit (FPU), and to the on-board L1 cache. The FPU could perform calculations on real numbers, those containing decimal fractions--thus improving the performance of certain types of applications (scientific and graphics programs, in particular).

The 486DX was available in clock speeds of 25 MHz, 33 MHz, and 50 MHz. The chip ran cool enough that no heat sink was required, although some systems shipped with one anyway. The chip could address up to 4 gigabytes of RAM, and like the 386, it ran in real, protected, and virtual 8086 modes.

Like the 80386, the 486 came in DX and SX versions. The cheaper SX version was available in 16-MHz, 20-MHz, 25-MHz, and 33-MHz clock speeds, but aside from the slower speeds and one other significant feature, it was practically identical to the 486DX. That one other feature was the lack of a math coprocessor. But even this wasn't actually true: The 486SX was simply a 486DX with the math coprocessor disabled. The DX/SX difference in the 486 family appears to have been little more than a marketing ploy: It was possible to upgrade the 486SX with floating-point capabilities by installing onto the motherboard a chip called the 80487SX, and the 80487SX was, in reality, an 80486DX. When installed, the chip disabled the original 486SX entirely. Strange but true.

Actually, the 487SX wasn't quite a 486DX: Its pin arrangement was different, to the extent that a 486DX proper could not fit into the 487SX slot on the (cheaper) 486SX motherboard. This pin arrangement allowed Intel to market another speed-up processor for the 486SX, the OverDrive chip. Introduced in 1992, the OverDrive chip fit into the slot for the 487SX; it came initially in a version for the 25-MHz 486SX and later in one for the 33-MHz 486SX. The idea was to double the clock speed of the SX original, and hence the chip came to be known as a clock-doubling chip. In the 25-MHz version, the chip maintained the 25-MHz speed when moving data from sources external to the processor, but internal processes occurred at double that speed, 50 MHz. With the 33-MHz version, the internal speed was 66 MHz.

Clock doubling was such a popular idea that in the fall of 1992, Intel decided to release clock-doubled versions of the main DX processor itself, calling them the 80486DX2-50 and the 80486DX2-66. These chips did not supplement the standard 486DX, as the OverDrive chip added to the 486SX, but instead replaced it completely.

Then, in 1994, the company brought out the 80486DX4, which (despite its X4 appellation) ran at triple the DX's clock speed (clock-tripling). Here, external processes still operated at the speed of the bus (25 MHz or 33 MHz), but internal processes were now capable of speeds of 75 MHz (for the 25-MHz DX) and 100 MHz (for the 33-MHz version). These processors used a voltage of 3.3V in order to keep the heat down (the 5V DX2 required heat sinks for heat dissipation), and as a result, they typically required new motherboards.

INSIDE THE LAGACY CPU

By Mark E. Donaldson

AMD and Cyrix

The 486 line of processors saw the first widely available copies, or clones, from other manufacturers. . Both AMD (Advanced Micro Devices) and Cyrix made their own versions of the 486DX, but their products became better known with their 486DX2 clones, one copying the 486DX2-66 and another upping the ante to 80 MHz for internal speed. The 486DX2-80 was based on a 40-MHz system bus, and unlike the Intel DX2 chips (which ran hot at 5V) it ran at the cooler 3.3V. AMD and Cyrix both later introduced clock-tripled versions of their 40-MHz 486 processors, which ran at 120 MHz. Both AMD and Cyrix offered power management ("green") features beginning with their clock-doubled processors, but Intel didn't introduce these features until the DX4. AMD's and cyrix's processors give Intel a run for the money with innovations of their own.

Although Intel stopped improving the 486 with the DX4-100, AMD and Cyrix kept going. In 1995, AMD offered the clock-quadrupled 5x86, a 33-MHz 486DX that ran internally at 133 MHz. AMD marketed the chip as comparable in performance to Intel's new Pentium/75, and thus the company called it the 5x86-75. But it was a 486DX in all respects, including the addition of the 16K Level 1 cache (the cache built into the processor), which Intel had introduced with the DX4. Cyrix followed suit with its own 5x86, called the M1sc, but this chip was much different from AMD's. In fact, the M1sc offered Pentium-like features, even though it was designed for use on 486 motherboards. Running at 100 MHz and 120 MHz, the chip included a 64-bit internal bus, a six-stage pipeline (as opposed to the DX4's five-stage pipeline), and branch-prediction technology to improve the speed of instruction execution. It's important to remember, however, that the Cyrix 5x86 was introduced after Intel rolled out the Pentium, so these features were more useful in upgrading 486s than in pioneering new systems. For the purposes of this article, this chip provides a segue into the world of the Pentium.

