

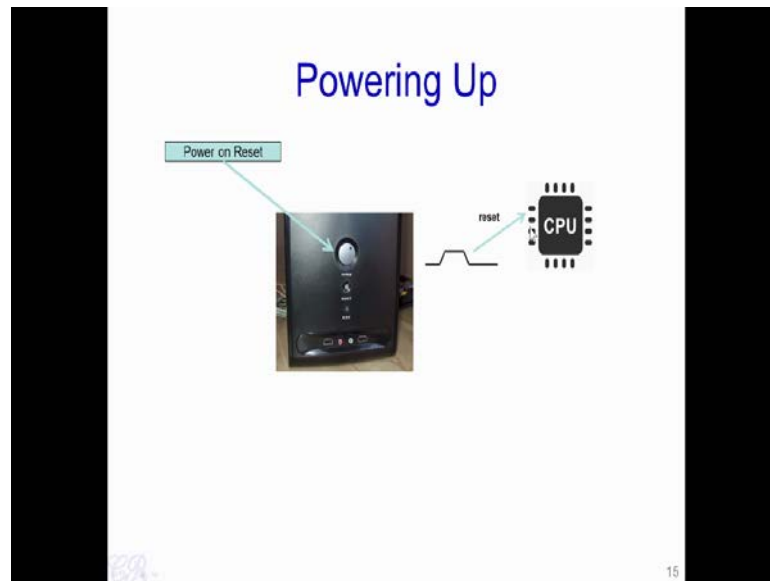
Introduction to Operating Systems
Prof. Chester Rebeiro
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Week - 02
Lecture - 10
PC Booting

Hello. In this video, we will look how a PC Boots, right up from you turn it on to the time the operating system executes. Now this particular video is especially applicable for Intel and AMD based platforms. So, one big thing which we should remember when we are talking about the Intel platforms that we typically use in desktops or laptops is the concept of backward compatibility.

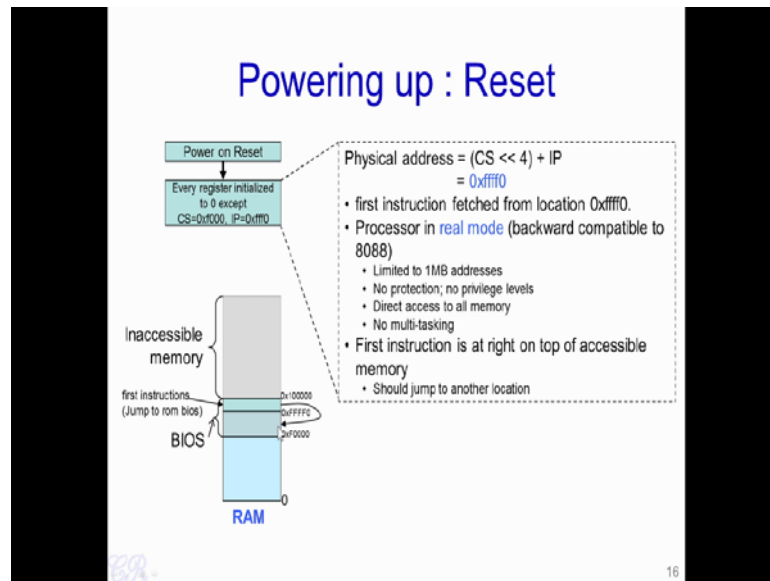
As we have seen in a previous lecture. Intel maintained backward compatibility. So, it ensures even today that any code which was developed and an Intel processor 20 or 30 years back which still execute on the and Intel processor are used today. Because of backward compatibility, lot of things that we actually do while loading an operating system would reflect on what was done 30 years back. Essentially, the things what happened in and system in around 95 or 97. That is when the 386 based processors where used, are still done today on the latest i7 processor from Intel. So, we will look at how PC Boots.

(Refer Slide Time: 01:50)



Now, we all know that in order to start a computer, we need to press the reset button or the start button present in the desktop. So, what actually happens internally is that when you press this button, it is going to send a signal to the CPU the signal for instance would be a pulse, these an electrical pulse which gets created when you press the start button or the reset button, and this pulse it sent to a specific pin on the CPU known as the reset pin and when the CPU obtains or gets this particular signal about the reset, it is going to start booting. Let us see what are the various steps involved when the CPU starts to boot.

