Embedded Systems Conference 2004
Class # 405: ***Top 10 Lies About Microprocessors***

Jim Turley
jim@jimturley.com
www.JimTurley.com

Lie #1: Intel's Pentium Dominates the Microprocessor Market

Reality: Intel's Pentium has approximately 0% market share. That's right—Pentium is a statistically insignificant chip with tiny sales volume.

Surprised? Try this: all life on earth is really just insects. Statistically speaking, there are more different species of insects than of all other forms of life put together – by a lot. If you round off the fractions, there are no trees, no bacteria, no fish, viruses, mollusks, birds, plants or mammals of any kind. If you need help feeling humble, mammals make up just 0.03% of the total number of species on the planet.

The comparison between Pentium and paramecia is a pretty close one. Ask a friend what's the most popular microprocessor chip in the world and chances are she'll answer "Pentium." The newspapers constantly shout that Intel has 92% market share, or some such number. Clearly, then, Pentium must the overwhelmingly dominant species and all other chips are struggling for that last 8%, right?

Ooooh, sorry, but thank you for playing. The fact is, Pentium accounts for only about 2% of the microprocessors sold around the world. Pentium is to microprocessors what viruses are to life on earth. No, that's too generous. Pentium volume ranks a little below viruses but a little above mollusks (i.e., snails) on the microprocessor food chain. The insects – the overwhelmingly dominant species – are the embedded microprocessors. They're the forgotten phylum that controls (approximately) 100% of the microprocessor kingdom.

Most of the embedded microprocessors are 8-bit CPUs like the 8051 and 68HC11. The sales volume of 8-bit chips is enormous and growing nicely. These little suckers are selling to the tune of more than a quarter of a *billion* chips every month! That's one new 8-bit microprocessor for every man, woman, and child living in the United States, every month.

In other words, Motorola (and NEC, and Toshiba, and a dozen other processor vendors) probably sells more microprocessors in a month than Intel has sold Pentium chips in its entire lifetime.

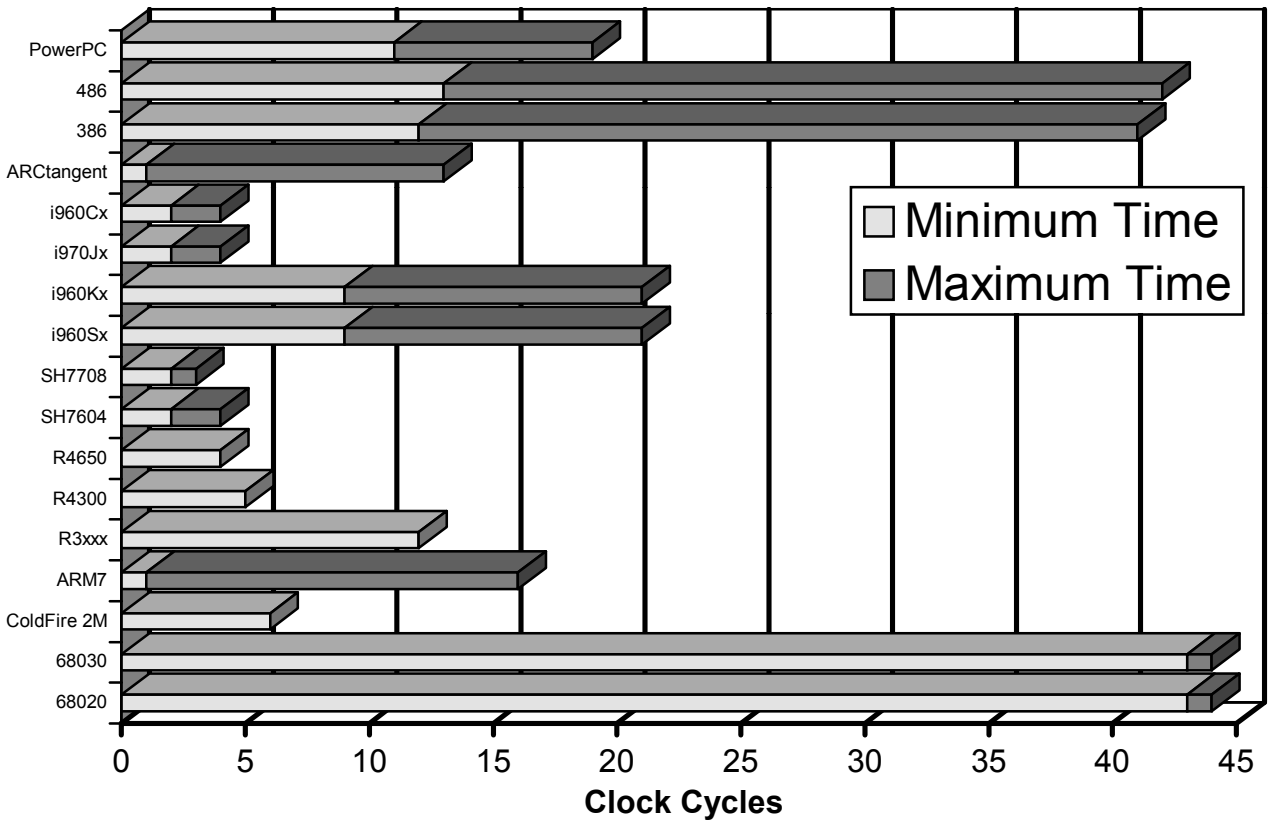
Lie #2: All Microprocessors Are Pretty Much the Same

