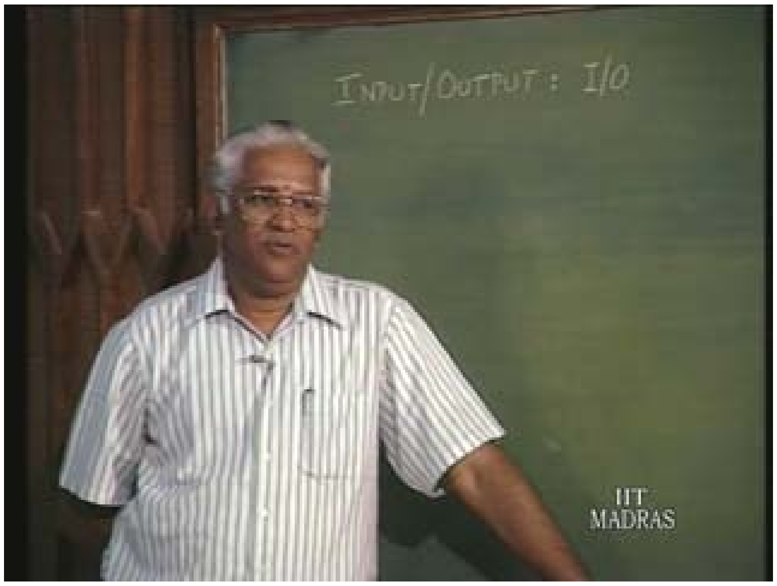
Computer Organization
Part – III
Prof. S. Raman
Department of Computer Science & Engineering
Indian Institute of Technology, Madras
Input/output
Lecture – 24
Programmed I/O

(Refer Slide Time: 01:02)



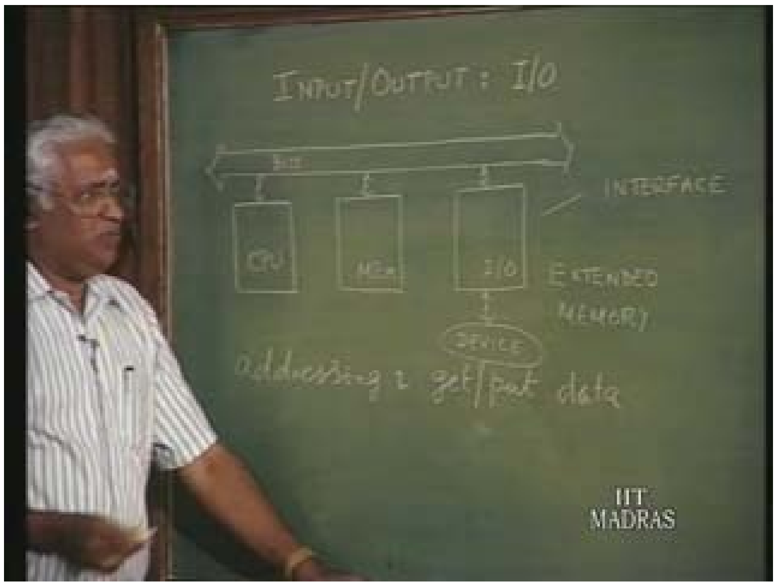
So with this lecture we are starting part three in this series. In part one we took a look at the details of the processor and in part two the memory. Now this is the third one, that is, the I/O – in short input/output. What are the various things that we knew earlier or we had seen earlier? We saw that CPU is essentially the master of the whole situation. Then you have memory; of course it is not just one block; we are talking of hierarchies and now for the present we are talking about the third item, that is, the input/output. Really speaking we have possibly a set of devices or a set of transducers, etc., coming under this input/output category. Now before we go to the other end of this block, let us see whatever we knew or had seen earlier. That is, interconnecting this we have the system bus and that is the system bus. These interact; we had already seen the CPU–memory interaction and we had also briefly talked about the bus, although not in detail. We will cover that particular thing later on. Here what we had said is that a bus is a set of signal lines and if you can standardize, then you can have any type of processor based on which you can construct a CPU board and you can accommodate different types of memory systems also or memory subsystem to be specific. So we need to standardize this particular one.

If I say the bus needs to be standardized, then obviously it implies that if we have a set of signal lines for CPU–memory interaction, then the CPU–I/O interaction is not going to be very much different because we have a set of bus signal lines. Of course, we may not use the same set of signal lines. But it will be somewhat similar. So what is that we knew about the CPU–memory interaction? CPU is the master of the system and addresses the memory; also it is the master of the bus; the memories as a slave responds, and let us say it sends the data to the CPU or memory may receive the data from the CPU if the CPU sends. It all depends on whether we have to read some data from memory or write some data into the memory. Now you can see a similar thing here; we are talking about this third block, that is, the I/O. It is possible that from this block, either some information gets in or some information gets out. Similarly, like reading from memory and writing into the memory, we can talk about inputting some information or data from I/O to CPU and CPU outputs something to the I/O. So the read–write of memory is similar to input–output with reference to this particular block.

