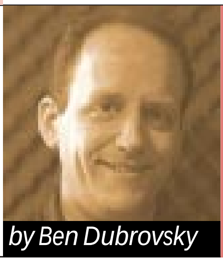
Mips, Megaherz, and Megaflops



by Ben Dubrovsky

Just about everyone these days knows how to operate a computer, but few people know how a computer operates. This month's column delves into the core of what makes your computer tick.

I recently received an e-mail from a reader regarding my column in the September '96 issue. In that article, I talked about the computer executing "instructions" and related these instructions to the "megahertz" rating that computers typically get. The reader properly pointed out that these two concepts are not the same, but also made the erroneous assumption that they were unrelated.

I have always made the point that authoring is a form of programming. I would also like to point out that the more we know about the basic environment in which we operate, the more sophisticated our decisions can be. That's why it's important to look at some computer basics from time to time. While reading this column, be on the lookout for definitions of those key words: megahertz, mips, and megaflops.

Heartbeat

The very first thing to know about computer processors is that they never stop processing. Like the human heart, a computer clock keeps beating. Also like a human heart, the signature of that clock's heartbeat is cyclical: It does the same thing over and over again. Like a mainspring on a clock or a crankshaft in an engine, the computer clock is the device that drives all the inner works of the processor. The clock's heartbeat is a digital signal that goes from zero to one in a particular pattern. Actually, zero is sometimes defined as an electric signal of anywhere from 0 to something like 1.5 volts. One is defined as a signal of anywhere from 3.5 to 5.0 volts. So the clock is a signal on a wire that repeats the same pattern of voltage over and over again. Zero-one-one-one-zero-one-one-one-zero-one-one-one. Each repetition of the clock's pattern is called a cycle. A computer's speed is measured as the number of clock cycles that occur in one second. A computer with a clock speed of 166 megahertz (MHz) repeats the clock's pattern 166 million times every second. Because all the rest of the computer's functions are driven by that clock, the clock speed is directly related to how fast the computer runs.

Like a mainspring on a clock or a crankshaft in an engine, the computer clock is the device that drives all the inner works of the processor.

Micro-code

The next thing to know about computer processors is that they do only three things: store binary numbers, test binary numbers, and manipulate binary numbers. Values are stored in memory, tests are made to determine courses of action, and numbers are manipulated to perform mathematic and string-related operations. These very basic operations are all completely hardware based. There is a physical set of logic gates that know how to test various conditions. Another physical set of logic gates knows how to add two numbers together. Another knows how to generate a request for a particular piece of memory. These basic operations are controlled by a computer language that we typically never see—a language known as micro-code. Micro-code, or micro-instructions, have direct access to the physical structures on the computer chip. They also have the unique property of executing in exactly one clock cycle. Therefore, a 166MHz processor executes 166 micro-instructions every second.

Macro-code

In addition to the micro-code, each computer chip has its own macro-code, or macro instruction set. Macro code is the lowest level programming language that we typically see. When low-level programmers build instructions for a computer chip, they build them out of this language or out of Assembler language. Assembler language is basically the same thing as a macro language, except that it's prettied-up for programming sake. The basic structure of a macro-instruction typically includes an operation, two operands, and a place to put the result of the operation. Operations may include addition, multiplication, data-word shifting and rotating, interrupt testing and setting, and the like. Operands are either memory references or references to registers. Registers contain data, and are directly connected to the processor. Hence they are accessed more quickly than memory, and are the staging areas for mathematical operations.

Each macro-instruction available to a programmer is actually built as a small micro-code program stored way down in the bowels of the computer. For instance, the macro instruction, "add A to B and put the result back into A," might be represented by a micro code program that reads,

1) get data A
2) get data B
3) do addition
4) put result into A

This macro instruction needed four micro instructions to execute. If all macro instructions could be executed using four micro instructions, our 166 MHz processor would execute 41.5 million macro instructions every second, also known as MIPS—millions of instructions per second.

All instructions, however, are not created equal. Some take more than one hundred micro instructions to execute. Such instructions are called "expensive." The price of an instruction is not measured in dollars, it is measured in micro-instructions, or cycles. In fact, computer processors have manuals published that disclose how many cycles each instruction costs. The number of MIPS a computer can execute is actually an elaborate average based on the frequency of use of particular instructions and their cost.

One other bit of information that comes out of this discussion is why Macintosh computers and Intel-based computers use different code. They each have different computer chips which, in turn, sport different macro languages. Code written for one macro-instruction set will just not be understood by the other.

Floating point numbers

The next level of complexity deals with floating point numbers. Computers tend to stay away from floating point numbers—they tend to prefer the predictability of integers instead. Floating point numbers are a complete algorithmic artifice. They're made up of three components: a sign, an exponent, and a value. The number 472.8 is represented as 1) Sign is plus. 2) Exponent is 3. 3) Value is 4728. There is an assumed decimal point in front of the value, so the number 472.8 is the same as +.4728 times 10, raised to the 3rd power, or .4728 times 1000.

Addition of floating point numbers involves a lot of work to normalize exponents, calculate signs, and avoid overflows. All this work is expensive—very expensive. Enter the "floating-point processing unit." This specialized piece of hardware moves floating point arithmetic out of the realm of micro-code and into its own specialized hardware. Because of the importance of floating point calculations to almost all applications, a special designation has been invented: megaflops or millions of floating-point operations per second. Computers with a hardware FPU will do many more megaflops than one without.

If all macro instructions could be executed using four micro instructions, our 166 MHz processor would execute 41.5 million macro instructions every second, also known as MIPS—millions of instructions per second.

Computer languages

Now we know that a processor's basic workings are accessed via microcode. Macro-code or assembler language, the lowest level programming languages available, are built up out of microcode. Higher-level languages, like C, C++, and Pascal are translated into assembler language. Finally, languages like Lingo (sometimes called fourth-generation languages), are executed through programs written in languages like C. So, when we write a Lingo program, it's interpreted by a program that was written in C, which had already been translated into an assembler or macro language, that gets executed inside the computer chip by a micro language.

This knowledge is useful because it should give you an understanding and appreciation for how your authoring is treated once it leaves your hands. Imagine what happens when you ask for one graphic to blend with another. It means that the computer must look at every pixel in each of the two graphics. The number of pixels to look at is a product of the length times the width, in pixels, of the graphic. Now think about the number of translations in computer language that take place, the increased complexity of each level of computer language, and ultimately the number of micro-instructions or cycles, processing those pixels must take.

Budget your cycles

Processing pixels is expensive. Budget the number of cycles to be spent on any particular part of a production, and pinch cycles where possible. And by the way, cycles can also be taxed by the "government" operating system of the computer. Every one-half second or so, the operating system butts in and grabs some cycles to do its own processing. There could be a number of good analogies between politics and processors, but let's leave that for another day, and for a chat over a beer. DV

Ben Dubrovsky *is a multimedia producer and programmer from Brookline, Massachusetts. Reach him at dubrovsky@dv.com*

For more information about this and other subjects check out our Web site at http://www.dv.com. Reach us at letters@dv.com or write to DV Letters, 411 Borel Ave., Suite 100, San Mateo, CA 94402 or fax 415-358-9865.