

MICROPROCESSOR HISTORY PART II

Tech Tip 55 - Microprocessor History (Part 2, More Than a Toy)

Article by Roy Davis

In the last installment we talked about the very early days of microprocessor history and how it grew out of a programmable calculator into the first personal computers. The history is just the foundation for understanding the features that make our current day microprocessors so powerful and helps to illustrate what is really important in your next computer purchase.

There have been many players in the microprocessor story, but the two companies that have had the most impact are also the two that dominate the market; Intel and AMD. Along the way, other big companies like IBM and Motorola made major contributions but have fallen by the wayside in the marketplace for personal computers.

1. Expanding Address Space

We started with the Intel 4004 and how that first device could handle only 4-bit numbers and up to 640 bytes of memory address space. Nowadays, it's common to plug in hundreds of megabytes in a single Memory Stick

[<http://www.geeks.com/details.asp?invtid=1024DD R3200-N&cat=RAM>].

Running the latest operating system like Microsoft Windows XP requires quite a bit of memory. Working on digital photographs or running the newest computer games requires even more. It's not uncommon to have a Gigabyte (1,000 Megabytes) of RAM in your computer.



2. Finding the Way

How does the microprocessor keep track of all that memory? To answer that, we need to talk about some of the components of a microprocessor. Last week, I

mentioned the program counter. The program counter is a register. Your microprocessor is full of these registers; little chunks of hardware that are like a sticky note you would use to remind you of something. Registers are like this for the microprocessor, easily and quickly referenced by many of the instructions the computer executes.

A microprocessor will have several of these registers dedicated to pointing to memory addresses. The program counter indicates the next instruction, while the index register is used to automatically step through tables of data. A stack register keeps track of memory addresses to return from program subroutines.

The memory addressing registers of an 8-bit microprocessor are almost always twice as big at 16-bits. When personal computers shifted to 16-bit microprocessors, the registers grew to 32-bits. That meant there were a lot of specialized instructions and multiple steps to manipulate the memory addresses. This slowed down the execution of programs. It became obvious that a microprocessor that can handle a full memory address in one chunk would run faster. That's why we are using 32-bit microprocessors even in our low-end systems [<http://www.geeks.com/details.asp?invtid=PX743AAR&cat=SYS>], and the big boys [<http://www.geeks.com/details.asp?invtid=PS578AAR&cat=SYS>] are sporting 64-bits at a time.

3. Segmented Verses Unsegmented

Early on, the struggle with handling large memory address space while maintaining microprocessor speed took two paths. Motorola took the simple approach by making the memory space "flat" so that it is addressed as one big continuous memory string. The instruction set was "symmetric", meaning for every read operation there was a corresponding write.

Intel took a more convoluted route. They broke the memory up into "segments". Simple and fast addressing modes were used within the segment. The program instructions and data usually lived in different segments. When a program had to jump to another segment or retrieve data from a different segment, the segment registers had to be modified, which could take several steps. The idea was that the slow crossing of segment boundaries was more than made up for by the simple and fast operation within a segment.

In the first IBM PC, the segments were 64 Kilobytes, which was pretty limiting then. Intel was quick to respond to this when they released the 286 microprocessor in 1982 by expanding the segments to 1 megabyte. The Intel 386 microprocessor took the segment size to 4 Gigabytes in 1985 and segment size hasn't been an issue since. This Intel segmented design became more popular than the un-segmented Motorola flat memory.

4. Memory Protection

The one feature Intel offered that made it the microprocessor of choice was memory protection. When we had a simple operating system and ran one program at a time, memory protection wasn't an issue. Now that we are running multi-threaded OSs and have a dozen applications in memory simultaneously, there was a need for a more sophisticated memory addressing method than the simple flat model.

In the flat model, when a program runs away it can mess up the code or data from other programs. This was difficult to debug because there was no indication of which application caused the problem. With the Intel segmented architecture, a program cannot delve into the memory space allotted to another application. If it tries, a memory fault message is generated and only that program crashes. You usually can recover from this situation without having to reboot and you know exactly which program was the culprit.

