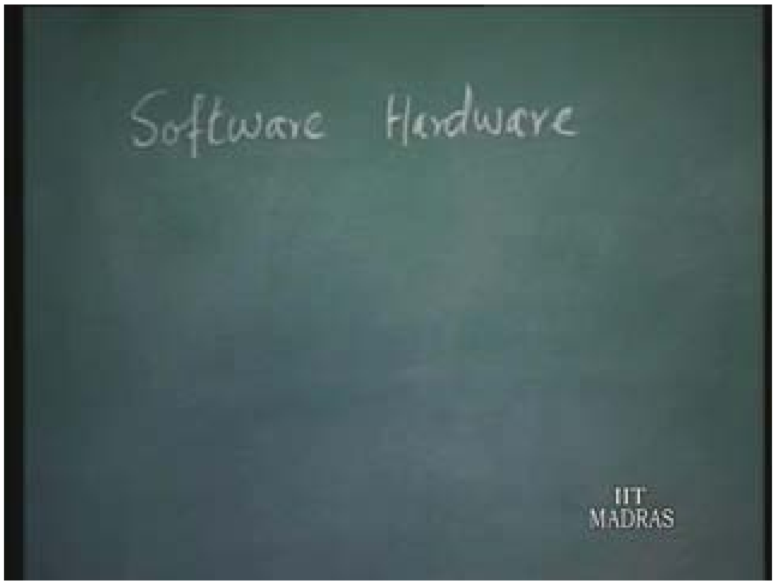
Computer Organization
Part – I
Prof. S. Raman
Department of Computer Science & Engineering
Indian Institute of Technology
Lecture – 2
Introduction to System: Software

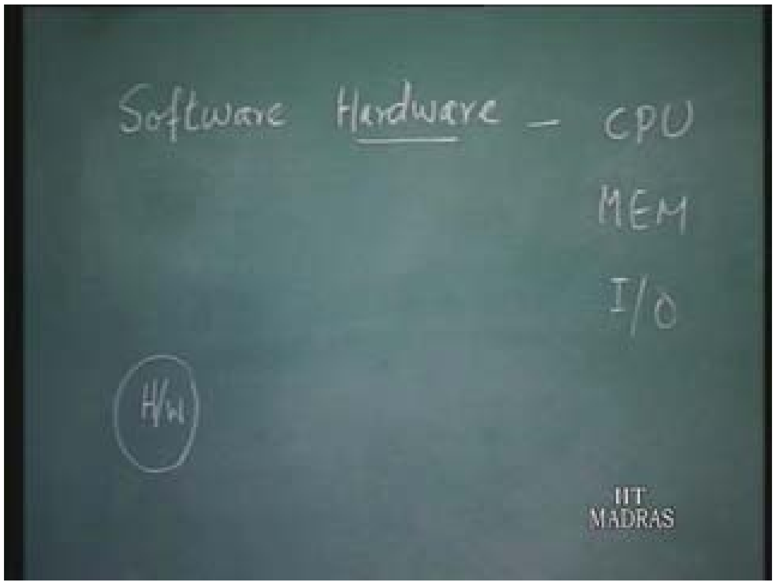
In the previous lecture I helped you get a glimpse, or you may call a bird's eye view, of the overall computer system. In this lecture I will try to narrow it down to again an introduction, but towards what we may call an introduction to software and hardware. We had talked about this earlier too and we will continue, in fact, to keep talking about this throughout this series of lectures.

(Refer Slide Time: 00:01:42)



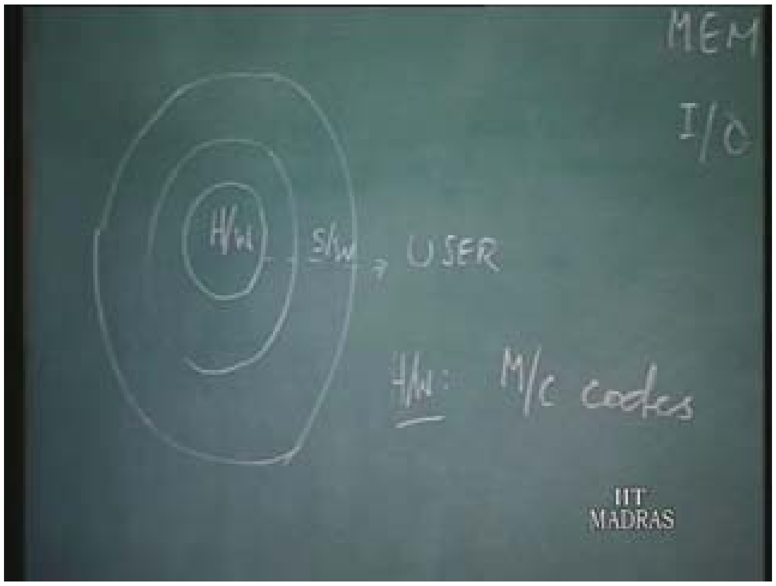
We started saying that hardware, in fact, is the core of a computer system, and that essentially the hardware consists of CPU, memory and I by O. This is the picture I was giving you yesterday.

(Refer Slide Time: 00:02:11)



Then towards the end, I was also telling about the layers and layers of software. So the user who is just outside is getting a view because the inner details of the hardware – the machine-oriented information of hardware – are hidden from him. In fact, that is the main purpose of software; that is, the hardware details are softened and the user is enabled to get some view from his point of view, not from the machine point of view. We also discussed that ultimately any program which the user develops will have to be run by the hardware at that particular machine codes, or M by C; so that's what is taking place at the hardware or the inner layer level.

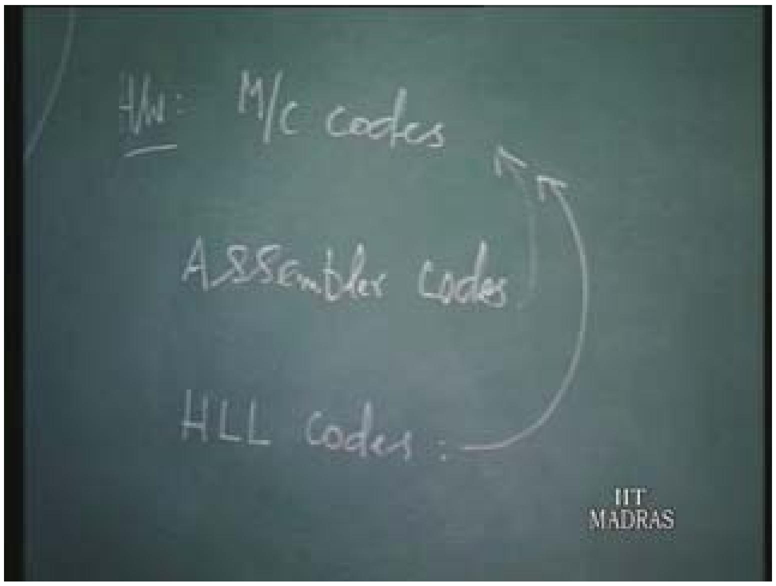
(Refer Slide Time: 00:03:37)



But these codes are very hard to remember, because essentially that will be a string of numbers; if it is binary numbers or string of 0s and 1s, it is very hard to remember. So to enable the user to remember or to aid his memory, mnemonic codes of a particular program can also be written in a different language called assembly language; and the codes developed thus are assembler codes.