(Refer Slide Time: 02:46)

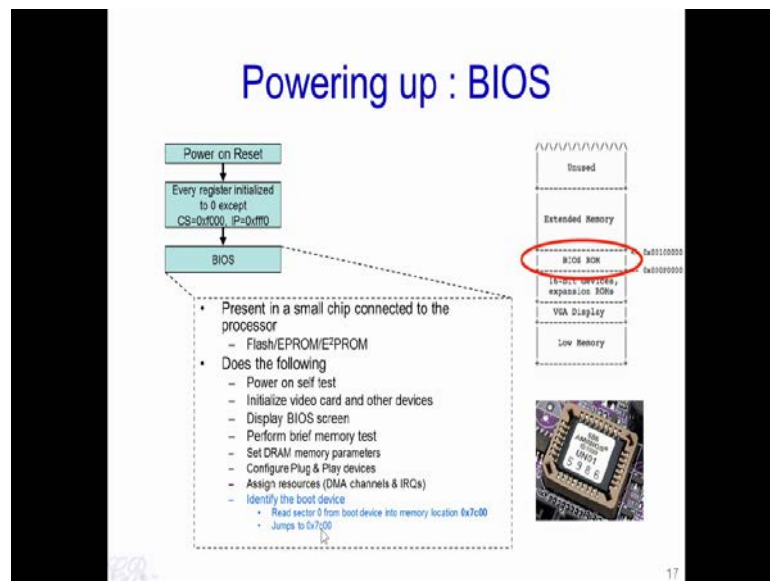


So, we are seen the power on reset. Now when the power on reset comes and the CPU detects it, what happens is that every CPU register which is present inside the CPU, is initialize to 0 except for 2 registers. These registers are the code segment and the IP. Now when the reset occurs the code segment is set to the value of 0xf000 and the instruction pointer is set to 0xffff0. So, if you recollect how an 8088 or an 8086 processor computes its address, is going to take the code segment register in this case f followed by three zeros shifted by 4 bites and add the instruction pointer.

As a result the physical address or the address for the first instruction to be executed will be present in ffff0. Now if you look up the RAM module, and which we had covered in the previous videos what we would see is that the memory address corresponding to ffff0 would be pertaining to the BIOS area. In fact, this particular memory location is just 16 bites below the 1 MB mark. So, this particular thing 0x12345 that is 1 MB is this particular point of this particular address and the first physical address that is put on the address bus by the CPU is ffff0 which is 16 bites below this particular 1 MB mark. So, as a result if you want to boot your system, it should be ensuring that at this particular memory location we have a valid instruction which is present. So, this point over here is the first instruction that is present.

Now, another thing what happens is that soon as the power is reset the processor in is set to what is known as the real mode. So, in the real mode, the processor is in a backward compatibility mode with the 8088 or the 8086. So, recollect that the 8088 or 8086 processor could address at most 1 MB. So, it could address at most this part over here till this thing and this is shown as the green region in this RAM there were other features of the real mode, they where no protection, no privilege levels, direct access to all memory and no multi tasking. So, the first thing that the instructions over here in this particular location ffff0 should do is, to jump to another location. So, essentially what would happen is that it would jump to a location in the BIOS. This jump would trigger the BIOS to start to execute.

(Refer Slide Time: 06:23)



Next, we have that the BIOS ROM over here would begin to execute. So, as we know the BIOS is the basic input output system. It is a read only memory often these days it is in the form of a Flash or is E square PROM and you would actually notice a particular chip like this which is present on your system. So, some CPU's also display that particular BIOS name while booting up for example; this particular chip is the AME BIOS, and it may get displayed by the system boots up.