We will talk in detail and also stress upon the differences. Now you had seen something similar, and if you recall from lecture one, I was telling I/O is nothing but extended memory. I had also told you why. What was that? Essentially memory calls some data, which had already been acquired so we can refer to it as frozen data and CPU may get the data from the memory, let us say. Instead of that, I/O can supply live data or raw data, which may be created as and when the external world prepares it. So, whereas memory holds frozen data, we can say I/O holds live data. Of course it holds the data means through I/O, CPU will get the data that is prepared in the external world. In other words, what we are talking about is that this I/O sitting on the bus is not really the I/O device by itself. Just like we are talking about CPU to memory, we are talking about some electronic circuits; that is, CPU generates some information or some request and memory response, and then we are talking about different speeds, and then we said the cache subsystem in the memory is the fastest one. We were also talking about other memory hierarchies also; that is, from the fastest to the slowest.

Similarly, here also we can talk about the slowest device and fastest device and so on and so forth. As far as the interaction is considered, whatever CPU was doing with memory, it can do with I/ also. In other words, the CPU was addressing the memory – now, instead, it can address the I/O, provided we create the necessary facilities for addressing. It was also getting or putting some data, I would say, addresses, which was read or write or input or output. So in the same way, I/O also can be dealt with but then there is the main problem, because we encountered this problem even in the memory. That is, we said the CPU will not be efficiently utilized in case it has to deal with slow memory, and most of the devices, I/O devices, will be slow and so the CPU should not get directly involved with this particular one. So that shown as I/O is some interface circuitry, which can be operated at electronic speed but in this case, the CPU gets a response of the electronic speed and then it has to buffer, act as a buffer, between the system bus and the actual device. So for each I/O device, there is going to be some interface circuitry. We say interface mainly because it is the one which comes in-between the actual device, which does the input/output, and the system bus, on which the CPU is there.

(Refer Slide Time: 09:29)

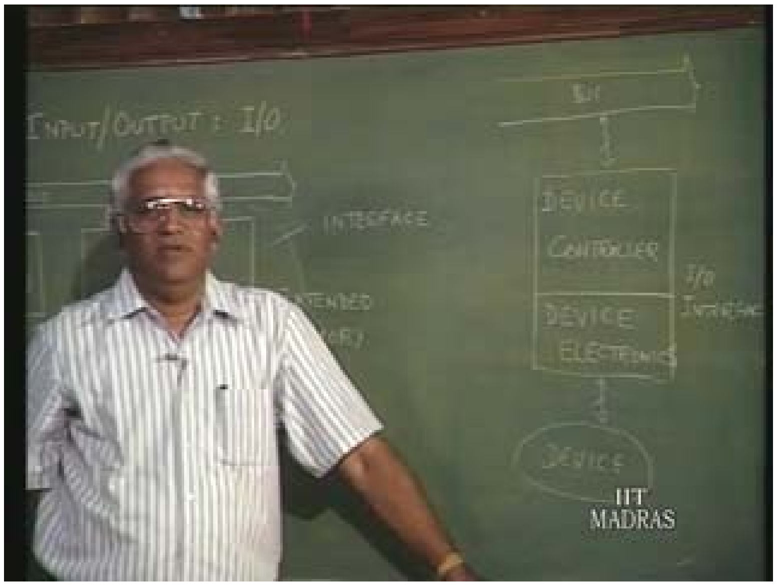


Essentially the CPU deals with the device, not directly, because we do not know about these devices. These may be electromechanical devices or any other type, whereas here what we have is a pure electronic system, and then we do not know the kind of signals that are generated or the data generated by this – it may be voltage signals or current signals; it can be any other type also, whereas here what we want is essentially something like digitized binary data. All these things will have to be converted and at this end, that is, at the bus end, we can talk about a standard. So whether you have one type of device or another type of device on the bus, the data that is transmitted between the device and the CPU will have to be standardized, which means the interfacing I/O has to observe what we may call as the variations of the device or idiosyncrasies of the device because some device may be generating only 1 bit at a time and it may be doing it very slowly, and some other device may be generating lets say 1 k bytes of data at a time; it all depends. So all these things will have to be observed so that at, this end, this will be uniform.