The Many Flavors of Pentium

The word pentium doesn't mean anything, but it contains the syllable pent, the Latin root for five. Originally Intel was going to call the Pentium the 80586, in keeping with the chip's 80x86 predecessors. But the company didn't like the idea that AMD, Cyrix, and any other clone makers could use the name 80x86 as well, so Intel decided on a trademarkable name, hence Pentium. Although this new chip was still a CISC-based product, it incorporated a number of RISC technologies into its design, and it was the first superscalar Intel processor. These technologies allowed the chip to execute over 300 MIPS (million instructions per second) by contrast, the much slower DX2-66 executed less than 60 MIPS. The rating of MIPS is hardly an exact science, and the measure refers only to the processor's ability (not the I/O or other factors), but these ballpark figures demonstrate the magnitude of the speed increase.

The Pentium introduced other significant technologies. First, as mentioned, it offered a superscalar architecture: It used two pipelines rather than the 80486's single pipeline, although for best performance, programs had to be optimized so that the pipelines would

INSIDE THE LAGACY CPU

By Mark E. Donaldson

work together. Second, it employed branch-prediction technology to help minimize the delays often incurred when a branch instruction alters the flow of instruction execution. Third, the Pentium increased the speed of data transfer from memory by using a 64-bit data bus instead of the 32-bit bus of the 80486. It sped transfer further by implementing pipeline-burst mode in both reading from and writing to memory, and by incorporating a 66-MHz memory bus (initially a 60-MHz bus); the 486 used a 33-MHz version. Fourth, the Pentium came with built-in power management. Fifth, two separate Level 1 caches, one for data and the other for instructions, allowed programs to be optimized fully in both categories, and a separate floating-point pipeline improved the speed of execution of floating-point instructions (those involving numbers with decimal places).

Perhaps ironically, it was the chip's built-in floating-point capabilities, or more specifically, its floating-point problems, that brought the chip to public awareness. Shortly after the Pentium's introduction, the media jumped on the news of a bug in the FDIV (floating-point divide) instruction. Essentially, if you divided number A by number B and the quotient contained a certain fraction (represented by the digits to the right of the decimal point), in some rare cases you could in turn multiply the quotient by number B and end up with a result that was different from number A (it would be off by 256). Intel quickly offered replacements for the flawed chip, and in fact the company managed to turn the design flaw into a marketing bonanza. It was the first time a microprocessor had ever hit the big time as a news event. (See www.intel.com/procs/support/pentium/fdiv/index.htm to determine if your early Pentium has the FDIV bug.)

The Pentium debuted in March 1993 with the P/60 and the P/66 (as usual, the number represents the chip's speed in MHz), and all future Pentiums--right up to the P/200, introduced in mid-1996--were based on these two. The P/75, P/90, and P/100 used a clock multiplier of 1.5, the P/120 and P/133 clock-doubled the original versions, and the P/200 clock-tripled them. (The P/150 and P/166 used a 2.5 multiplier.) All versions contained over 3 million transistors, and all required heat sinks for heat dissipation.

Meanwhile, AMD and Cyrix capitalized on both the success of the Pentium and the chip's much-publicized FDIV flaw, introducing the 6x86 and the K5 respectively and marketing them as more reliable than the Pentium. Here, the two companies veered away from producing Intel clones and concentrated instead on original designs that, despite their originality, retained compatibility.

The 6x86, a CISC processor, offered two pipelines, each with seven stages, as opposed to the Pentium's five-stage, two-pipeline technology. It improved pipelining by offering the results of instructions to both pipelines at once to reduce stalling, and by including better branch prediction and out-of-order completion (allowing instructions to be executed out of their coded order, as long as they don't rely on results from previous instructions).

