No. 147, February 2009

THE COMPUTER CORNER

No. 147. THE BOOT PROCESS

- by Stan Kaplan, WB9RQR
715 N. Dries Street
Saukville, WI 53080-1664
(262) 268-1949
skaplan@mcw.edu

I wrote about this very topic twelve years ago, way back in August 1996 (No. 34), but it is time to revisit it again. There are significant changes between modern computers and those relics we used 12 years ago, plus the readership has turned over since then. So, lets revisit the topic.

Once you hit that power button, the motherboard springs to life and initializes its onboard chips, some of which have firmware on board – little packets of hardwired or flash memory that give instructions as to what to do. But the CPU (Central Processing Unit – the big, expensive chip that runs most everything in a computer) has not started doing its thing, just yet.

Many modern motherboards have more than one CPU, or, (as is true for the computer I am writing this on), several independent cores inside the single CPU. If there are, for example, two or more CPUs or cores, a binary coin is flipped and just one CPU is chosen. The other(s) are halted, while the chosen one runs all of the BIOS (Basic Input Output System) and just the foundation (called the kernel) of the operating system. Only later, after the operating system is working, will the other inactive CPUs or cores be called on to come back to life. But that is getting too far ahead. Here are the step-by-step essential details.

So, the winning CPU/core is chosen and springs to life. At this point, it is able to only address 1 megabyte of memory, just like the old Intel CPUs of the 1970s and 80s (remember the term, “real mode”?), with one major exception. It is able to look at a single instruction, only 16 bytes long, hidden well above the usual 1-megabyte ceiling. It does this, and finds there a command to “go look at whatever instructions you find in the BIOS chip residing on your motherboard and clear the instruction your are reading right now”! Sort of like the secret agent who is given instructions on a piece of paper, and told to then destroy that paper. So, the CPU clears those 16 bytes and begins reading the instructions coded in the BIOS chip. At this point, all of the usual RAM chips (Random Access Memory – the memory “sticks” in your machine) are still full of random ones and zeros. The CPU then starts executing the instructions as it reads them from the BIOS, and this starts some of the hardware in the machine. The CPU then starts the POST (Power On Self Test) that tests other hardware components in the system such as the video card. If the video card passes, it moves on; if not, it will send a series of beeps to let you know (since the screen will not be working). If all is well with video, logos are printed on the screen, memory is initialized (all the bytes in the chips are set to zero), the memory is tested and the keyboard, mouse and other connected devices are tested. In the background, the CPU (still following instructions from the BIOS) sort out all the resources in the machine such as ports, memory and so on. It even builds a table of devices that are present in the machine, to be used later.

Next, the CPU goes looking for boot instructions for an operating system, commonly found on the hard drive. However, in today’s machines, we can configure where it looks and in what order. It may go looking in order at a floppy, a CD, a USB flash drive, or the hard drive, depending on what the user has selected and written in the user-configurable portion of the BIOS. If it finds no instructions in any device it has been commanded to look at, the old (circa 1980s) “Non-System Disk or Disk Error” message appears.

Let us assume the hard drive is the first device in the BIOS’ list of places to search. The CPU will then search the first 512-byte sector of the hard drive, known as the Master Boot Record (MBR). It reads that and plunks it into memory (the RAM sticks are now in use). Two major things are there in the MBR. One

is a Partition Table, a little 64-byte table that tells how the hard drive is divided up (for example, in C:, D: and E: logical partitions, or whatever). Also present in the MBR and physically before the Partition Table is a little (less than 512 bytes) boot program for the Operating System. Assuming it is some flavor of Windows, that little boot program says, "Go look at the first sector of the active partition (only one is active, no matter how many there are) to get further instructions." The CPU then searches the active partition and normally finds instructions there to load the kernel of the Operating System. Just prior to loading the kernel, the machine can go into protected mode, wherein it can access more than 1 megabyte of memory for the first time since the boot process started.

Now that the kernel is loaded, at least the fundamentals of an Operating System (OS) are in place. The kernel can and does instruct the CPU on how to load the rest of the OS – there are files here and there that are read into memory and contribute to building of the OS. That "Windows is Starting" message on your screen signifies the building process is underway and nearly complete. Voila! Your desktop appears and you are ready to go.

A pretty amazing and intricate series of events! For me, the most astounding fact is that, except for a few very small files that the CPU reads directly from the BIOS early in the boot process as outlined above, everything must get put into RAM before it can be read and acted on by the CPU. I constantly have to remind myself that nothing appears on the screen if it is not first in those RAM sticks. And, when editing a document on the screen as I am doing now, until the edits are saved on the hard drive, everything will go "poof" if those RAM sticks loose power! That is exactly why I just this moment clicked the Save Document icon in Microsoft Word. Whew! I am safe now. Happy Computing!