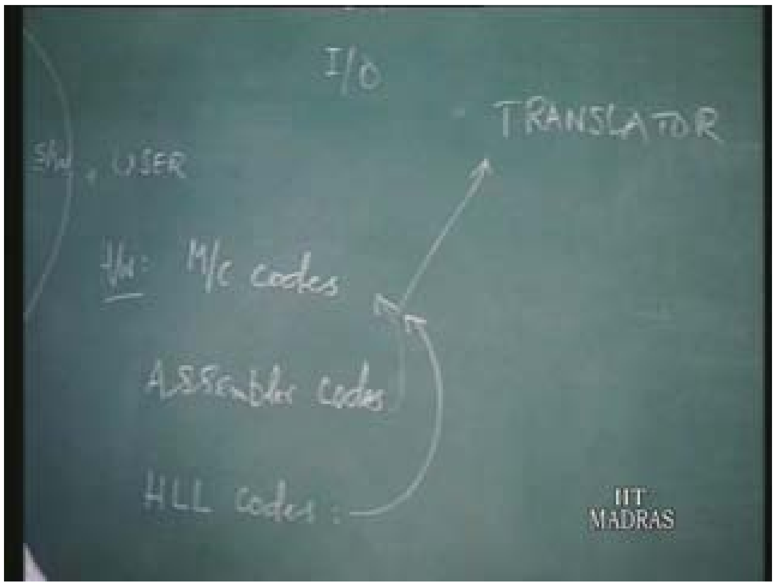
That is one level away from the hardware or the machine part of it; but the whole thing can further be simplified. I said high level language codes at that point are called statements. Statements can be made so that an expression or a function can be given; that is, a mathematical expression can be given making use of the high level language. I also mentioned in their context that the program can be written in FORTRAN, BASIC, or PASCAL. Now ultimately, since the hardware is going to run the whole show because that is in fact the core of it – that is the CPU memory I/O the hardware part of it – whatever the user may be doing, whether he is developing the program at the assembler level or at the high level language, ultimately one must provide some mechanism so that these codes can be translated.

(Refer Slide Time: 00:05:29)



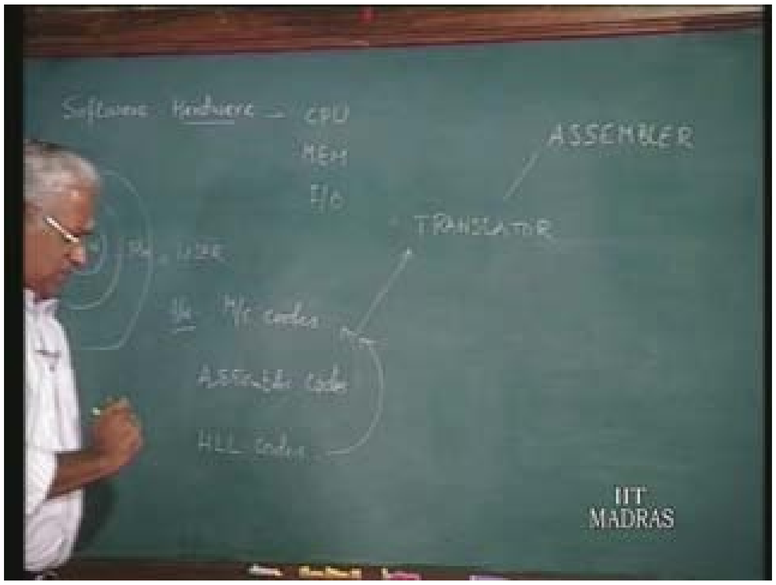
It is not really necessary that one must translate high level language code to assembler code and then to machine code; it can also be direct. So this is one type of translation.

(Refer Slide Time: 00:05:48)



That is, from assembler code to machine code; and then high level language code to machine code will be another type of translation. The point here is that this is in one language and you have the machine language here; so from one language to another language, there is a translator. As the name itself suggests, translation takes place there, from one natural language, say from Tamil to Telugu or English to French or something like that. So the same thing holds good here; however, these translators are somewhat different. For instance, when you translate from this level to this level – that is assembler level to machine code level – they are specifically called assemblers; that is, you translate from assembly language.

(Refer Slide Time: 00:06:50)



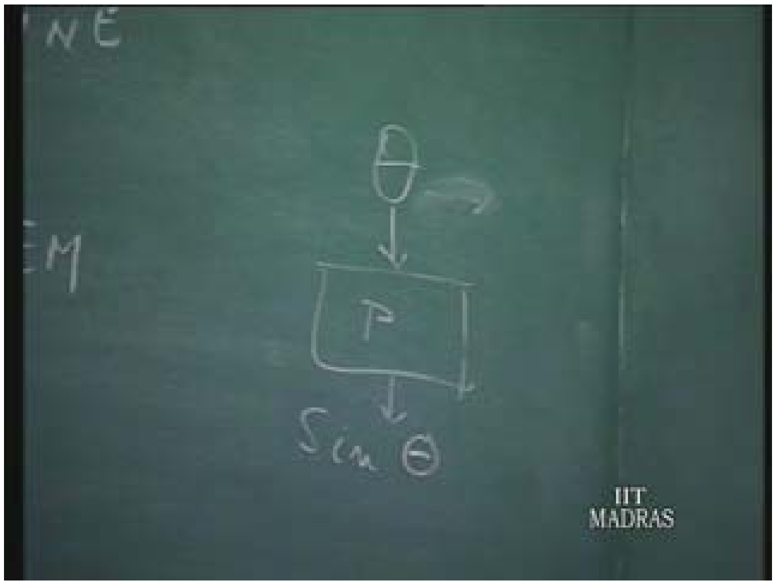
So the program which translates the assembly language codes to machine codes is called an assembler, whereas when these are translated from the high level language code they are called compilers. Apart from compiling, there is another aspect also that is called generally interpreter. In fact, this term is not very different from what we already know. When a person speaks in one language and when the translator listens to it and immediately translates into another language, it is called interpretation. For instance, you would have seen when Russian dignitaries visit our country, there will always be someone between the Russian dignitary and the corresponding Indian dignitary; he will keep listening to this person speaking in Russian and immediately he will keep translating. He is not called a translator, he will be called an interpreter because he keeps listening and immediately translating to the other person. So, we also have translation from high level language to machine code to interpreter – that is one class.