5. Multitasking

The Intel 386 was a break-through microprocessor in another very important way. It had hardware and special instructions that supported true multitasking. Nowadays, we take it for granted that several applications can run simultaneously. In actuality, only one program is executed at any particular time. Your computer runs so fast that it can let each program run for just a short bit of time, then switch to the next program.



Prior to the 386, in order for two pieces of software to be in memory and multitasking back and forth, the software had to be specially written to run for that short time. This scheme depended on the software to cooperate, a burden that rarely worked out in the real world.

The 386 could stop a program in its tracks and suspend it while other programs ran. Then the OS would switch back to the first program as if nothing had ever stopped it. This is called preemptive multitasking and is an important concept to remember for later. Finally there was a window of opportunity for the Microsoft Windows operating system!

6. Multi Data

The microprocessor was just now catching up to the features of the mainframe and minicomputers that came before it. In 1997, Intel introduced their MMX feature with the Pentium II microprocessor. This is a trade name for Single Instruction, Multiple Data (SIMD) capability.

Playing video games on a personal computer was all the rage and even engineers wanted faster graphics performance for rendering 3D models. The Pentium class microprocessors could handle data in 64-bit chunks, which was great for heavy-duty scientific calculations, but graphics number crunching only needs 8-bit numbers and it needs it fast!

A peculiarity of video processing is that the microprocessor has to do the same calculation over and over for all the pixels on the screen. The Intel engineers devised a way to reuse the 64-bit calculation hardware by splitting it up into smaller pieces; either two 32-bit, four 16-bit or eight 8-bit registers. The calculations were restricted to integers making it ideal for the simple graphics processing of the day.

7. Faster and Faster



AMD countered the Intel Pentium II MMX with their 3Dnow! SIMD design. This expanded the SIMD concept with floating point calculations that extended the range of numbers that can be crunched. Only a year later,

the Intel Pentium III [<http://www.geeks.com/details.asp?invtid=PIII500S1-2&cat=CPU>] came out with SSE, or Streaming SIMD Extensions that further improved on the MMX design with more flexibility with registers expanding to 128-bits.

Note that SIMD is not multi-threaded or even multitasking. SIMD only works when special instructions are used by the programmer. The data has to be organized in a very strict format to fit into the SIMD registers. Even with all the restrictions, SIMD freed the PC as a graphics processing powerhouse that unseated the specialized graphics silicon from companies like Silicon Graphics.



8. Contrary to KISS

In most cases, KISS (Keep It Simple Stupid) works. Simple is better than complicated, except in the case of microprocessors. One group of companies tried to keep to the KISS principle and designed Reduced Instruction Set Computer (RISC) microprocessors. They believed that by keeping the instructions very simple they could make the microprocessor perform faster that it would make up for having to use more of the simple instructions.

Intel went the CISC (Complex Instruction Set Computer) route where they have many many instructions and lots of those could do some pretty complicated

things. What the RISC guys didn't count on was that Intel and AMD could fabricate their own microprocessors with the complexity of more than 100-million transistors and get that complex machine running at gigahertz rates.

Scientists and engineers had favored the expensive RISC workstations for their believed superior speed performance over the PC. When it became obvious that RISC was a dead end, scientific and engineering software migrated to the PC with those inexpensive yet powerful CISC microprocessors.

9. Supercomputers On the Cheap



The supercomputer is the ultimate number crunching machine for physicists to work out the mysteries of the universe, meteorologists to predict the weather or for doctors to dissect human genes. They have to process lots of numbers over and over. Hey, wait a minute, that's just like the video processing for our video games. Well, guess what. Those supercomputers are built out of the same microprocessors that are the brains of our PCs! They just connect lots of them together to break the problem up into smaller chunks and process the data in parallel.

Final Words

That was some pretty dense technical stuff! I hope you caught the gist of it because we'll be using a lot of these terms in the next installment. All this talk about processing data in parallel is leading up to something. We'll get around to

talking about hyper-threading and how even within a single microprocessor there are a lot of things going on in parallel.

It will lead to other features like superscalar processing where more than one instruction can be executed during a single clock cycle. All this to make your programs run faster!