The K5, on the other hand, was at the core a RISC processor that translated the x86 instruction set into simpler instructions. It contained six pipeline stages (instead of the

INSIDE THE LAGACY CPU

By Mark E. Donaldson

Pentium's and 6x86's two), it decoupled decode and execute functions, it included six functional units (a branch unit, two load/store units, a floating-point unit, and two arithmetic-logic units), and it included many advanced features similar to those of Cyrix's entry. One central difference was that the K5 was late to market and didn't hit its expected speed grades, and thus it failed to capture significant market share. AMD's follow-on K6 product, derived from its 1996 acquisition of NexGen, has been far more successful.

The MMX Effects

In 1997, Intel introduced the Pentium with MMX, a set of 57 additional instructions designed to improve the multimedia capabilities of the Pentium. These instructions focus on parallel execution and employ a technique called single instruction, multiple data (SIMD) to do their work. As the name suggests, SIMD allows a single instruction to work with more than one piece of data at the same time, thereby allowing the instruction to produce results more quickly. But this wasn't the only change in the MMX-adorned Pentiums. The pipeline increased to six stages from five, the two Level 1 caches were each increased from 8K to 16K, and branch prediction was improved.

The Pentium Pro and Pentium II

More than a year before the Pentium with MMX hit the market, Intel introduced its successor to the Pentium, the Pentium Pro. The Pentium Pro improved on the Pentium in several ways, and in the process it introduced a new way of executing instructions. The internal core is a RISC processor, and the x86 CISC instructions are built from RISC micro-instructions, which are simpler and thus faster to execute (the combined microinstructions are called RISC86 instructions). The Pentium Pro increased the pipelining stages from 5 to 14, with three pipelines rather than two, to achieve significantly greater speed of execution. Furthermore, up to four Pentium Pros could operate simultaneously in a single system, double the multiprocessing capability of Pentium systems.

The Pentium Pro's 5.5 million transistors caused some heat concerns, but a reduction in the size of the transistors themselves helped keep this problem in check. Unfortunately (as PC Magazine was the first to discover), the Pentium Pro did not process certain 16-bit instructions efficiently, and thus it performed no better than a regular Pentium with similar clock speed on Windows 95 and performed worse on Windows 3.1.

One extremely significant change from the Pentium was including a built-in 256K Level 2 cache in addition to the earlier processor's 16K Level 1 caches, a feature that offered obvious speed improvements but, at the same time, greatly increased the cost of manufacturing a chip. This meant that the Pentium Pro never dropped in price to the extent that the Pentium did, and was in part the basis for developing the Pentium II.

In the Pentium II, Intel doubled the Level 1 caches to 32K but replaced the Pentium Pro's Level 2 cache with a larger, 512K cache with its own bus running at only half the speed of the Pentium II. The result was a cheaper processor but one technically not as fast as the Pentium Pro at the same clock speed. In practice, however, the Pentium II runs many of today's

INSIDE THE LAGACY CPU

By Mark E. Donaldson

programs faster than does the Pentium Pro, because the clock speed itself has gone well beyond the Pro's 200 MHz--all the way to (at last count) 400 MHz, with a 100-MHz bus for the first time. You can expect to see a 450-MHz Pentium II later 1998.

As you might expect, AMD and Cyrix worked hard to keep themselves in the picture. AMD released the K6 in 1997, offering a strong, low-cost competitor to the Pentium Pro, the Pentium MMX, and the Pentium II. Like the Pentium MMX, the K6 uses dual voltage, also called split-rail voltage, in which the processor's external voltage is higher than the internal voltage. Typically the former runs on 3.3V; the latter, with 0.25-micron chip designs, runs on as little as 2.2V. The K6 offers a L1 cache of 64K, the full range of MMX instructions, a large branch-prediction table (8,196 entries) and two levels of branch prediction, and seven parallel-execution units, more than any competing processor. Like the K5, it is highly compatible with Intel's designs. As of this writing, the K6 is available at speeds up to 300 MHz.

A bit later in 1997, Cyrix released the 6x86MX. The 6x86MX and its recent faster sibling, the M II, have a superscalar design, with two separate pipelines, 64K of L1 cache (like the K6), and multiple branch prediction (although the chips each have a branch-prediction table of only 512 entries). Unlike both the Pentium II and the K6, the M II does not have a RISC core; but since this has not hindered speed, it is not a significant issue. In addition to the L1 cache, the processor also features an additional L1 cache of 256 bytes, designed to make the L1 cache itself more efficient. The M II is making significant inroads into the lower-priced PC market, although its performance at the 300-MHz level would suggest that it could fit the higher-end PC market as well.