Reality: The differences are pretty major and if you ignore them, your software probably sucks for no good reason.

To a programmer writing in C or another high-level language, it's true that all processor *appear* to be the same. After all, C code is C code and it doesn't matter what processor the code will ultimately run on. In fact, that's the whole *point* of high-level languages. You're not supposed to know, or care, what kind of processor you're writing for.

That's true, up to a point. You can write generic C code for any processor and it will run, but your choice of processor can have a tremendous effect on how *well* it runs. All processors are not created equal, and the differences among them can make a big difference to how fast, efficiently, or compactly your programs work.

Consider the example below. This chart shows how different



processors perform a simple multiplication operation. That is, they're multiplying two 32-bit numbers together. Simple, right?

Yet notice that the speed of this operation varies greatly. Hitachi's SH7708 processor, for example, can multiply *eight times faster* than a 68030. They both produce the same answer, of course, so they're both correct, but one is far faster than the other. If you looked at the software alone, you'd never notice this difference; it's not visible to software at all. Only by properly evaluating different microprocessors would you even know about this difference.

(Note that this graph is normalized for clock cycles, not frequency or MHz. The relative speed of these processors has nothing to do with the comparison.)

But wait – there's more. Notice how most of these processors have a min/max speed. That means the time to multiply two numbers changes; it's not the same all the time. In fact, the time is data-dependent: it depends on the numbers (the multiplier and the multiplicand) begin multiplied. Just as with people, it takes processors longer to multiply some numbers than others.

How would you ever notice this in software? You wouldn't, because as far as a program in concerned, multiplying is simply one instruction or one line of C code.

This difference can be especially important in real-time systems. Counting clock cycles or calculating latency is impossible when execution time is data-dependant. Unless you can predict exactly what numbers you'll be multiplying (in which case, why bother doing the multiplication at all?), you can't predict run time.

Lie #3: RISC Is Better Than CISC

Reality: Not necessarily, and it depends on what you consider "better."

RISC is very fashionable these days. It sounds new, modern, and somehow better. CISC is berated as "the old way." Yet what is it exactly about RISC that's supposed to be so great?

RISC is, by definition, a "reduced instruction set" processor. In other words, fewer instructions for software to use. Put another way, RISC architects decided that anything that could possibly be done in software should be, and anything that can be *removed* from hardware should be. It's a minimalist, Spartan, point of view. And it leads to bulky code.

Because of this reliance on software instead of hardware, it takes more software to get the same amount of work done, compared to a CISC chip. For example, most RISC processors cannot divide two integer numbers. Division is considered unnecessary and better done in software. So instead of one DIV instruction in your program, you need an entire software division routine that handles even and odd results, overflows, divide-by-zero errors, carries, borrows, and all the rest of the fun. This is an improvement?

As a rule of thumb, RISC code density is only half as good as CISC. In other words, the exact same C program is likely to be twice as large after it's compiled on a RISC chip, compared to a CISC chip. Same code; same functionality. Just twice as large. This is an improvement?

Lie #4: Java Chips Are Coming

Reality: They're already here, but this is about as good as they're ever going to get.

