Hello. A lot of how the operating system works depends on the underlined hardware. Essentially, as we have seen in the previous video the operating system is in charge of managing the hardware. So, before we go any further with how an operating system works and how the OS functions, let us take a brief background about the PC hardware, essentially we will look at how a computer hardware is designed.

(Refer Slide Time: 00:50)
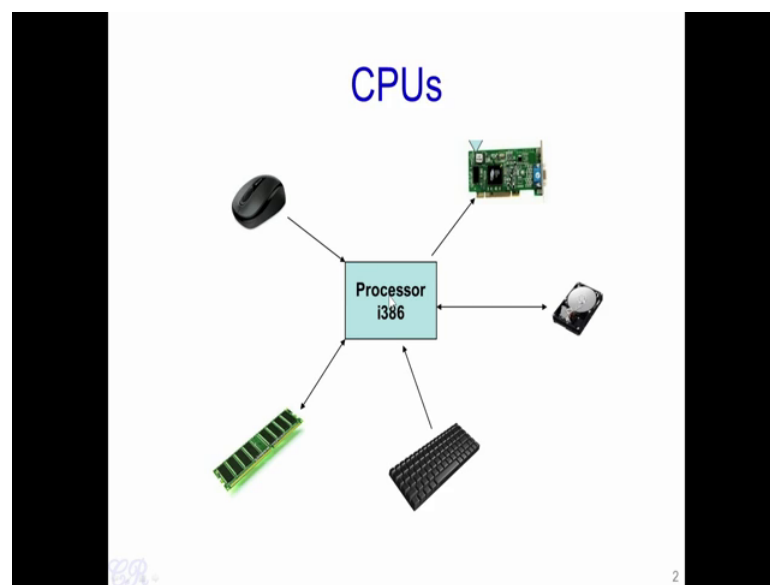
So ,we know that the heart of any system is the CPU; and the CPU is interfaced to several devices such as the VGA card, a hard disk, a keyboard, RAM, mouse and so on. Now, in order that all these devices work with single processors - what is required, is addresses.

(Refer Slide Time: 01:19)
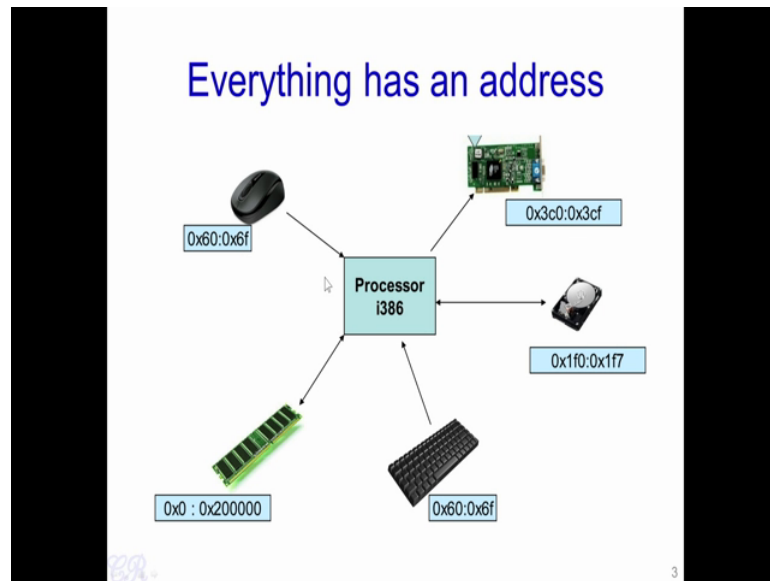
Essentially, each device in the system would have a unique address. And it is ensured that no two devices in the system would have the same address. For instance, we would have the hard disk which is present from the address range 0x1f0 to 0x1f7. So, what this means is that when the processor sends out an address on its address bus, the hard disk would identify that the address is within this particular range and therefore, it will respond.

All other devices will then ignore that particular transfer on the processor bus, because it does not fall in its particular address range. For example, if the processor puts out the address 1f2 then only the hard disk will respond, because 0x1f 2 falls within this particular range of the hard disk. So, if you look at something else like the mouse, which has address range of 0x60 to 0x6f, it is not going to respond to the processors request.
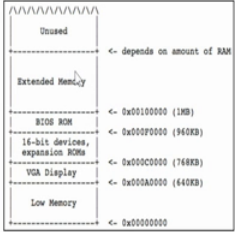
(Refer Slide Time: 02:47)

Now in systems, there are 3 types of addressing which are generally used. One is call the Memory Addressing, next is the IO Addresses, and the third is the Memory Mapped IO Addresses. So, we will look at each of these Types of Addresses more in detail.