Also, you have high level language to machine codes, which are translators that come under compilation; that is, compilers. So, for instance, if there is a program written in a FORTRAN you need a FORTRAN compiler; if a program has been written in say PASCAL, you need a PASCAL compiler – all these are required so that finally the machine codes can be got and a particular program can be run. Specifically in the case of BASIC, generally we make use of interpreters – there are some reasons for that. In general, translator is the one that that is needed essentially to see that the courses which are in a language further away from the machine codes can be translated. It is not necessary for every person to keep developing the translator.

When a person knows the details of the language and he keeps developing the program, if there is a FORTRAN compiler, it is going to do the translation; so that particular compiler which does the translation from FORTRAN code to machine code is going to be fixed, whoever may be the user. Anyway, that particular FORTRAN compiler itself is another program though. The same way for a program written in PASCAL – you need a PASCAL compiler; so that again is going to be fixed for whoever may be the user. The user really need not bother about developing these translators; now you can see that starting from the hardware, we are going to one level of software which is permanently there with the system, say the FORTRAN compiler, the PASCAL compiler, or a BASIC interpreter, so that the user can write appropriately and then make use of whatever that is needed, translate the machine code so that the program can be run. Subsequently now you can see that though we said the core of the hardware is CPU memory I/O, which I would just call bare machine – with this kind of translator, we are going to one level of software away from the hardware. And now, from a machine, the simple concept of machine, we are slowly graduating towards the concept of system.

The system not only includes the basic core of CPU memory I/O hardware, but also this kind of translators, which are part and parcel of the entire system. Whatever I have said here also holds good – suppose I have written a program and that program essentially is for some given data lets; say angle θ – that's the input to the system. Let me put θ as the angle system. Now I have written a program: call it p, and the output of this is sin θ. Now given this, any particular person who wants can make use of this; he need not keep developing that.

(Refer Slide Time: 00:12:35)

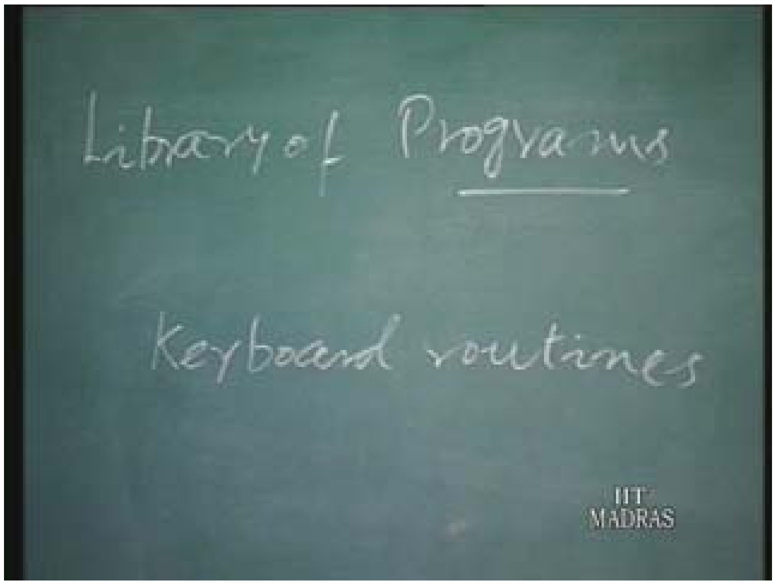


(Refer Slide Time: 00:12:59)

Library of Programs
IIT MADRAS

So you can say that programs like this can be written and we can create a library of programs, which again form part of this particular system software. These programs can be made available all the time to any user. So you can just see that from simple hardware, we can add some extra facilities, so that high level language programs can be written. Ultimately the computing system can run the machine code. In addition, you can also think of other programs. Take a look at this I by O there would be different types of I by O devices; you would have a printer or a plotter. You would a keyboard, and also the display unit. What about the programs for these? For instance, for your keyboard you would need keyboard routine.

(Refer Slide Time: 00:14:21)


Library of Programs

Keyboard routines

IIT MADRAS

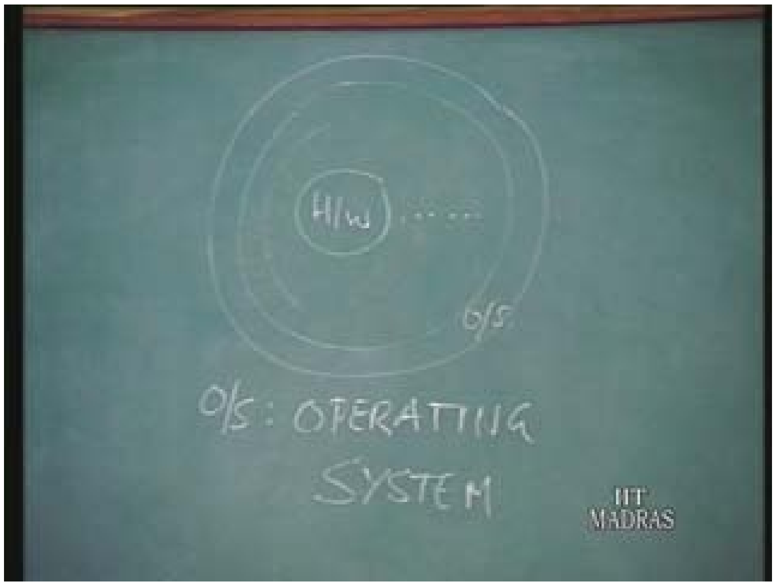
You will see that quite new jargon is developed – now I have introduced the term routine – keyboard routines. These routines are such that you just take them as programmed: we had key board routines, then for display, you might need display routines. Here you will have the display routine. Similarly, for printer, you may be having appropriate printer routines and so on and so forth. Essentially these may be called utilities; which means that these are things which are quite useful or which can be utilized by the programmer. You need these keyboard routines, display routines, printer routines, for accommodating different types of I/O devices.

Any user need not keep developing these again; he need not keep developing certain functions; like this can be made available as the standard thing, as a part of the whole system. In the other case, the system can come apart from this. You can also include this software so that FORTRAN compilation can be done or PASCAL compilation can be done. You have a translator; you have certain special function programs, like this library of programs. For instance, I may have developed some program or a series of programs and I may keep them available for use by others in addition to these, which would be used.

These are again needed basically to see that the system can be used by any user. If you have a dot matrix printer, a dot matrix printer routine and interface will be needed. If you have a laser printer, then you need a laser printer routine and the necessary hardware interface for that. All these are standard; so now you can see that from a bare machine, which just consists of CPU, memory and I by O, the whole thing has graduated into a system, which obviously includes layers and layers and layers of software.