Though we say CPU–memory interaction is similar to CPU–I/O interaction, remember that I/O speeds are generally low. Now when we say generally low, it means that there can be some devices, which are very fast. With a faster device, there is not much of a problem; the CPU itself can directly deal with it. Only when it becomes slow, we have to see that the interface takes care of that particular problem. Now let us go into the details of this – what do we have here? Essentially, we can say that there is an interface from the bus end and the I/O interface – let us not worry much about what is going on here – and here we have the device. Sometimes it may not be device also; it can be a transducer. Now we know that this is an interface. Essentially we can think of two parts of this interface – I said there are different types of devices, for instance, this might need some mechanism by which a floppy disk will have to be rotated. If it is a tape, the tape will have to be bound, unbound and the whole tape will have to be moved. If it is something like a disk that is going to be a moving arms, where exactly to position all these things – these are also part of the I/O device problem.

So there is one set of problems, which we will not go into detail, but I will just indicate. Generally we can talk about the device electronics part. All that I mentioned just now can be taken care of having the appropriate electronic circuits, and that is not really contributing to the data transfer that takes care of movement of an arm by the necessary angle for moving or rotating a drum or rotating a tape. So you have a device electronic part of it, which takes care of purely the device specific information to see that the device can be operated. The one which we will be concerned with in our discussion is the other one that is at the bus end or what we may call a device controller.

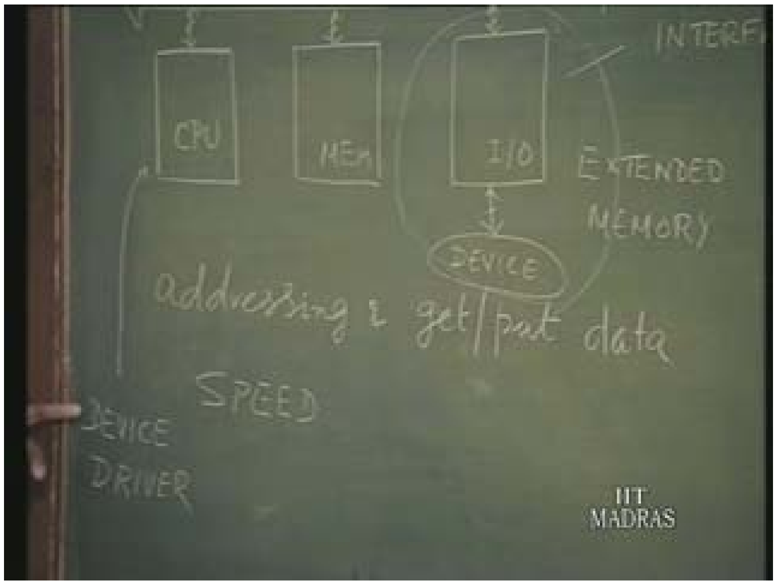
(Refer Slide Time: 15:04)



Essentially it is the device controller with which any processor or CPU will be dealing with during the data transfer. So what are the details here? If we follow the same thing as CPU, which addresses and then gets or puts data or inputs or outputs data; getting data is inputting because we always talk with reference to the CPU. When you say read CPU reads data from memory; when you say CPU write, CPU writes data into the memory. Now CPU gets the data from I/O; that is, the inputs are put in by the CPU and the data in I/O is the output. So for this particular thing addressing and getting the CPU will have to actually deal with the controller part of it. Both are in fact electronic circuits; but this is the one which will take care of one side of the bus part, whatever standard bus signals, and on the other side, it will have to deal with the device specific thing. But then the interface circuitry also includes the device electronics, which is specific to that device. This also will be specific but then at one end it takes care of it. This is the one which will deal with the logical part or the data part; this is not going to deal with the data part; it is going to deal with the device part of it. Now having talked about the device controller and having also said that the CPU–I/O interaction can be similar to whatever was there with reference to CPU memory, then the device controller possibly will be addressed by the CPU. The CPU will address and then the data will have to be transferred in or out.