(Refer Slide Time: 03:10)

Let us start with the memory addresses. So, the memory addresses and almost system

would have a large amount of such memory addresses correspond to addressing the RAM. Each memory unit in the RAM is given a unique address. For instance, a RAM in a typical Intel system with 32 bits could be of size at most 2 power 32 that is a Intel system with a with a 32 bit processor could have at most 2 power 32 or 4 gigabytes of RAM. So, each and every memory unit in this RAM has a unique address. So, this memory unit could be as small as a single byte, or in some systems it could be a word that is it could be of 16 bits or 32 bits.

Now, how is this RAM used? So, essentially the RAM is configured in this particular way as shown over here, and this configuration is especially for IBM PC compatible Intel machines. So, this RAM as I mentioned has addresses to access each part of the ram, and the RAM could be as large as 4 gigabytes. The addresses from 0x0 to 640 kb represented in hexadecimal as a followed by 0000 is known as the low memory. So, in this particular memory was used in legacy computers like the 8086, 8286 and so on so.

The old operating systems like ms dos would use this particular low memory. So, above this low memory from 640 KB to 768 KB, all addresses would pertain to the VGA display. So, in again this is a legacy issue, where this particular area in the memory would correspond to the VGA display. This means to say that any ASCII characters placed in this particular area would then be taken by the video card, and display it on to the screen.

The region from 768 KB to 960 KB was known as the 16 bit expansion RAMs, so these also were legacy aspects present in legacy computers. And it is pertaining to every device that is used. Essentially, in the previous generations of computers in the early eighties to early nineties, the devices that were attached to the computer could have what was known as expansion ROMs, ROMs are the read only memory. And these ROMs could be accessed or rather these ROMs could be addressed within this particular memory region. Now what is important for us and what we still use in the present Intel systems that we use for a desktops and laptops is this particular region from 960 KB to 1 MB. So, this was where the BIOS recite.

So, as many of you would know, BIOS is the basic input output system; and it is read

only memory which is present on your system. And it ensures that your system boots correctly and it also ensures that the operating system gets loaded. The area of the RAM essentially all memory addresses below 960 KB is typically not used in modern day systems. While the area from 960 to 1 MB that is 960 KB to 1 MB is used by the bios and only used during the booting of the system. What is actually used in modern day system is the RAM above this 1 MB region that is starting what was known as the extended memory. And this extended memory could go far as the amount of RAM is present in the system.

For instance, if you have 4 gigabytes of RAM or 8 or 16 gigabytes of RAM then this extended memory will extend up till that particular region. The parts or the addresses above this RAM area will be generally unused. So, as we will see in the later classes, in this particular course, we will see how the operating system loads itself starting from this 1 MB region. We would also see how the BIOS is going to be used to boot the operating system. Now, as we know this particular RAM is also used to contain various aspects of the applications that we execute. For instance, the code segment or the instruction present in the program that you execute, the heap, the stack as well as the operating system recite in this particular memory. Essentially all of them will recite above this extended memory part.

(Refer Slide Time: 09:41)

Address Types : (IO Ports)

- Range : 0 to 2^{16}-1
- Used to access devices
- Uses a different bus compared to RAM memory access
  - Completely isolated from memory
- Accessed by in/out instructions

inb $0x64, %al
outb %al, $0x64

I/O address range	Device
00 - 1F	First DMA controller 8237 A-5
20 - 3F	First Programmable Interrupt Controller, 8259A, Master
40 - 5F	Programmable Interval Timer (System Timer), 8254
60 - 6F	Keyboard, 8042
70 - 7F	Real Time Clock, NMI mask
80 - 9F	DMA Page Register, 74LS612
87	DMA Channel 0
83	DMA Channel 1
81	DMA Channel 2
82	DMA Channel 3
8B	DMA Channel 5
89	DMA Channel 6
8A	DMA Channel 7
8F	Refresh
A0 - BF	Second Programmable Interrupt Controller, 8259A, Slave
C0 - DF	Second DMA controller 8237 A-5
F0	Clear 80287 Busy
F1	Reset 80287
F8 - FF	Math coprocessor, 80287
F0 - F5	PCjr Disk Controller
F8 - FF	Reserved for future microprocessor extensions
100 - 10F	POS Programmable Option Select (PS/2)
110 - 1EF	System I/O channel
140 - 15F	Secondary SCSI host adapter
170 - 177	Secondary Parallel ATA Disk Controller
1F0 - 1F7	Primary Parallel ATA Hard Disk Controller
200 - 20F	Game port
210 - 217	Expansion Unit
220 - 233	Sound Blaster and most other sound cards

ref : http://bochs.sourceforge.net/techspec/PORTS.LST

7

The next type of addressing is the IO addresses. Essentially in legacy computers like the 8086 and the 8088 and 8286, there was a separate memory region for devices. IO devices like keyboards, mice, hard disks and other programmable devices like a programmable interrupt controller could be mapped. So, unlike the previous addressing that we seen that is the memory addressing, so the IO addressing could be from 0 to just 2 power 16 minus 1. This is roughly around 64 kilobytes and the IBM PC standard would define what the uses of certain addresses are.