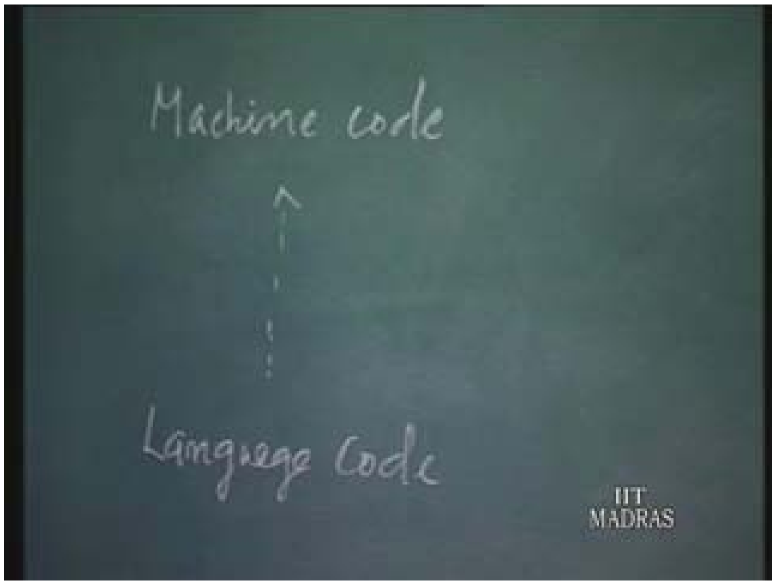
Ultimately, you have to find that any user is not really going to be bothered very much about the inner hardware, but after some layers it comes to deal with an OS or just an operating system; that is, finally it is not just a machine consisting of a collection of few hardware things like CPU, memory, and I by O but also a quite a bit of software, which are available as part of the whole system. Now talking about operating system, this itself has its own commands.

(Refer Slide Time: 00:17:42)



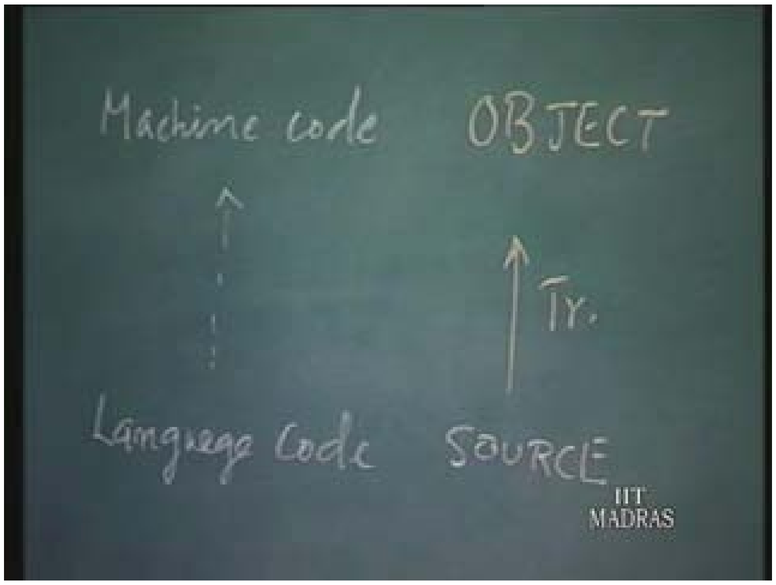
Just as we were talking yesterday about some instructions for the processor or CPU, now we can talk about instructions – they are called commands in this case; commands at the system level. This itself defines a language. Now you can have some hardware and you can have different layers of software, and can come up with one operating system. With the same hardware and different set of layers of software, you can have another operating system. For instance, let us say you have an Intel 5486 system or a Pentium system. That is just the hardware part of it; now layers and layers of software can be added and you can have a DOS operating system or may be a Windows operating system or you might have a UNIX operating system; the core of the entire thing may be just the same. Just these layers of software make all the difference; if one uses this particular thing, then DOS commands will be used; Windows commands will be used; UNIX commands will be used. This is how we graduate from a simple machine to a complex system.

(Refer Slide Time: 00:19:39)



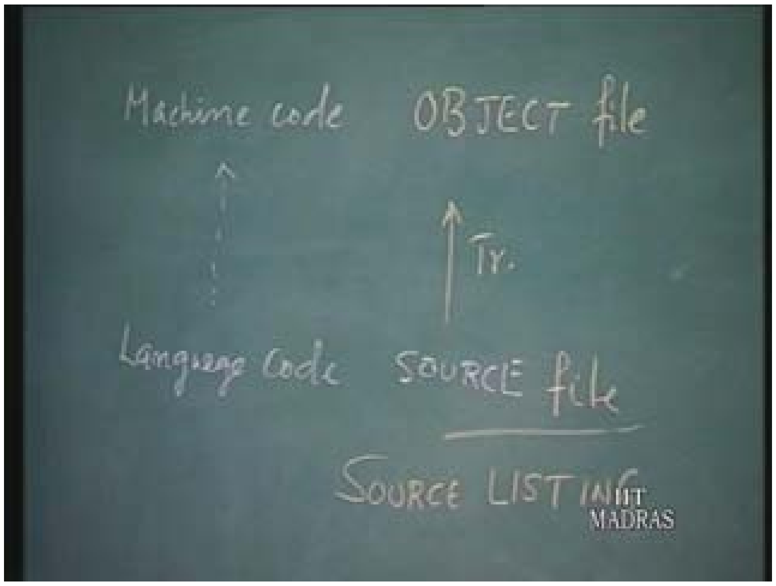
We were talking about a system; let us go back a little further and then see; essentially it said hardware ultimately runs the machine code and then user develops his program or codes in some language whether it is assembly language or whether it is say high level language like FORTRAN, PASCAL or C. Now, since the program is written at this level and that is the starting point, it is some language. So this particular one is also called a source code or source level. When this gets translated, it is going to be machine code and that in fact is called object code.

(Refer Slide Time: 00:20:40)



Later on, you will find that these codes are going to be stored in the form of files. We talk about something like the source file or object file and generally when you take a print-out of this, let us say the original program, then that particular thing is usually called a source listing, which will keep giving information about how this whole program was written, what were the assumptions that were made by the programmer, etc. Ultimately, at the machine level, it is just the object code.

(Refer Slide Time: 00:21:04)



Later on, we will talk about the source file and object file. Going back to the system, I was mentioning that we graduate from the machine to system and then I also mentioned about the OS or the operating system. And then I also said that we can start with any machine, and then depending on what you add, you will be getting different types of operating systems. I also gave you a few examples. Inside in the core, you may have same processor but then you can also end up with a different operating system. So we may say that starting with the machine hardware core, we are going on adding layers and then we are talking about a system with which the user deals.

This is an important thing – in other words, this software, which would be a part and parcel of the system, that particular thing we may call system software. This particular one could be seen as different from what the user actually writes, which we may call as application software because the user is very much concerned with the specific application. So the application software is what may be one kind of user will write. For instance, just like we talk about layers of software, you can also talk about classes of users.

(Refer Slide Time: 00:22:57)

One particular type of user may be developing the system software; another particular type of user may be developing application software; and the third type of user might just be using all these things together. For instance, a particular designer designs the hardware and then somebody adds layers and layers. So the ultimate user is going to be outside of the whole system and then you may be talking about layers of application software, layers of system software etc. Finally you have the hardware, which is the core, and all these details are going to be hidden. What the user will be doing is that he will be making use of those system commands, that is, the operating system commands, and a few commands or instructions, which this particular application package asks him to do. It all depends upon the kind of application; how exactly the details will be worked out.