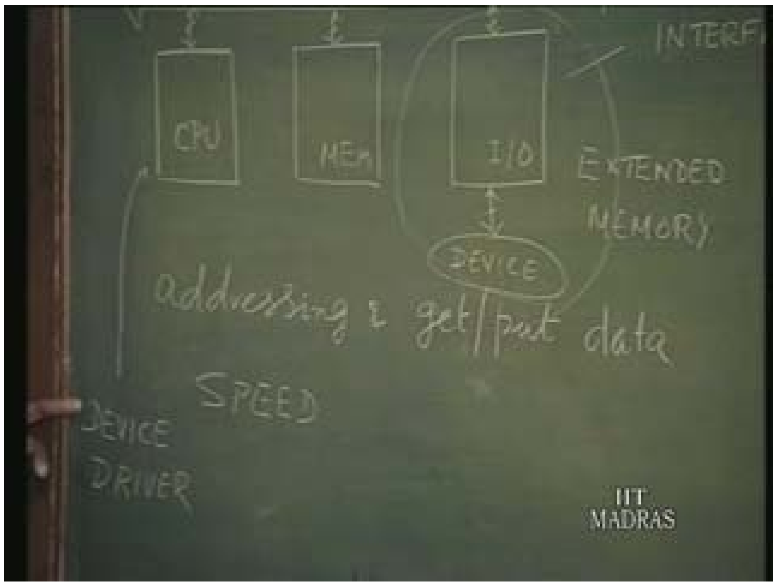
So we have to create some kind of an addressing mechanism, meaning, each device must be identified with a unique code, that is, the actual address, some unique code. And then each device will be generating the data. Now that data will have to be assembled possibly here and then sent over because, in case this is the bus which can transmit say 16 bits at a time and in case we have a device which generates 1 bit at a time, imagine the situation. Suppose the device generates 1 bit at a time and here the bus supports 16-bit data part, at the exact moment that the device generates 1 bit, will it be put on a 16-bit wide bus and sent? It is going to be wastage of the bus bandwidth. So, it is meaningful for the device to generate 1 bit at a time; 16 times. So that device controller will assemble that bit and then the moment 16 bits have been sent, the 16 bits will be placed and then sent over. These are the things the controller will have to be taking care of. Now when I say this, do not forget that there must be some program which is running at the processor end also because, after all, the CPU is the master of the whole situation. So the corresponding software to make this particular device operable or addressable and so on will be called a device driver. This, in fact, is the software.

(Refer Slide Time: 19:35)



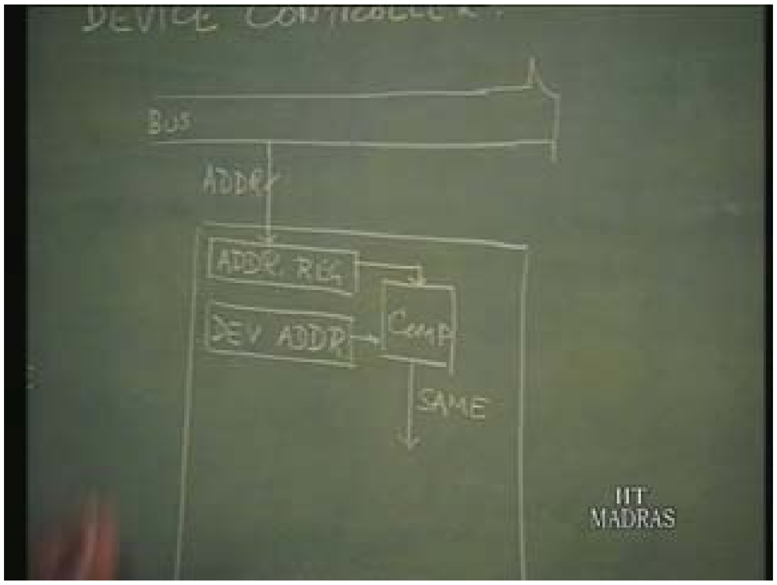
So device driver is software, which will be run at the CPU end and this particular device driver will be different for different devices. Now you can see that there has to be software at the CPU end for a given device and there must be an appropriate interface circuitry for that particular device. That interface circuitry would consist of specific electronic circuitry for ensuring that the device operates, which will take care of mechanical movement and all those things and the electronic part, which is concerned mainly with the data or the logic part of the data transfer, will be the controller. So it is actually the device driver, which, when run, will generate a code, which will be placed on the bus that helps a particular device to be identified and something more will be done; we will go in to the detail.

(Refer Slide Time: 19:37)



And if the data is ready, if it is an input device that data will be passed over the bus and the device driver will receive and route it to the necessary registers. Now let us go into some details of the device controller. What is that? First thing is let us not forget about the bus being there; that we should remember all the time. I am going to expand the interface part that is, going into the details of it. I said let there may be a unique code for identifying the device; it is similar to each memory location having a unique address; so that is first of all needed.

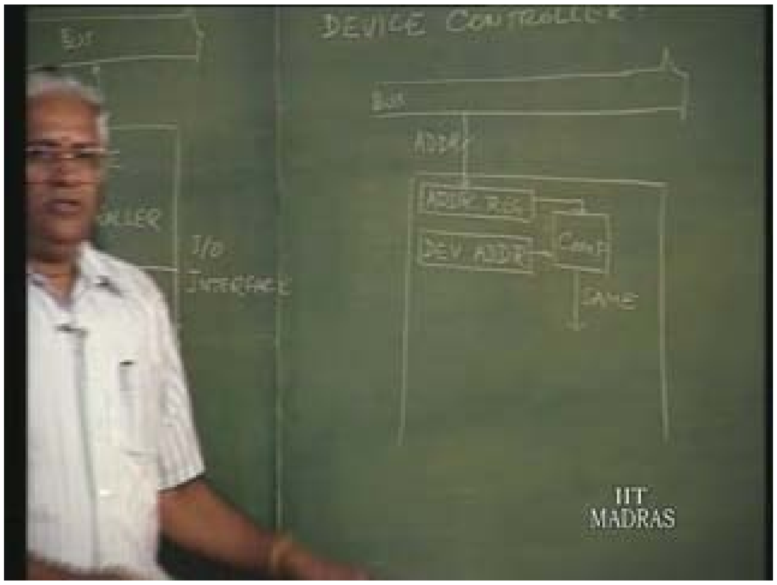
(Refer Slide Time: 24:27)