People have wanted Java chips almost since Java, the language, was invented. First of all, this is supremely ironic. The *whole point* of Java was to make programs hardware independent. Using a special-purpose Java chip defeats the whole purpose of the language in the first place.

No matter. The reason most people want Java chips is to improve the miserable performance of Java programs. Java, unfortunately, is terribly slow when it's run on any normal microprocessor. Surely that's only because the market hadn't delivered the *right* Java processor yet? Then things would improve, right?

Wrong. Java is inherently a low-performance language. It does not, and cannot, map well onto any microprocessor, no matter how it is designed. This is not a coincidence, and it's not going to go away any time soon. Bright people have been designing computers and microprocessors for decades – literally hundreds of different architectures have been designed.

And none of them looks like Java. There's a reason for this: Java is a very inefficient system that doesn't work well in the real world. As a language, it might have its strong points, but as a real, live platform it's terrible.

The current crop of Java "accelerators" from Nazomi, inSilicon, Aurora VLSI, and others do help to improve Java performance but not one of them is a true "Java processor." They cannot execute all Java byte codes, only some of them. The rest are trapped as illegal instructions and emulated in software (which is the way all Java code is run on most systems). Even Sun Microelectronics itself canceled its ambitious Java-processor program after some initial dismal failures.

Lie #5: Dhrystone Benchmarks Are Useful

Reality: Dhrystone MIPS is a totally meaningless, made-up number that only measures the fortitude of the marketing department.

Once upon a time, a benchmark program called Dhrystone was useful. It was used in the 1970's to measure the relative speed of big mainframe computers from DEC, Sperry, IBM, and others. Its original purpose was to compare other mainframes to the VAX 11/780. And it wasn't originally written in C, it was written in PL/I.

Somehow, this historical anomaly has persisted into the present day. It performs utterly useless work, which a good compiler can optimize away, and reports its run time. It knows nothing about floating-point performance, caches, memory latency, buses, coprocessors, accelerators, or superscalar performance. Dhrystone is so easily modified and perverted that some compilers have a `-dhry` switch that performs special "optimizations" to improve test scores.

Dhrystone measures a compiler much better than it measure the processor it's running on. For example, text runs of different (and identical) 486-based PCs yielded Dhrystone benchmarks that varied by 100%, based only on different compilers.

Lie #6: Processor "Brand X" Has Lowest Power Consumption

Reality: Processor power consumption drops all the time, so comparing yesterday's processor to today's isn't really fair.

A few years ago, ARM processors, and particularly the ARM7, developed a reputation for being ultra-low power processors. It seemed as though ARM-based systems could run all day long on just a little sunlight or the static electricity from rubbing a cat.

First, ARM has never manufactured a single processor chip so it's impossible to measure the power consumption of a chip that doesn't exist. ARM's *customers* manufacture chips by the thousands, however, so why not measure *their* power usage? That's not fair either, because all of those ARM-based chips have lots of other electronic circuitry in them. The ARM processor core is probably only 5% of the entire chip, so it's not very relevant to the overall power consumption of the chip. It's a bit like judging the value of a car by measuring the size of its wheels.

Second, ARM's reputation for low power was in comparison to much older processors that were far less efficient. It's no secret that newer silicon technology uses less energy than older silicon processes. Thus, newer chips automatically get a low-power boost (reduction?). Comparing today's hot new processor to yesterday's lukewarm processor doesn't tell anyone anything.

To be fair, ARM7 processor cores are small and efficient compared to larger, more complex cores like a DSP or a 386. But you also get less for your money, so to speak, because simpler cores provide fewer features. There's no free lunch.

Also to be fair, there are a number of 32-bit cores that use even less power than the famous ARM7, and provide pretty much the same features. So if low power is what you're looking for, be sure to look beyond the hype – and beyond ARM.

Lie #7: Performance is Proportional to Price

Reality: Price is determined by nothing except marketing. This is particularly true at the high end of the 32-bit range, where processors cost $50 or more. And most of all, it's true of the PC business.

Microprocessors are not priced in relation to their performance or feature set. They're not even priced in relation to their manufacturing cost. If that were true, most embedded processors would be selling for about $0.12, because that's about how much silicon they use. No, prices are purely at the whim of the marketing department from the relevant vendor.