Now talking about operating system, I would like to mention a few types of operating systems or classes of operating systems. The reason is that it all started originally with what is known as batch processing, because the details of the operating system will vary depending upon the kind of system one has. It all started with batch processing, when the currently available concept of desktop computing was totally absent. That is when in fact the processor got the name CPU or central processing unit because everything was centralized in a computing centre or something like that.

Today we do not talk about computer centre; it is all distributed even to the level of one computer on each desk, and even lap now. Soon we will be having finger top computers too; this is how things are going. But in those days when we talked about a computer centre, all these hardware resources – CPU, memory, IO devices – were all centrally located in one place and people would be submitting their source programs may be as a deck of cards and subsequently as files and floppy and so on.

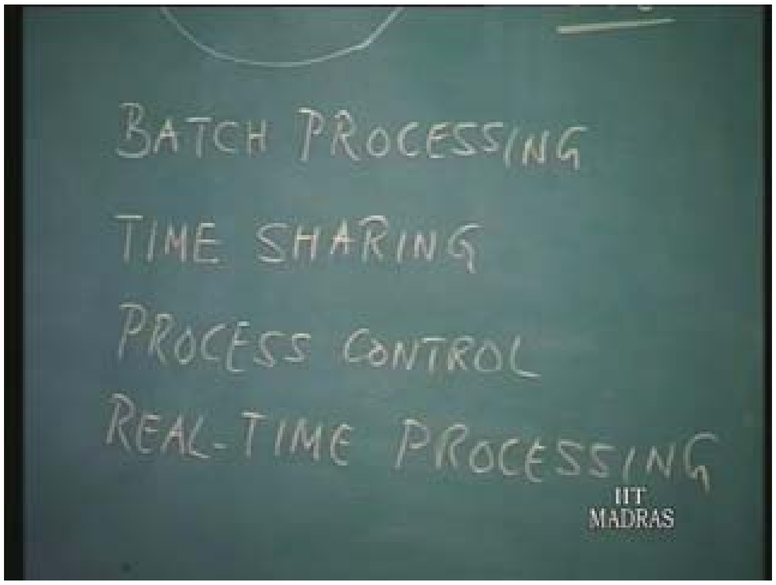
A user's program will be called a job and these jobs will be listed according to some priority and a batch will be created, and one after another, these jobs will be processed. This is how the operating system came about; in fact at that time it was even called JCL or job command language. Job command language is not different from what we are talking about as operating system commands and is related to that. That is, the system was taking only one user's program and it some priorities were allocated to that. Subsequently, when more number of users had to be catered for, what was known as time sharing concept came; meaning the system will turn its attention towards different users and it will share its time among these users.

Here the user may be able to interact with the system. Unlike in this particular case, here the jobs are submitted; they are all queued up. The user in fact is not interacting with the system. In the case of time sharing, usually the user may be able to interact; also, if the user is online, this will be called on-line time sharing system. The operating system for the batch processing was different from the one for time sharing. So depending on what exactly the configuration is, you would see that the operating system characteristics also will keep varying – these are all from data processing point of view.

Remember, earlier I was mentioning the broad spectrum in which at one end we had data processing and at the other end we had control applications – then the computers from very large mainframes came down to mini size. When the mini sized computers first came, the first series of computers were used for what are known as control applications, specifically process control applications. Now you can just see, depending on the configuration of the system the kind of application differs – that's for data processing and this is for control application. The operating system used for process control will have slightly different characteristics altogether.

For instance I will give one typical application – in petroleum refineries, there will be transducers. These transducers will help analog signals picked up from some 1000 or 1024 different points. All the signals will be coming to the particular system and the process will be controlled by the whole thing. So this is totally different from this; here the data comes in real time right from the external world, whereas in these things generally, say in batch processing, it is all data which is frozen in time sharing – it could be either this or that. Process control: the moment they saw that computers can be used very effectively in process control, then the concept of real time came; that is real time processing. In the sense that you are not only taking signals from the external world, what we may call live signals.

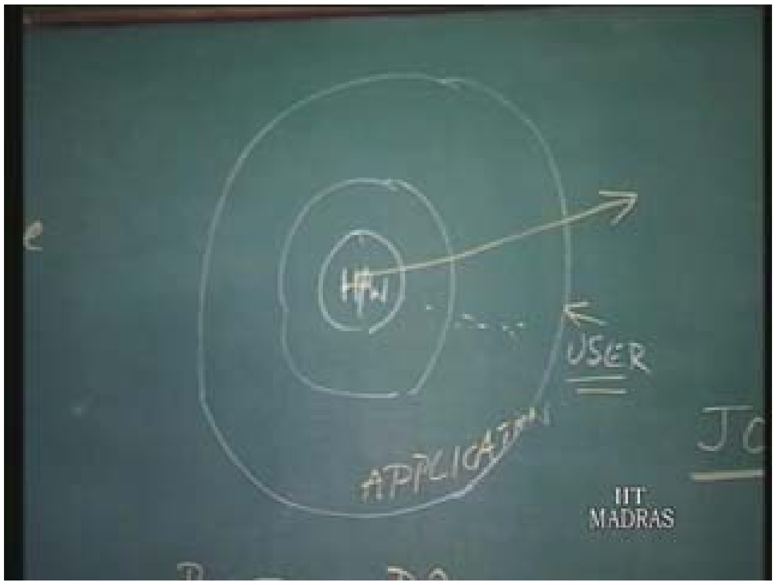
(Refer Slide Time: 00:29:49)


BATCH PROCESSING
TIME SHARING
PROCESS CONTROL
REAL-TIME PROCESSING

You may also take the signals, process them, and return the processed information back to the system in real time. Real time is a very relative definition: there is no such thing as 1 second is a real time and 100 millisecond is not; it all depends on the application. So the system can wait for a particular period; within that period if the processed information comes back, then it is real time. So we now have batch processing system, time sharing system, process control – this is usually an online system – and then real time system. The operating system for each of these will be different because each of them has different characteristics. The latest are distributed processing and distributed systems. These have come about mainly because of desktop computers and the process can be distributed. Then we talk about network system, and of late, we have to talk about an operating system for the network or network operating system (network OS). Depending on the configuration of the system and the application you would have different types of operating systems.