So we will call that something like a device address. There must be a device address and at the other end we have the device; let us not worry about it.

We are just looking into what is going on at this end. If necessary, we will expand the set of signals that are there; so that is why I am leaving this blank. Let us see: the CPU places an address and that address will have to move in. For instance, if this is so, then it is possible that I call it device address, in which case let us say the incoming address goes to some register, address register. That is, whatever address that has come or that has been generated by this CPU, comes over and it goes into this. Now remember that there may be many devices, so there may be many interface blocks. For instance you can assume that when the CPU generates an address, it goes into address registers of all the devices on this. Then we have a unique device address for this particular one. So this is the address received from the CPU side, whereas this device address is unique for this device. If you have 10 devices, we would be having 10 different addresses; we will just call it address 1, address 2, I/O 1, I/O 2, I/O 3, and so on – something unique. So obviously, these two will have to be compared; there will be a comparator, I mark it here. With a comparator, these two will be compared, and suppose they are the same – this address and then this unit – suppose they are same, then we say that this device has been addressed, and this of course can happen uniquely only to one of these ten different blocks.

(Refer Slide Time: 24:27)



So this particular signal will be up only for one of those 10. That is one part of it; that is, we are saying that the CPU is addressing and now one of the devices addressed is going to respond. Now what will it respond with? There are two things: first we have to worry about the speed thing. I said compared the memory; the devices are generally slow, very slow, in fact. For instance, if the cache memory can respond in 10 nanoseconds or 20 nanoseconds, assuming a device, a keyboard, suppose this particular interface is a keyboard interface and the device is a keyboard. If someone types 50 words per minute, we talk about 50 words per minute; will you work out a simple calculation? I do not know very much about typing speed; what is normal; 50 words per minute and assuming each word has say some 6 letters, 6 characters, we talk about 300 characters per minute – 300 letters or characters per minute. Now a minute consists of 60 seconds; so how many characters per second would that be? We have 60, so 300/60 would be 5 characters per second.

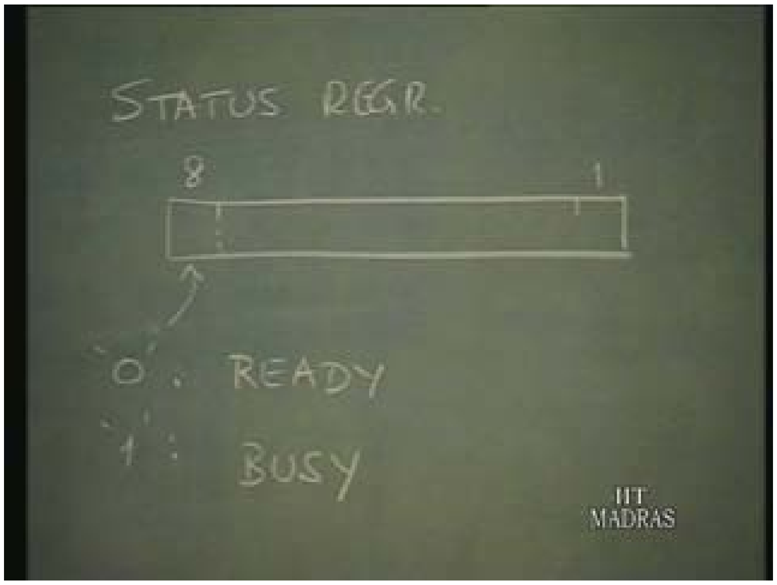
In 1 second, there can be only 5-character strength; that is, 5 ASCII codes. Remember 5 codes will be sent. In 1 second, there will be 5 codes that are sent, whereas here, 1 code will be sent in 20 nanoseconds. Compare this: 20 nanoseconds, whereas here, in 1 second, only 5 codes will be sent. So look at the enormous speed difference. Of course we can also talk about a fast device; so let us work out for fast device. Let us take display, which is same thing as TV. On one line of the display, assuming there are 80 characters and for each character there will be 5 dots. So assuming one in between each interval, that is, inter-character space of 1 bit, you have 6 bits; 6 × 80 is 480 bits; on each line you have 480 bits. That is, just 1 dot and the whole thing is going to be scanned in about 60 microseconds, so in 60 microseconds 480 bits will come. I am just assuming 60; it will be 50+ something. In another words, in 1 microsecond, we are going to have 8 bits, you know roughly 1 character for 1 microsecond; whereas here we had 20 nanoseconds for 1 character.

