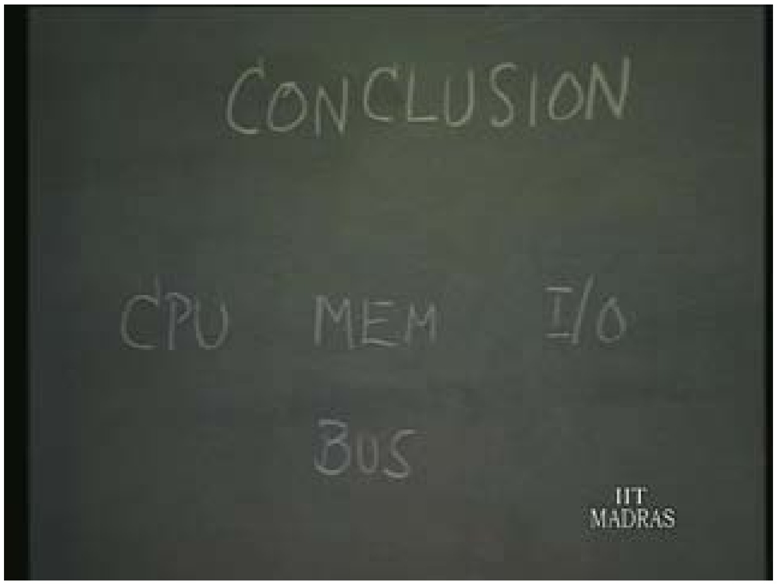
Computer Organization
Prof. S. Raman
Department of Computer Science & Engineering
Indian Institute of Technology, Madras
Part – IV
Buses
Lecture – 33
Conclusion

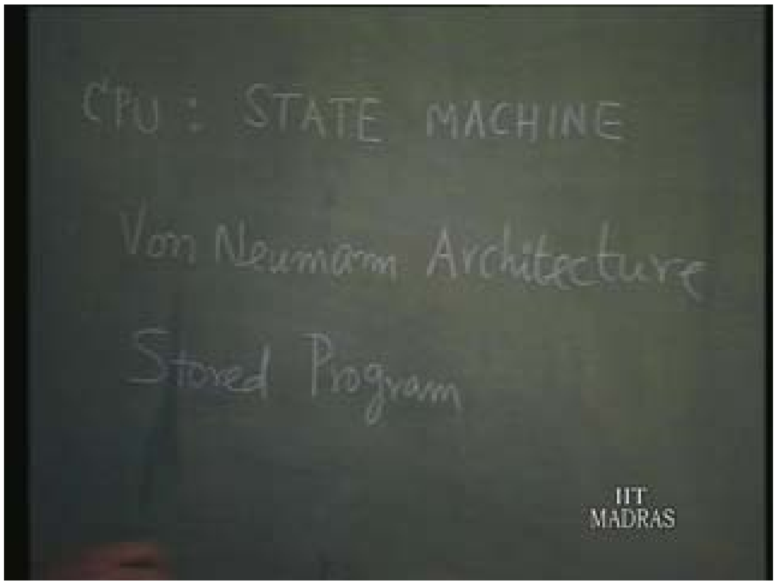
In this series of lectures we had seen the settle difference between computer organization and architecture. That is how we started with. The organization as we said is from the user point of view and the architecture is from the designer point of view. After grasping this fundamental difference, we went in to the details of the components of the system. That is essentially we want CPU, Memory and I/O to identify separately.

(Refer Slide time: 02:03)


CONCLUSION
CPU MEM I/O
BUS
IIT MADRAS

Essentially these 3 are the units and then to interconnect them we have the bus. First we went in to the details of the processor or the CPU and then the memory and also the interaction between CPU and memory. Later on we took a look at the I/O and then how I/O interacts or gets governed by the CPU or interacts with the memory. In the whole process the bus is involved. Towards the end we also took a look at the different types of buses not in detail, but just a bit of introduction. It’s time to conclude this series with a few general observations on, what has gone in the development of whatever we have seen so far and also possibly to provide you some thought about what can come out of the whole thing. That is what possible feature extensions are there. In fact towards the end of the first part itself I was mentioning that is, we were taking a look at the processor as a state machine. Processor as a state machine means, it always goes through specific states and execute some simple steps in each state. That is what the state machine is about and in this we noticed that we have both instructions and data to be handled.