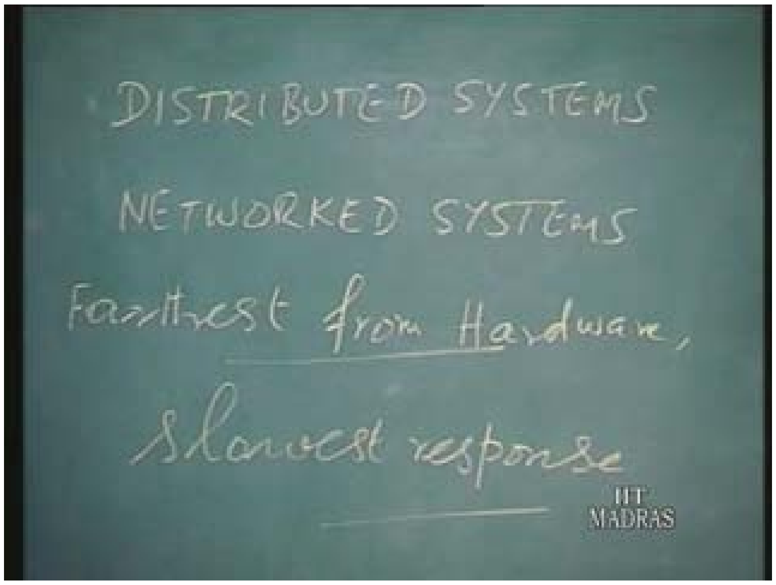
As I have said earlier, we can talk about layers and layers like this. The inner one is hardware and the outer one we can put it as application, as a part of the system. There is some application software which the user interacts with. Right now you can see that the inner one is hardware. When you go out, it is going to take more and more time of a program. If it is run at this level, it will take less time; a program, when it is run at this level, takes the maximum amount of time.

(Refer Slide Time: 00:32:05)



Why? Because this one will call for some program execution at this level, which in turn will call something at this level; and this in turn will call at this level – this is how it keeps going. So if that is running, an object program would be the fastest and in some of this, real time processing will be required. That is, at least some portion of the program will have to be written and run only at the innermost level or the object level, because it is a time critical application – that's what real time is about. So the farther it is from the core or the hardware, slower will be the response – this is important.

(Refer Slide Time: 00:33:29)



DISTRIBUTED SYSTEMS

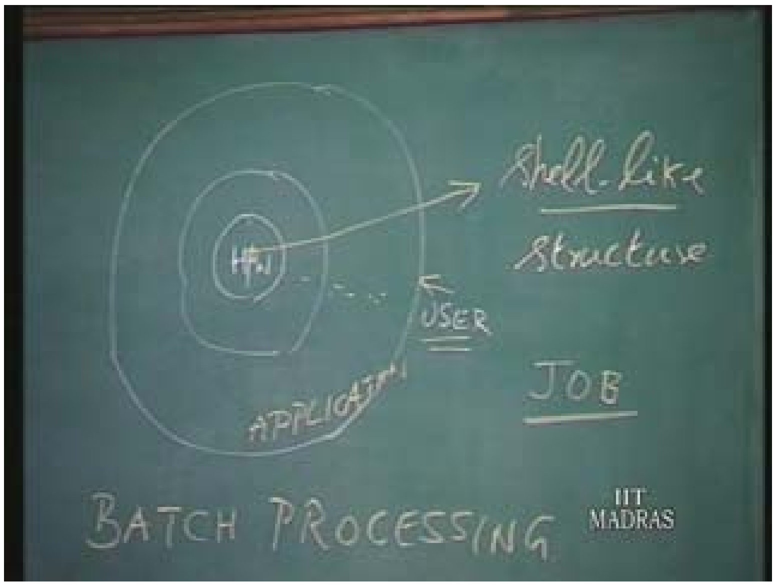
NETWORKED SYSTEMS

Farthest from Hardware,

Slowest response

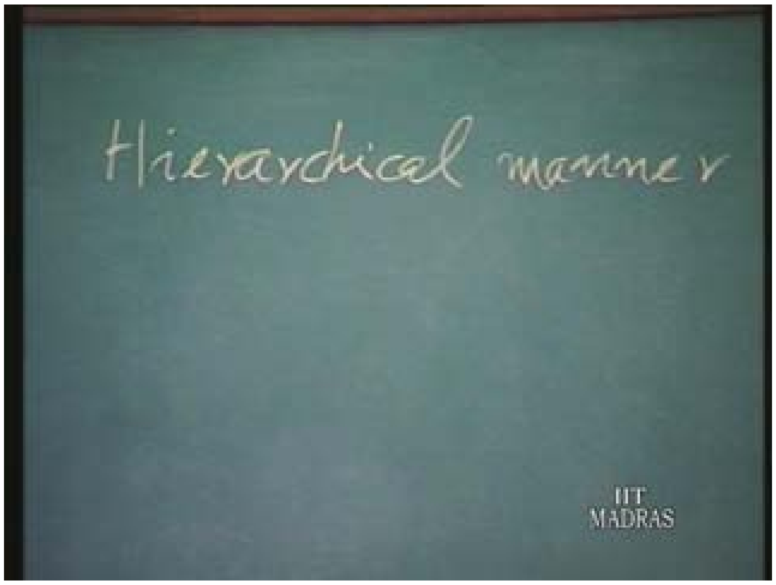
This particular thing will have to be had in mind when we develop the programs. So there is no point in going on adding layers and layers if ultimately you find that the response from the system is going to be very slow. For instance, let us take one application, let us say translation. Suppose I am interested in developing a translation system; let us say from one language to another language let us say from Hindi to Tamil or something like that. Now if the Hindi input is given and the Tamil output is going to come 2 minutes later, who is going to wait for it? Is it not? Similarly, this is one way of looking at it; that is, we are talking about shells – that is layers and layers, very much like an onion. This is a shell-like structure; that is, you have a shell and then you think in terms of inner shell and outer shell and so on.

(Refer Slide Time: 00:34:36)



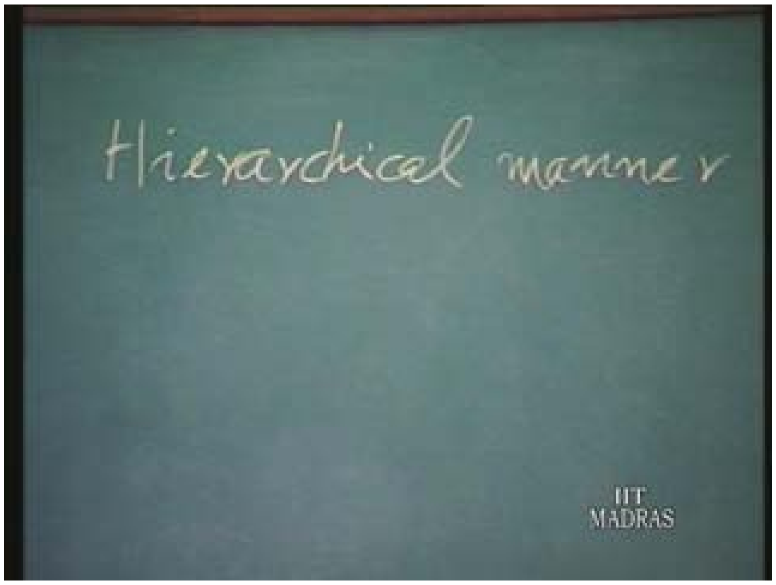
So we can go on talking about layers and layers of software. There is another way of looking at it, which is a hierarchal manner. That is the same thing we are going to write in a slightly different way.