So, the IBM PC standard is a standard which was followed to develop the systems like especially the desktop systems and a certain servers. And this standard mentioned what devices should be present in what IO addresses. For instance, it mentions that the keyboard should be present between 60 to 6F. So, the IO address - 60 to 6F.

In a similar way, the DMA controller would be present from C 0 to DF. So, the other things like ASCII or a hard disk - a primary hard disk present in your system will be present from 1f0 to 1f7. So, in this way, several devices had very specific addresses in the system. So, what is the use of such having such a specification is that we obtain compatibility. So, even now when you boot your laptop or your desktop PC, which is an Intel system or an AMD system, the bios will expect several things like it will expect that there is a keyboard which is connected to your system, and the keyboard is present or rather the keyboard is accessible from the address range 60 to 6F. Similarly, it would expect other things like the programmable. A programmable interrupt controller to be present from the IO addresses 20 to 3F.

Essentially, what is maintained even in the modern systems is the backward compatibility to the old computers that we used. So, the memory addressing as we are specified over here, even though a lot of it is legacy and what was used quite a bit in the early 80's and 90's, even now since Intel and IBM followed the backward compatibility. A lot of the systems that we use for a laptops and desktops still follow this particular hierarchy. Now one problem, which was noticed with this type of IO addressing, what is the limited amount of address range that was permitted? So, for instance over here we had at most 64 KB of addresses that could be present. So, this means that the number of devices that could be connected to this system through the IO addressing was limited.
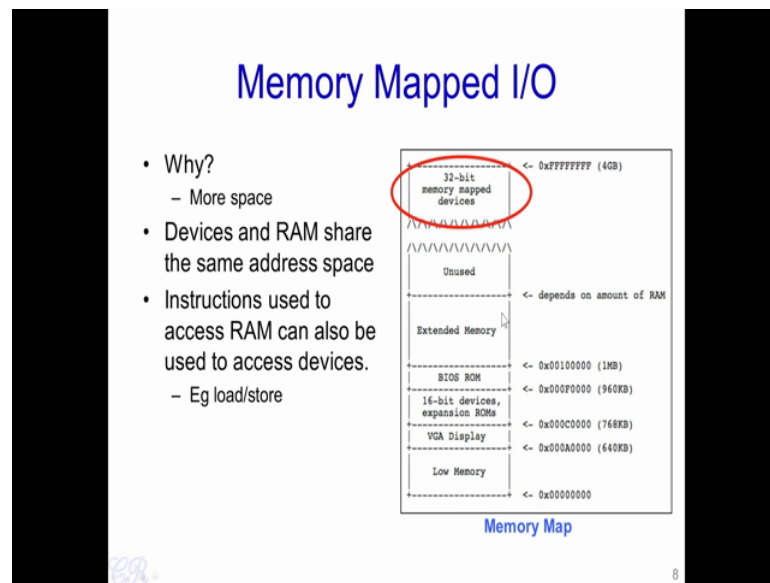
In order to extend this, what was used was known as the memory mapped IO. In this particular memory mapped IO, hardware devices would then connect or would be then mapped into regions in the memory addressing itself. As we have seen before, the RAM or the random access memory of the system would be addressed from the location 0 and extend upwards until the end of the RAM.

So, typically in the systems of the early and late nineties, we would have something like 256 MB, 512 MB or at most 1 GB of RAM. So, the space above this was unused. So, what devices would do is they would map a certain region in this particular part of memory into their devices. What this means is that when the processor generates an address in this upper region of memory that is a high address in this upper region of memory then that particular address would be directed to this specific device and not to the RAM.

(Refer Slide Time: 15:13)

So, we have seen various address ranges that have been allocated to RAM such as the low memory region, the location where the bios should be present the location where various devices like the keyboard, hard disk and so on should be present. Now who decides these particular address ranges? Essentially these address ranges have been decided by standards and a lot is impacted by legacy computers of the eighties and nineties. So, one very famous standard was the IBM PC standard. Essentially, this particular standard was fixed for all PCs that are all personal computers right up from the eighties would be compatible with IBM PCs standard.