(Refer Slide time: 04:14)

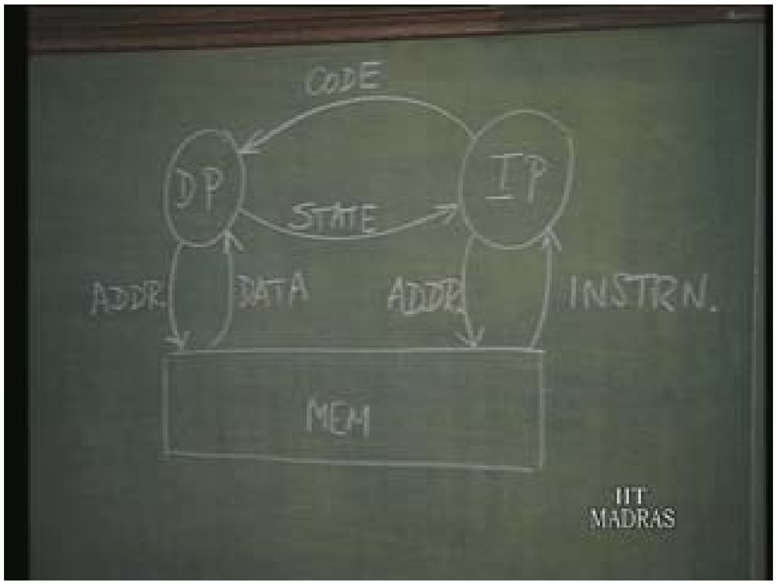

CPU : STATE MACHINE

Von Neumann Architecture

Stored Program

Because as I said in the Von Neumann Architecture you have the processor storing both the program and the data. In other words what you have is the stored program digital computer that is what you have as per the Von Neumann Architecture. The program consists of instructions and data, both are stored in the processor. Processing involves fetching an instruction and executing the instruction. As we said essentially the processing involves for instance you can call one part as an instruction processor and another part of the CPU as the data processor. Both instructions and data are stored in memory.

(Refer Slide time: 06:33)



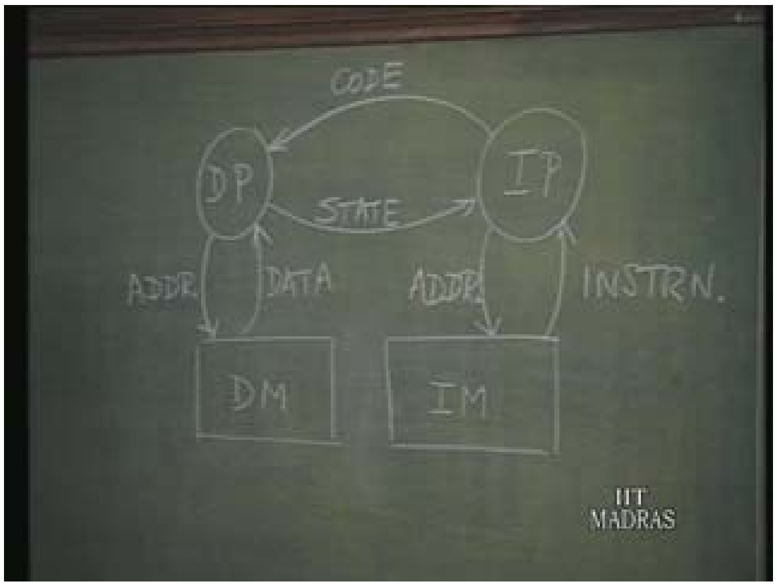
First the instruction processor addresses the memory and gets the instruction which is in the form of some code.

We said instruction processor sense out address to the memory and get the coded instruction that is it will be in some sort of code format. Then this instruction is interrupted or decoded and the decoded part actually goes to data processor. So we may call it as instruction or just some code. Instruction is interpreted and in case the instruction refers to data, then like instruction processor, the data processor also sense out the address and fetches the data from the memory. The program consists of instruction and data and these are stored in the memory and the processor fetches and executes. After one instruction is fetched and executed, the data processor will give some information feed back to the instruction processor which generally we call as processor status or just some state code will be passed on. For instance it will be obvious, suppose we have a branch on minus instruction. Suppose there is an instruction, so it says branch in case there is effect of that particular whatever it fetches in, happens to be a minus number.

It will fetch that instruction first, and then execution of branch minus will take place, like it will keep looking for the data and then it will indicate that it is negative or positive data. This information will go on. Let us say in the case of addition, there can be an arithmetic overflow there can be a carry result of it or there may be some other fault. At the end of fetching the instruction and executing the instruction that state information that it resulted in a carry or an overflow or the addition resulted in a negative number or it is a zero number or non-zero number. These are all the state information that the data processor passed on to the instruction processor because some time instruction processor will look for the state and then it will decide what is to be done next. Should it go and fetch the next instruction or it should do something else. Decision as such will be taking place.