(Refer Slide Time: 00:34:57)


Hierarchical manner

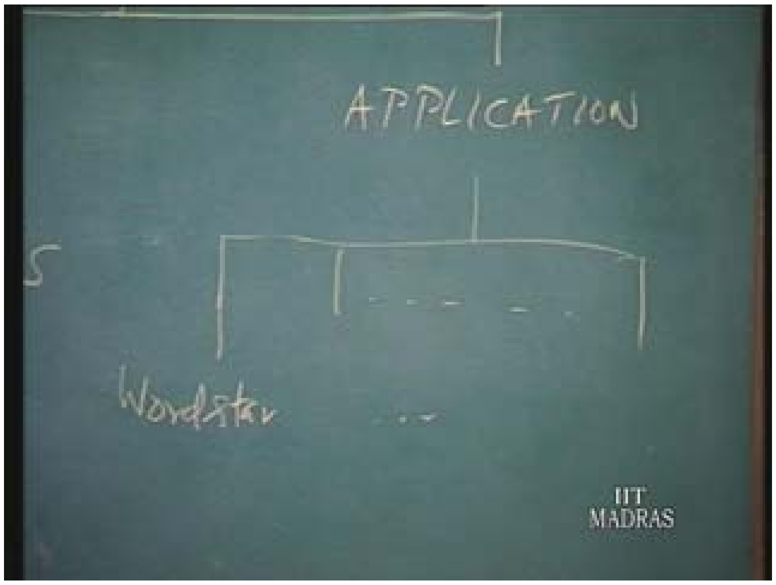
I am mentioning this because there is absolutely no uniform way in which we can describe this. For instance, it is not as if this one will not have direct access to this. At this level, it need not really go through all the shells; some of these may directly access also. So it is not that it is not possible. In the case of onion, for instance, the outer shell has to be removed totally to go to the inner shell – it is only some kind of imagery we are trying to build. Now talking about just the software, I said about two things: that is system software and application software. You can mention the two things as this: at one end we have the system software and at the other end we have the application software. Then this system software at the highest level will include the operating system and from the user point of view, there may be a translator. That is also very much part of the system and I already mentioned the translator can be a compiler, or it can be an interpreter. When we are talking about compiler, for instance, I can talk about the FORTRAN compiler or a C compiler or a PASCAL compiler and so on and so forth. It is in the same way in the case of interpreters also.

(Refer Slide Time: 00:36:53)


Hierarchical manner
IIT MADRAS

Now this is at the system level. When we go to the application level there can be many things; in the same way there can be many levels – I will indicate just a few things. For instance in office application, one would be using a word star like this; there can be many things here.

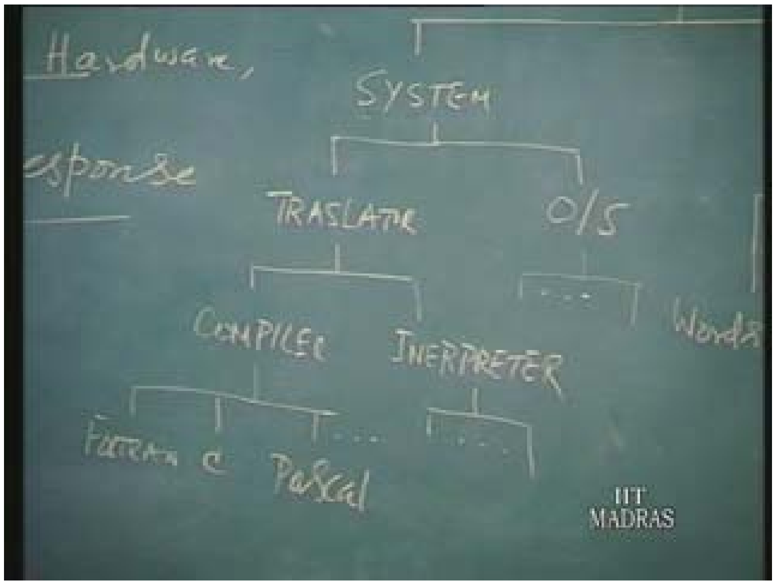
(Refer Slide Time: 00:37:17)



We can talk about spreadsheet and so on and so forth and somebody else may be using a dbase program package. So here you will be interested in knowing that though they call it software, dbase application program, in fact is a language by itself.

So when a user is using the dbase application package at this level, he is dealing with one language, one specific language, which is specific to that particular type of dbase. So this is how things keep building. Now you must know that there are many ways of looking at it – the shell as I have given you and then in the hierarchical manner also. Now for instance, we can further divide the operating system into many things.

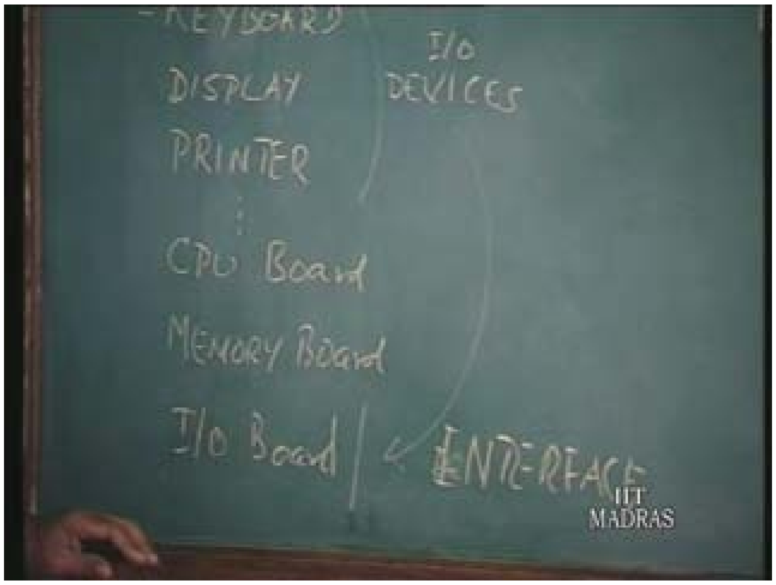
(Refer Slide Time: 00:38:22)



There is something dealing with file handling – because ultimately there are different types of files. The routines or the programs needed for handling these files and the different devices, the routines for handling the respective I by O devices and all these things are included under that.

Hope now you have some picture about what essentially the software part of a computing system is. Earlier I said in this lecture you will get a bird’s eye view of software and hardware. So far, we have seen the software; now let us take a look at the hardware. As I said earlier, any system must be studied from the user’s point of view. So we will follow the same thing here too. What does the user see? He would be interacting with the system and so if he sees physically a keyboard, display, then printer, and inside the system he will also be watching it, if you open some system or CPU board or memory board and I/O board – essentially, because these are devices or I by O devices they cannot interact with the CPU and memory, and these devices need the extra interface, these I by O boards provide the necessary interface.