Here we have to find out 5 characters in 1 second. So with regard to speed, actually this is the way you have to analyze whatever that may be. We said we do not have a device which is as fast as the cache memory; that is the very fast. We have to remember that speed matters. Now why did I talk about speed? That is because the CPU is going to address and then one of these will say I am the addressed device. That same signal will come. Right at the time the device may be doing something; because of this enormous speed difference, we do not know what the device is doing.

So the first thing that has to be checked is what exactly the device is doing – let us say if it is an input device, without loss of generality I can just check that this is an input device. The same thing holds good for output also. Let us say if it is input device, then that device is busy generating a data, which can be passed on to the CPU. At that particular instance when the CPU addresses, the device may be busy preparing the next bit of data or byte of data. So you need to have some information which indicates the status. Now what we are talking about is a status register, which will tell whether the device is busy or whether it is ready with the data. Without much generality, let us assume that there is an 8-bit status register and I will just assume 1 bit in that. Let us say this is the most significant bit, which, if it is 0 then it is ready; that is, the device is ready. If it is 1, it indicates that the device is busy.

So there is a status register; actually it is nothing but a device status register because it indicates to the CPU whether the device is ready with the data or whether device is busy. So we must have a status register. The moment the interface recognizes that the CPU is addressing, 1 of the 10 is going to respond, at that instant it will have to check this particular one. How is the checking to be done and who is going to do the checking? This is where different schemes come. In the first one, we will assume that the CPU itself is doing the status checking.

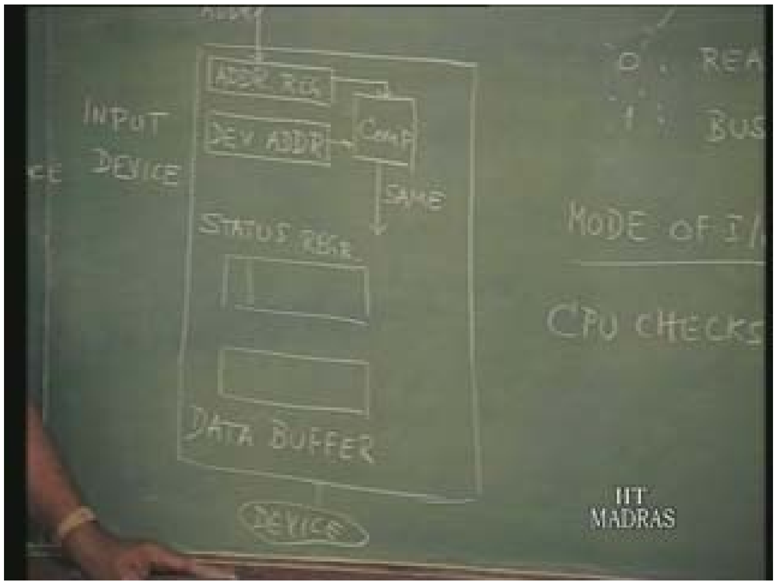
(Refer Slide Time: 32:07)



So it is just one mode of data transfer. I will say one mode of I/O if a CPU itself is doing the status checking. CPU, let us say, checks status; how will it do that? This is one mode. The way to do it is: the moment this device is addressed, this status information must be passed on as the data. The status information must be passed on as data to the CPU and the CPU takes that status data and then sees in our case and then specifically sees this particular bit, whether it is 0 or 1. By interpreting that, it knows that the device is not yet ready with the data, or the device is ready with the data. Now you can see one simple thing here – since we have assumed the most significant bit indicating the status of the device, what it does is it takes this particular information as the data. So this status information will go on the data bus. The status information will put on the data part of the bus; the CPU will take it in and the moment it goes to the CPU, the CPU will see whether it is 0 or 1 and since it is MSB, suppose this is treated as the signed data, 0 means plus and 1 means minus. So all it has to do is it has to see whether the status data is a positive data or a negative data number. If it is positive number, then it knows that the device is ready with some data. If it is negative number, then it knows it is not yet ready. Now what it will do when it is not yet ready is a different thing altogether; we will not worry about that.