Motorola and PowerPC

There's more to computing than Intel. Motorola processors have powered Apple's Macintosh since its inception, and SPARC processors have been at the core of Sun's Unix workstations and more. And these are just two of many processor families. Here, we'll look at the processors that have served as the guts of the Macintosh.

The original Macintosh, released in 1984, contained Motorola's 68000 processor. The processor (introduced in 1980) offered 32-bit computing internally, but it was slowed by its 16-bit data bus and a somewhat faster (24-bit) memory bus. The 68000 was a popular processor, powering not only the Macintosh but also the Commodore Amiga (originally the Amiga Lorraine) and the Atari ST, along with some early Unix workstations. It featured a two-stage pipeline and built-in expandability for floating-point operations. The floating-point capabilities weren't actually included until the introduction of the 68040 in late 1990, but the chip was clearly designed with the needs of the future in mind.

That future found its way through the 68010, 68020, 68030, 68040, 68050, and 68060. The 68010 offered support for virtual memory but otherwise remained much like the 68000. The 68020 provided the first major change, increasing the data and memory buses to 32 bits and thus becoming the first fully 32-bit 680x0 chip. It also offered a three-stage pipeline and a

INSIDE THE LAGACY CPU

By Mark E. Donaldson

small (256-byte) data cache. Next up was the 68030, a 20-MHz processor that appeared in 1987. The 68030 provided separate caches for data and instructions, and it included a memory-mapping unit (MMU) directly on the chip. Three years later, the 68040 shipped with data and instruction caches of 4K each, a six-stage pipeline, and a Motorola 68881-compatible floating-point unit. The 68040 was the first Motorola chip built around the Harvard architecture, in which program and data buses are kept separate for faster execution.

In the past few years, the 680x0-based Macintoshes have been supplanted by those bearing PowerPC processors. Developed jointly by IBM and Motorola, the PowerPC is a RISC processor top to bottom. The first implementation, the PowerPC 601, allowed up to three instructions to be executed per clock cycle, and the chip offered 32-bit memory addressing and 52-bit virtual-memory addressing.

The PowerPC line continued through the 602, the 603 and 603e, the 604, the 620, and up to the 750 (more popularly known as the G3). Of course, the G3 is the best known of the PowerPC chips, because it is the star of a series of TV ads parodying the futuristic, song-and-dance Pentium MMX ads. In the G3 versions, the Pentium performers get toasted; Apple has centered much of its promotion around the fact that in testing, the G3 processors consistently garner stronger performance numbers than Pentium IIs of comparable or even higher speed. For example, the 266-MHz G3 tests faster in everything from floating-point operations through graphics performance than a similarly configured 300-MHz Pentium II or Pentium Pro. Part of this difference is owing to the G3's L1 cache size of 64K (versus the Pentium II's 32K), and much is also owing to the all-in-one design of G3-based systems, with all components (including a graphics component based on ATI's 3D Rage II) built onto the motherboard.

One of the most significant reasons for the high speed of the G3 is its inclusion of a backside cache. The Pentium Pro has the L2 cache built into the processor chip; similarly, the G3 bypasses the system bus and uses a dedicated bus to handle L2 transactions, and the L2 cache is built right into the chip design. This method allows the processor to access the cache much more quickly than it could through the system bus. The result is a significant gain in the speed at which the Mac can execute instructions. But the RISC basis of the PowerPC can claim some of the credit as well, with the chip's simpler architecture leading to faster execution of processes. It remains to be seen whether the PowerPC will maintain this lead as Intel-based machines move into the next generation, but for now the difference is certainly real.

Wrapping Up

The history of PC microprocessors demonstrates how technologies can be grafted onto other technologies in order to achieve improved performance and greater sophistication. But as fast as the changes have been, in the future these changes, if anything, will be even faster. Computer use will become more demanding, and the processor must bear the brunt of the demands. It is, after all, the heart of the whole thing.