Again even many of the systems that we use today that is our Intel and AMD laptops as well as desktops are backward compatible to this IBM PC standard. So, the reason for this IBM PC standard of for that matter, why a standard was required was to ensure that the bios and operating system is portable across platforms.

So, for instance if I would write up an operating system today my operating system would know exactly where the keyboard should be present irrespective to one of what IBM PC compatible system, it is going to be loaded on. So, I could make a lot of assumptions about where various devices are present in the system. So, having such a standard would essentially ensure that the software that we write especially the software

that interacts with the hardware such as the bios and the operating system will be portable across several different platforms.

The other way in which addresses are decided upon is by something known as plug and play. So, plug and play devices do not have any fixed address location as we have seen for things like the keyboard or VGA, memory and so on. But rather when the system boots and when the bios begins to execute, the bios will then detect the presence of some particular hardware in the system. For instance, the bios would detect that the system has a network card present; it would then ask the network card as to how much memory it requires, and what type of memory it requires whether it is IO memory or a memory address. Based on this, the bios will then allocate a portion of memory for that particular network card. So, the allocation would be fixed of for each boot of the PC, but on the other hand the location in the address map would vary across the each boot.

(Refer Slide Time: 18:38)
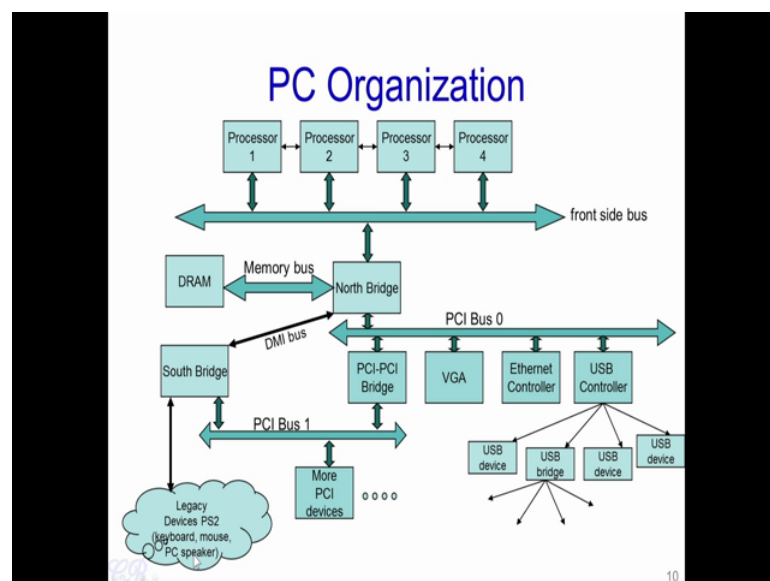
So, this particular slide here shows how a typical PC is organized. So, we may have multiple processors present in the system. So, this could be 1 processor, 2 processor, 3 or 4 and so on. So, it could also be possible that each of this processor have multiple cores inside them; and each core has 2 or 4 threads. Now all these processors are connected to each other through what is known as the front side bus. So, there is always a sharing

between all processors with respect to the front side bus. And now an important device which connects to the front side bus is known as the north bridge or the chip set. So, the north bridge would interface with the memory through what is known as the memory bus; and it also interfaces with the PCI bus.

So, on the PCI bus you could have several devices like the Ethernet controller, USB controller. And the USB controller could have many USB devices, which you see at the front or the back of your desktop. Now as you would notice over here, the USB devices are connected in a tree like structure.

Now the PCI bus also has a hierarchy type of structure; and each bus for instance, in this case, PCI bus 0 which is a closest to the north bridge. And you have in this way a PCI bus 1, which is connected to the north bridge through the PCI bus 0. So, the interface between bus 0 and bus 1 is through a PCI to PCI bridge. In addition to this, there is what is known as the south bridge. And this is south bridge interfaces with the PCI bus or you could have a special protocol or a special connection with the north bridge which is known as the DMI bus. So, in the south bridge, various legacy devices like the PS 2 devices keyboard, mouse, PC speaker, and so on are connected.