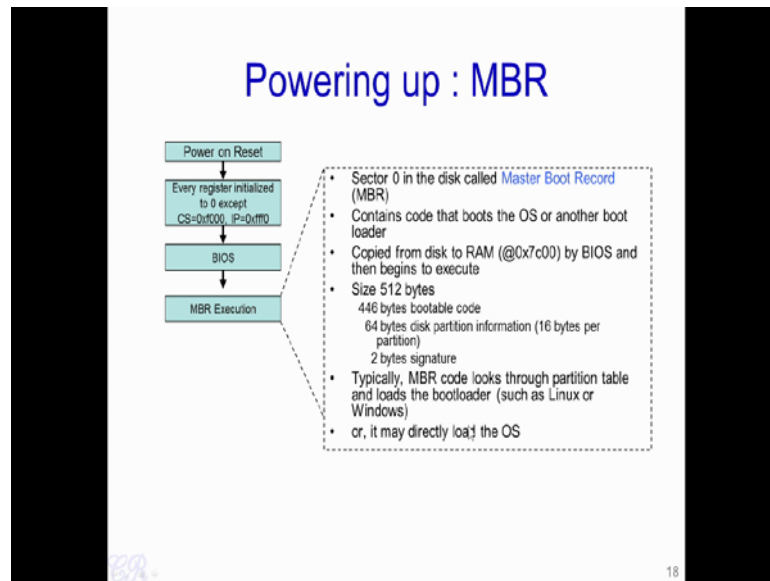
The BIOS present in this particular area of the RAM will begin to execute code in real mode. So, the BIOS do the following essentially first does a power on self test where it checks the system for correctness. It ensures that all parts of the system are working properly. Next it initializes the video card and all other devices which are connected to the system then, optionally it may display a BIOS screen on the monitor. Note that we have initialized the video card, and therefor the monitor is activated and it is capable of displaying things. So, the BIOS screen will now be able to display on the screen.

Then it performs what is known as a memory test and sometimes some of the BIOSs also determine what memory is used and also the amount of memory that is present in the system. After the memory test some parameters are set. For example, these correspond to DRAM parameters and in the BIOS will ensure that various requirements of the DRAM such as the frequency at which the DRAM capacitors are refreshed, are set adequately.

Then plug and play devices are configured, in the sense that all devices which are plug and play are going to be queried and the BIOS will then determine how much memory is required for each of these devices, and these devices are then allocated memory in the system. After that the BIOS will assign resources to DMA channels and the various IRQ's that is the interrupt request from our prospective what is important is the next step where the BIOS identifies the boot device, that is the device which most likely would hold the operating system. It would read the sector zero from that boot device into the memory locations 7c00.

Note that, 7c00 is a memory location in the low memory region of the RAM. So, what it would do is, it would copy the sector 0 which is typically of 512 bytes from the boot device which is typically the hard disk into the memory location 7c00. So, at the location 7c00 we would have 512 bytes of code, which would help in booting the operating system. The BIOS then causes a jump to 0x7c00 what it means, is that the code present in location 7c00 which is presented in the low memory of the RAM will begin to execute.

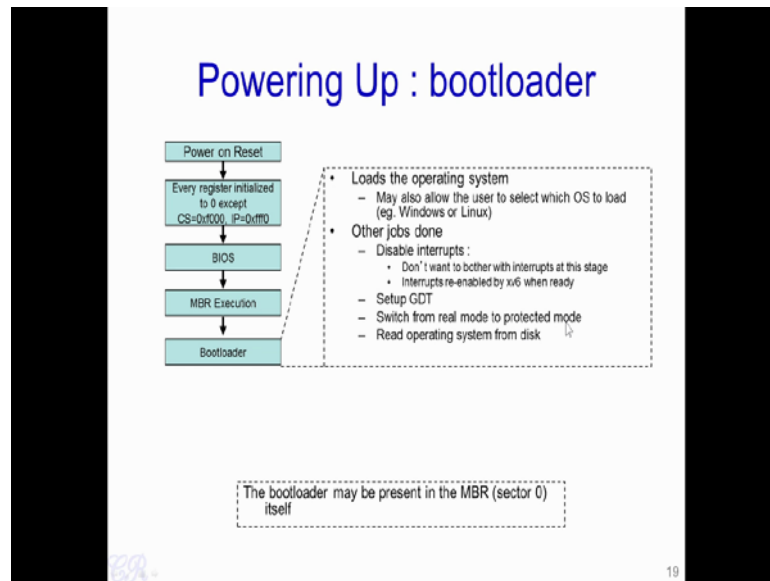
(Refer Slide Time: 10:21)



Now, this memory the memory present in 7c00 and a copied from sector 0 of the hard disk into the RAM, is known as the MBR or Master Boot Record. So, this particular code is of 512 bytes out of which 446 bytes are instructions and contain bootable code, about how to boot the system there are 64 bytes with which have information about the various partitions that are present on the disk. Essentially the 64 bytes are divided into 16 bytes per partition and then there are 2 bytes of a signature which is used to identify whether this is in fact, an MBR code.

So, this code begins to execute from the location 7c00 present in the RAM. What it typically does is that it is going to look into the partition table which is present, and it is going to try to boot the operating system. So, essentially in order to do this, it first loads what is known as the boot loader of the operating system. So, each operating system would have its own boot loader for instances LINUX would have its own boot loader or windows would have its own boot loader and so on. Optionally it may directly load the operating system by itself. So, we will see what happens in the boot loader load.