So specifically when we say the CPU addresses, we know that the CPU must specifically address the status register and then take the contents of it. In this particular mode, the CPU checks the status – just assume this is one mode. Now assuming the status board indicates that it is ready with the data, well, the data must be somewhere as part of the device controller. So we need one more register, which we call a data buffer or a data register. So we have one register. When we say register do not confuse; it is part of CPU, it is a part of the device interface. This is some interface part of the interface circuitry status register, so this indicates to the CPU. We have seen in general that it is the status because there are many bits in the status register; we can use them for different purposes. This is the fundamental, basic requirement. So now, you can see assuming the device has prepared the data and then put in the data buffer, we have assumed an input device.

(Refer Slide Time: 38:04)



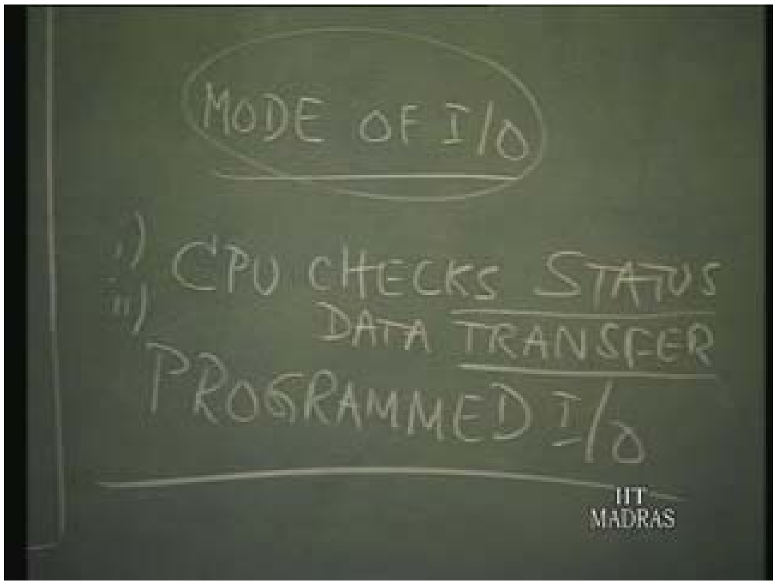
So at some point in time, the device has been ready with the data it has loaded here, and it would have set this as 0, meaning that it is ready. I will just arbitrarily assume for instance this bit to be 1 for ready and 0 for busy also. So when the status register indicates the device is ready, then it is an indication to the CPU that the data buffer here is ready with the data, and from the buffer, contents of the data buffer will now be placed. They will have to be placed on the bus and then over the bus, the CPU will take the data. So that is an input or cycle. For the input device, now we have seen the need for a status register and a need for a data buffer. In fact, specifically this is also like another register, data register. We call it buffer mainly because the speed at which the device readies the data is going to be different from the speed of the CPU.

So it acts like a buffer in between and I had also said earlier you do not know about the characteristic of the device. May be the device generates only 1 bit at a time, whereas on this we have a 16-bit data bus at this end. So, one by one, the 16 bits will be assembled and that assembled data now in this case will be available in the data buffer. When all the 16 bits are ready the status bit will be indicated to show that the device is ready with the data. Then the CPU, while reading this, knows that it is ready and that we had already seen. When it knows that it is ready, the data buffer contents will have to be placed. So now the CPU does it in two steps – CPU addresses the status register, takes the contents analyses and then sees whether it is ready or not. If it is ready, the CPU addresses the data buffer and then read the data contents. It would have got 1; we may call 1 unit of data; whatever it is 16-bit data or may be 8-bit data. So essentially, this particular one for this has to happen.

The CPU addresses, gets the status, let us say reads the status, CPU addresses data buffer and gets the data. In other words, a program is always running at the CPU end; all that the CPU has been doing is whatever it was doing with memory, except that it is the interface part of the I/O that CPU is dealing with.

So now you can imagine like this. For instance, the status register can be assigned one address; data buffer can be assigned another address; I said it should have some id, some identification, some unique code so that the CPU can place the code corresponding to the status register; read that; the CPU next can place the address or the CPU can place the address of the unique code corresponding to the data buffer so that it get that. How is it different from reading from two different memory locations? It is similar. So the CPU–I/O interaction is not different from the CPU–memory interaction, except that the memory is usually fast. It is always ready and whenever the memory is not ready we had seen that a wait state will have to be introduced. Here similar to wait state we indicate that the device is not ready. So the CPU reads from the status register, and then, if the status indicates that it is ready, then the CPU reads from the data buffer. So in two read cycles or in specifically the read cycle there is nothing but input. So in two input cycles, the CPU gets the required data, which means a program is running and CPU is controlling the whole situation. This mode of input/output is called a programmed I/O mode, in which first CPU checks the status and next it does the data transfer. Now this transfer which we were talking about is for the input; this is the next step.