(Refer Slide Time: 21:03)

Let us start with how x86 or the Intel x86 process has evolved. So, it all started with the 8088 processor or the 8088 processor as it was generally known as. So, this was a 16 bit microprocessor which had a 20 bit external address bus. And therefore, could address up to 1 MB of memory. So, how did we get this 1 MB is essentially 2 power 20, so that 2 power 20 is 1 MB. So, the registers within the 8088 where 16 bit; and it could be divided into various types such as the general purpose registers that is the AX, BX, CX and DX. We had pointer registers which were used to point to string which are stored in memory, so these were the base pointer, stack index, destination index and the stack pointer. So, as we know that the base pointer is used to point to a frame present in the stack, where the stack pointer is used to point to the bottom of the stack.

Then we have the instructor pointer, which points to the instruction that is being executed. And you had several segment registers such as the code segment, stack segment, DS and ES segment registers. Now, in order to load or store an instruction or data in memory what the 8088 processor would do was, it could take a segment base that is one of the segment registers which forms the base, it could left shifted by four and add an offset.

For instance, in order to fetch one instruction from the memory into the processor, the CS register would be used. So, CS register would be this for the code segment. It would be left shifted by 4. And the IP of the instruction pointer would be added to that. In this way, even though there are the registers used are 16 bit, so each of this registers CS and IP are 16 bit, still by shifting it by 4 bits and adding the IP. It was possible to address 2 power 20 memory locations.
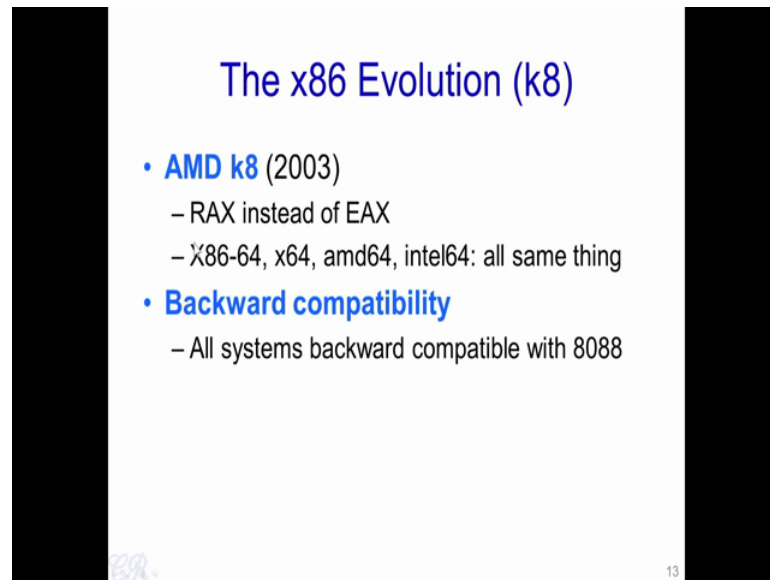
One big step in the Intel systems came when the 80386 was invented in 1995, so these are 32 bit microprocessors. So, this particular processor had a 32 bit external address bus. So, therefore, it could address 2 power 32 memory addresses. So, this was 2 power 32 is 4 gigabytes of memory. So, what this means is that system could have up to 4 gigabytes of RAM present in it. Now the same amount of registers which were present in 8088 are also present here except that now it has it is extended from 16 bit to 32 bit, and because of this extension the registers are called EAX, EBX, ECX, EDX, EBP, ESI, EDI and so on.

There were also a lot more features like the protected operating mode as well as virtual and segmentation schemes, which were present. So, one thing you would notice is that while all the registers were extended to 32. The segment registers continued to be 16 bit. Now Intel ensured that even though we switched from a 16 bit processor to a 32 bit processor, still backward compatibility with the old 8088 systems was maintained. This meant that any software which was written in an 8088 or 8086 processor would straight away run in an 80386 without much of a problem.

To ensure this, even though the registers were extended from 16 bit to 32 bit still programs could access the registers as if they were 16 bits only available. For example,

the AX, BX, CX, DX would access the lower half of the extended registers that is register with 0 to the bit 15. So, this particular feature in the 80386 processor was taken forward to the 486 Pentium and so on.

(Refer Slide Time: 26:10)

Now more recently in 2003, there was the next big step in the x86 Evolutions. So, this was when the AMD k8 was introduced. So, this k 8 moved from a 32 bit processor to a 64 bit processor. As a result, the registers which were known as the extended registers and where of 32 bit, where now extended further to a 64 bits. So, the EAX register which was previously known called in the 32 bit platforms like the 80386 was now called the RAX register, which essentially was for 64 bit. In this way, other registers were also extended to 64 bits. So, in spite of extending from 32 to 64, the companies Intel as well as AMD ensured backward compatibility that is a program written in 8088 would still under certain circumstances will be able to run in the 64 bit platforms as well.

Thank you.