(Refer Slide Time: 12:04)

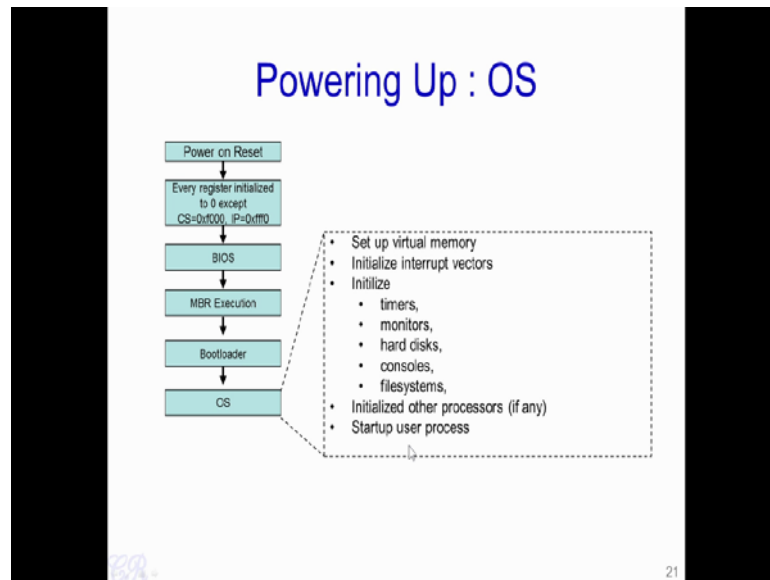


So, after the MBR executes the boot loader would execute. So, the main job of the boot loader is that it loads the operating system. So, it optionally like some operating systems that we see today, it may give an option to the user to select what operating system to load. The other jobs that are done by the boot loader is to disable interrupts, set up the GDT switch from real mode to protected mode and read the operating system from the disk into the RAM. So, these are things which are done by the XV6 operating system. So, there may be slight variations when we go from one boot loader, form one operating system to another operating system.

Sometimes what may happen is that we do not have this MBR code present at all, in such a case the BIOS or rather in such a case the boot loader itself is present in the sector zero of the hard disk, and the BIOS will load the boot loader into the location 7c00 and junk to the boot loader. Essentially what is happening is we are skipping this particular MBR execution. So, once the boot loader executes and sets up the processor and the GDT and switching from real mode to protected mode, it would load the operating system from the disk.

Now the protected mode is a 32 bit mode, essentially where we extend the memory region that can be accessed from 1 MB to the entire region of 4 Giga Bytes. So, we will not go into more details about how this particular protected mode is activated so on.

(Refer Slide Time: 14:13)

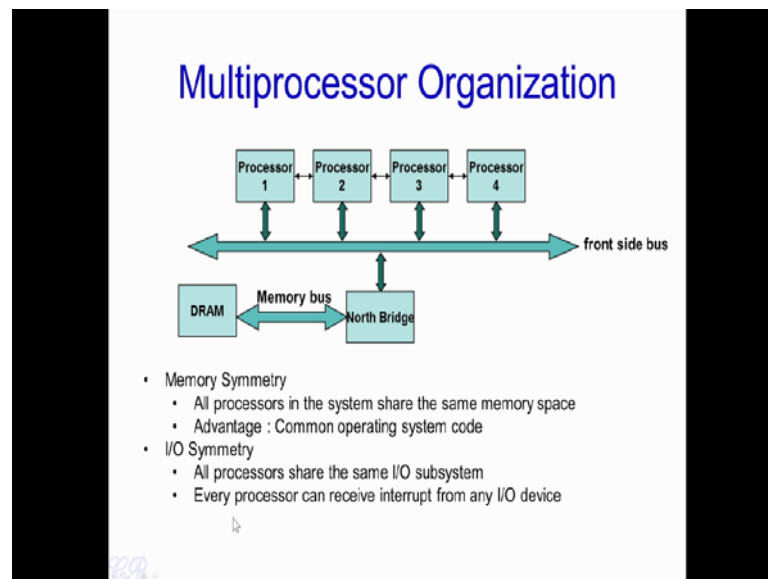


So, once the boot loader loads the operating system, it then transfers control into the operating system. The operating system does several things, like it sets up virtual memory. So, this includes setting up page directories and page tables so on. It initializes interrupt vectors and the IDT interrupt descriptor tables, and the other aspects pertaining to interrupts, then it initializes various devices present in the system like timers monitors hard disks consoles file systems so on.