(Refer Slide Time 44:22)



MODE OF I/O

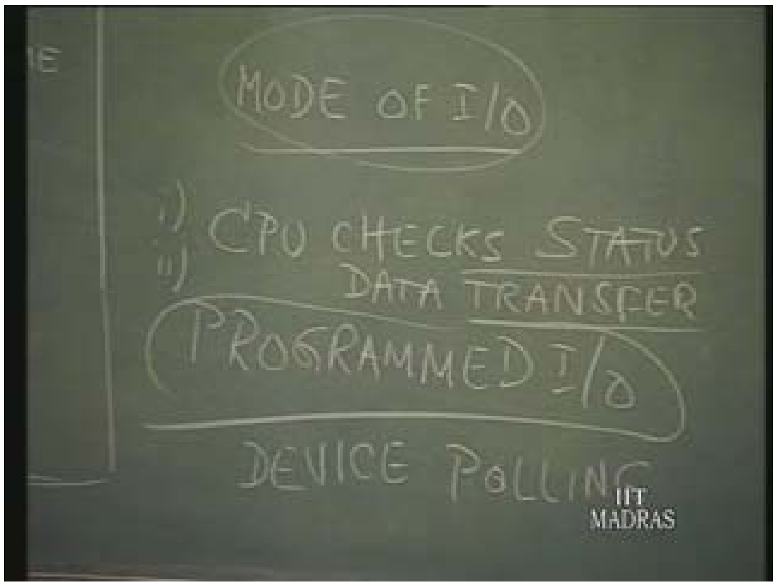
i) CPU CHECKS STATUS

ii) DATA TRANSFER

PROGRAMMED I/O

That is what it has been doing in programmed I/O mode. This is one of the different modes. So you can see that the whole thing is taking place at the initiative of the CPU. Earlier we said there may be many devices; in fact let us say if there are 10 devices, what should the CPU do? CPU possibly addresses the first device; let us say only the third device is ready. First, it addresses the first device and sees it is not ready. Then, the CPU addresses the second device from its register, device status register number 2, it sees now it is not ready and then it addresses the third status register. We assume that the third device is ready so it sees that the third device indicates that it is ready. Then, immediately the data from the third register will be read. Now let us say the device addresses fourth, fifth and so on and so forth. That means basically in this programmed I/O, the CPU takes the initiative and then we can say that the CPU keeps polling among the various devices. This arrangement, programmed I/O, is also called device polling mainly because the CPU goes about it.

(Refer Slide Time: 46:34)


MODE OF I/O
i) CPU CHECKS STATUS
ii) DATA TRANSFER
PROGRAMMED I/O
DEVICE POLLING
IIT MADRAS

The CPU polls among the different devices and then, whichever device is ready; it gets the information by looking into that status register. So whichever is ready, it takes the data from that particular device. Just as I said out of the 10 devices, some may be input some of them may be output. In case it is an output device, the situation is not different. What it will do is, suppose these were an output device, then the CPU sees whether the device is ready; it is an output device. Then when the status indicates that the device is ready, the CPU sends a data to the data buffer and after the data buffer gets the data from the CPU, that will be passed on to the output device. Now while the data is going to the device, the status will indicate that the device is busy. When the data had been completely passed on to the output device at the end of it, the status will indicate that the device is free for the next unit of data. So the CPU will be outputting the data and then passing it on in the case of output device; otherwise the situation is similar – the same status register and the same data buffer. This is only one mode in which the CPU takes the initiative. You must remember that the CPU takes the initiative; how? The CPU takes initiative of that which is programmed and then holds among the different devices. CPU takes the initiative in checking device after device – how does it do it? It is as per a program. It checks in any order; it all depends on the particular program; that is why it is called a programmed I/O. Today you can have the program the way you want; tomorrow you can change that. I said the device 3 is ready device 1 and 2 were checked and found not ready.

Now after it services the device, that is, after it takes the data from the device, because we assume the input device, after it device 3, it goes to device 4. This you can change in the program and then, after the data has come from device 3, it can go back and check whether device 1 is ready. You can do it in any order; that is why we say it is programmed. So the sequence in which the device can be polled entirely depends on the program and that is left to the user. Now what is the problem in this? The problem in this particular thing is the CPU is all the time tied up with addressing one device or another, because most of the time it is just carrying out a device: input/output.

Meanwhile, it is not doing anything with memory; of course let us not worry about that particular aspect. If not, some processor will be doing it. So it is not necessary that though it is convenient for us to assume that the CPU is doing it, just change the word a little – instead of CPU just say a processor is doing it. We will talk about that particular aspect later. So this is only one mode, I/O mode, in which the CPU takes the initiative, checks whether the device is ready, if it is ready it effects the data transfer. You say effects the data transfer; meaning it can be input or output depending on what the particular device is. Now there are other modes, which we have to talk about and which we will do in the next lecture.