(Refer Slide Time: 00:40:02)

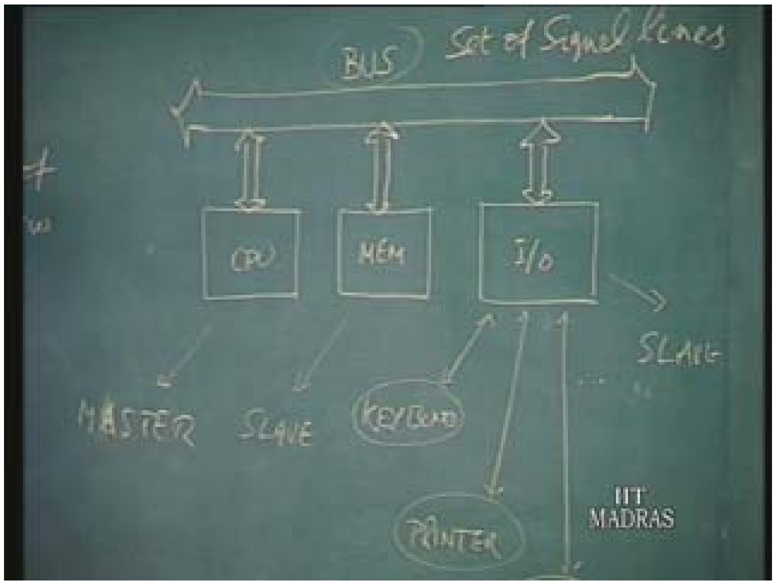


The CPU memory forming a system with a set of interconnecting lines is generally called bus. The I/O is very much part of the system; what is on the bus is actually the I/O interface and then the devices are there as a part of the real world, through which the user interacts. There are different types of keyboards; similarly there are different types of displays, printers, and so on. And what is more important is we talk about the CPU and the memory. In the previous lecture also, I was mentioning that CPU and memory will interact at electronic speed, whereas these devices, which are in the external real world, will not be able to interact at that level and so we bring this interface. That is why it is also called an interface. We bring this interface so that at this point, the communication between these will be at electronic speed, whatever may be the speed at which these devices actually operate.

Now, we have to talk about a processor that is the CPU, and then the memory, then the I/O, and also now, for the first time, you have been seeing that we are talking about the bus or the set of signal lines. This in fact is the total system and for the moment we will forget about the software because that will also have to be studied from a slightly different perspective. When we talk about this, obviously the signal lines must be defined; that is, the sequence in which the signals go over the lines must be defined. When we have a common set of signal lines like this, then obviously there must be something which controls the whole show. Generally what we find is that the CPU is usually the master of the bus, and that, in fact, will be orchestrating the flow of the data or the signals, and memory is always the slave because it has by itself nothing much to master over. The I by O is usually the slave of this entire thing. I said usually CPU is the master; I by O is usually the slave; memory is always the slave.

So in this particular way, we would be talking with reference to this set of signal lines – the CPU will demand memory for something and the memory as a true slave will respond, or the CPU may demand I by O and the I by O usually as a slave may respond; but there may also be situations where I by O may demand, in which case, generally it is the CPU, which will be responding for a short while. But usually this is the master and usually this is the slave; occasionally this may become a master and occasionally this may become a slave. These are the different I by O ports; you may put it as I by O transfers, which may take place, making use of the protocol of the signals, which are defined for these lines or the bus.

(Refer Slide Time: 00:43:10)

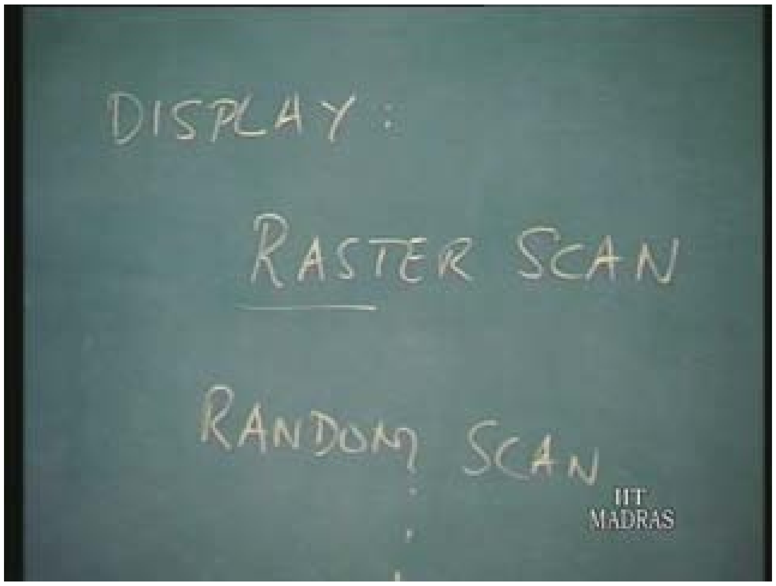


Now let us go back to these interface boards. Depending upon the type of keyboard, the appropriate interface is required. Talking about keyboards, we can talk about, let us say, just the keyboards alone; we may talk about some keyboard, which generates a code and call them as ASCII codes. ASCII codes – now what does it mean? When a key is pressed on this keyboard for that particular key there is one code, unique code, which is generated. That is, for a particular key there is a unique code that is generated and that unique code in this particular set is called ASCII. ASCII, if you want to know, is American Standard Code for Information Interchange.

When A is pressed, there is separate code for it; there is another specific code for B, and so on. CPU is only looking for that code, which defines the set of signals that go to that. So let us say the CPU demands from the keyboard and the keyboard responds. And now how will the keyboard respond? The user will respond by pressing a key and CPU collects the code by looking at the code. By looking at the code the CPU knows what key has been pressed. Now keyboards which make use of this are called ASCII keyboards. There can be different types of keyboards – it can be a pressing type of keyboard – there are different ways in which the keyboards can be designed. We will not go in much detail at this stage.

Right now depending upon the keyboard type, the interface is required – that is more important. Similarly the printer may again be of different types – I will just list out a few – it may be a dot matrix printer, there may be a line printer, there may be a laser printer, or there may be an ink jet printer. Depending upon these types of printers, again the appropriate interface circuitry is needed; that is what we are talking about. Whatever may be the type here, the communication between the CPU and I by O must be uniform; only then, at this level, we can talk about some standards. Otherwise there will be a problem. So the CPU to this interface must be standardized and all the idiosyncrasies of the respective types here will have to be absorbed here. In other words, we are going to see that the details of these devices or kinds of devices will be hidden from this side. That is what precisely the interface is doing – the same thing holds true for display also, for that matter. Display can also be of different kinds – normal display, scan display, and so on. We will talk about more about these in the next lecture.

(Refer Slide Time: 00:48:42)



DISPLAY:

RASTER SCAN

RANDOM SCAN

IIT MADRAS

Now, depending upon the types of displays, again this interface will have to be defined. So precisely whatever I was talking about printer, holds good for the display too. We will cover more about this display and some more about the hardware with the details in our next lecture.