Then it may also initialize other processes if they are present, and finally it would startup the first user process. So, in a feature video, we will see what the first user process is. This particular user process is the first process that executes in user space. So, you may recollect that all of this executes in the operating system and essentially this executes in the Kernel Space and it is only at this particular point during the boot up sequence, will the user process start to execute.

So, after this, what is expected is that these first user processes will spawn various jobs or user process jobs various DRAMs and so on; and one of its jobs is to create a shell. So, this shell would be then new used by the user, to run various programs and commands and so on.

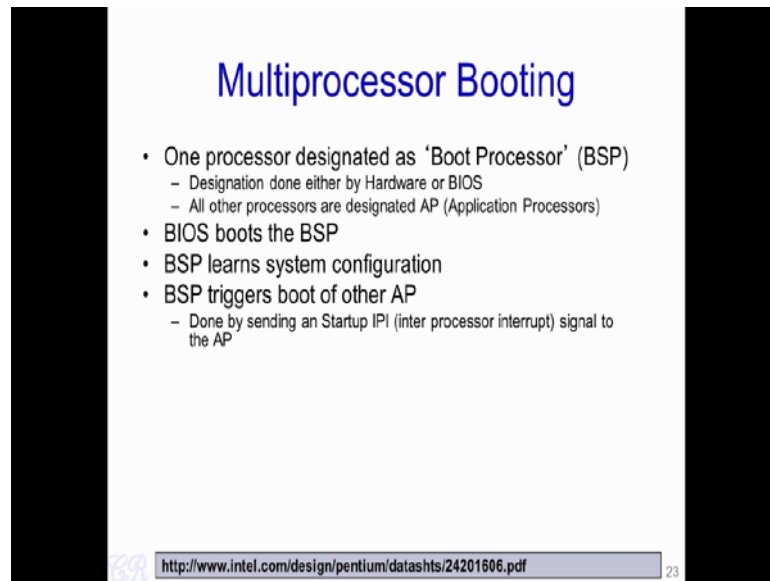
(Refer Slide Time: 16:05)



So, we will now look at systems which have a multiprocessor present in them. So, as we have seen in a previous video in the Intel type of architecture which has a multiprocessor present. So, the all processors share a front side bus, and on the front side bus there is a chip set or the north bridge which interfaces with the memory bus. So, essentially in this Intel type of architecture we have memory symmetry. So, what this means is that, all processors in the system share the same memory space, essentially in order to access a particular DRAM location.

All processors would need to send the same address to the DRAM, and the advantage of having such a symmetric view of the memory is that we can have a common operating system code which could execute in any of these processors. Similarly, there is what is known as the IO symmetry. Essentially what this means is that all processors share the same IO subsystem essentially all processors can receive interrupts from any IO device.

(Refer Slide Time: 17:30)



Multiprocessor Booting

- One processor designated as 'Boot Processor' (BSP)
 - Designation done either by Hardware or BIOS
 - All other processors are designated AP (Application Processors)
- BIOS boots the BSP
- BSP learns system configuration
- BSP triggers boot of other AP
 - Done by sending an Startup IPI (inter processor interrupt) signal to the AP

<http://www.intel.com/design/pentium/datashts/24201606.pdf> 23

Now, in order to boot multiprocessor system, what is generally done is that one processor is designated as the boot processor or the BSP. So, this designation is done either by setting a particular signal in the hardware or by the BIOS itself, and all other processors are designated as application processors. So, when the system is powered on, it is the only the boot processor which begins to execute. So, the BIOS will execute in the boot processor and that is the BSP, and the BSP then learns about the system configuration.

It determines how many other APs are there that is how many other application processors are present in the system. Then it triggers the booting of the application processor. So, after it does all the required initialization it would trigger the boot of the application processors. So, this triggering of the boot is done by something known as the startup IPI or the startup inter processor interrupt.

This is a signal from the BSP that is the boot processor to the application processor. So, when the application processors see this signal they will begin to boot and of course, they identify that they are not the main BSP, but rather the application processor. So, they skip various aspects such as initializing the various devices present in the system and so on. In this video we had seen how the CPU boots right from the time the power on reset

is provided to the processor to the point when the operating system begins to execute and sponsors the first user process.

In the later part of this course we would see various aspects about how the operating system manages memory and manages different processes which are running in the system.

Thank you.