For example, Hitachi's SH7750 processor (used in the late Sega Dreamcast video game) can perform 3D geometry calculations better than a Pentium II. Yet when both were at the height of their production runs (about 1998), the SH7750 cost $40, while the Pentium II sold for $630.

As another example, Motorola's 68060 processor ($500) was slower – by a lot – than some $25 MIPS processors of a few years ago. Motorola has since dropped the price of the '060 but the MIPS processors have only gotten faster.

Don't let price be your guide. Prices change all the time and they bear no relation to anything real, anyway. Evaluate processor performance for yourself and make your own choices.

Lie #8: MIPS Chips Are Good At Graphics

Reality: Hardly. MIPS chips are often used *in* or *near* graphically intense systems, but they're not responsible for the graphics.

Gee, if Silicon Graphics, Nintendo, and Sony (PlayStation) all chose MIPS processors for their systems, MIPS chips must be red-hot at graphics, huh? As if.

The fact is MIPS processors – like most RISC processors – haven't got the first clue about graphics. In fact, MIPS processors probably have less graphics ability than an old 68030 or '386 chip. Although MIPS processors are (or have been) used in a number of impressive 3D systems, the main microprocessor itself had almost nothing to do with the graphics. Silicon Graphics, Nintendo, and Sony all used separate graphics processors (or boards) to handle the graphics. In some cases, this chip was larger than the MIPS processor that was supposedly driving it. RISC philosophy dictates that all unnecessary features be removed from the processor and MIPS adheres to this doctrine even more closely than most. There's no harm in

using a MIPS processor in a graphics system. Just don't expect it to do any of the heavy lifting for you.

Lie #9: I Should Use an x86 Processor Because They're the Best

Reality: Depends on what you mean by "best." They're not the most common, nor the fastest, nor the cheapest processor around. Is that best?

As mentioned earlier, the popular of Pentium and the rest of the x86 line has little to do with their technical features and a lot to do with PC compatibility. Obviously, x86 processors dominate in computers, but that dominance does not extend to embedded systems. Embedded x86 processors rank about fifth in popularity, behind 68K, ARM, SuperH, and MIPS. No small potatoes, but hardly a dominant position.

Embedded designers choose x86 processors like the '386 or '486 for one of two reasons: PC compatibility or tool availability. Both are good reasons, but performance, low-power, and price are generally not among the reasons one picks an x86 processor. If you're looking for the best hardware features, look elsewhere.

On the other hand, there's a lot to be said for having the world's best support infrastructure. It's easy to find x86 programmers, it's easy to find compilers, it's easy to find operating systems, and it's easy to get help and support. Whatever bugs there are in the chips were discovered long, long ago. You won't be blazing any new trails.

Lie #10: There's a Shakeout Coming and My CPU Will Disappear

Reality: Possibly, but not likely. The number of embedded processors is growing, not shrinking.

Fears of a shakeout in the embedded processor industry are fed by similar events in the PC and workstation business. In the early 1990s new RISC companies sprouted up weekly. RISC was going to overthrow the CISC (read: Intel) dominance of the computer (read: PC) industry and unleash unprecedented levels of performance. That didn't happen, and one by one, all the RISC makers (except Sun) quietly folded their tents. Most

either went out of business or started sleeping with the enemy and using Intel processors in their systems. The RISC war was over, and RISC lost.

So when is a similar shakeout coming for embedded processors? It isn't. Intel dominates the computer business (PC and workstations) because all computers are essentially the same. One CPU architecture can serve them all, with only minor variations. But all embedded systems are not the same, or even very similar. Embedded systems account for 99% of all the microprocessors sold in the world, and they are extremely varied. No one CPU family, and no one vendor, can ever hope to supply all that demand.

Now, with user-configurable processors on the scene, the choice of processors will become even greater. Programmers and engineers can design their own, personal "boutique" processors for specific applications.

Jim Turley is an independent analyst, columnist, and speaker specializing in microprocessors and semiconductor intellectual property. He is editor of Silicon-Insider*, a columnist for* Embedded Systems Programming *(ESP), was past editor of both* Microprocessor Report *and* Embedded Processor Watch, *and host of the annual* Microprocessor Forum *and* Embedded Processor Forum *conferences. For a good time call (831) 375-8086, write jim@jimturley.com or visit www.jimturley.com.*