This particular figure exactly goes on in a Von Neumann Architecture based computing machine. In carrying it out the machine goes through series of states and as I early said that state is different from this state. This if you want you can call it as the status of the process that is status of the instruction that has been currently executed. Some times we also call it process state. As I said again and again that I/O can always be considered as extended memory. So I/O can be looked as extended memory and we would not have to specially consider from the functional point of view. Though in the actual use I/O has its own specific role and memory has slightly different role. The picture given in the below slide is the typical architecture of the machine. The first modification came about, for instance when this memory was split and one had one part as instruction memory and another part as data memory. What is the advantage of this?

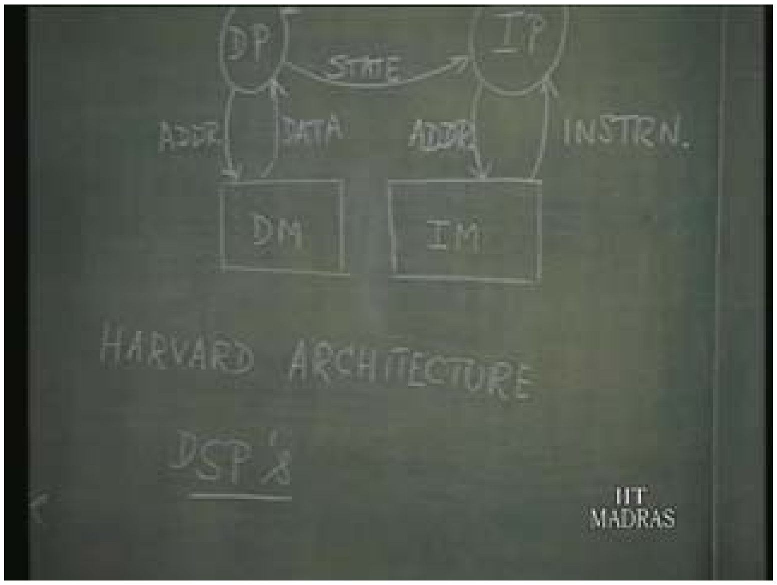
(Refer Slide time: 09:37)



The advantage of this particular one is generally we find that the bus which connects the processor and memory can now be split in to 2. We can call this part as an instruction bus and that part as data bus. This gives some possibility of parallel execution. That is while an instruction is being fetched, some other instruction can be executed. Because the other instruction may involve some data fetch that can go on in parallel. Some amount of parallelism get introduced this way. In fact this very first thing that is the first change to the original one had let to what is called a Harvard Architecture.

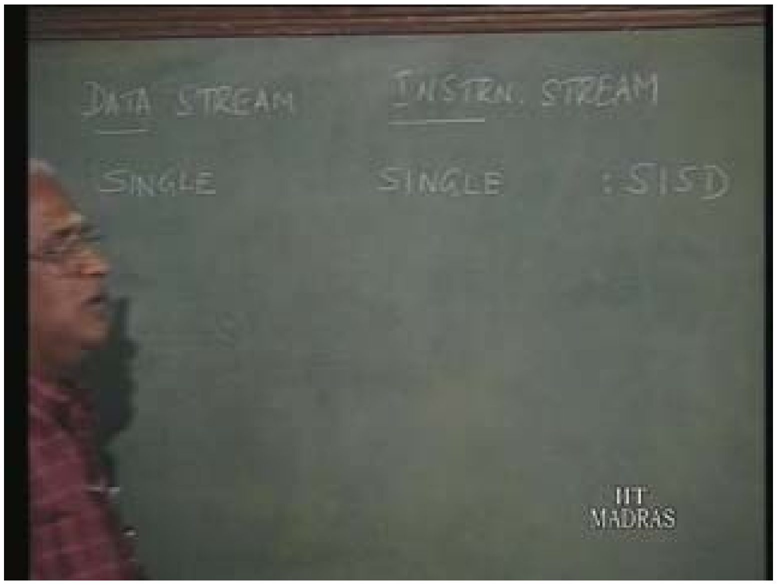
When you have like this then its no more called as Von Neumann Architecture, though as you can see that is the very small step, you have to separate memories and the associated bus. That leads to Harvard Architecture and this Harvard Architecture is very much useful in special purpose processors that are the digital signal processor (DSP) which is quite useful. Because in DSP there is huge amount of computation involved. When you have huge amount of computation we have got to see that every nano second or every small time unit that is possible can be fully utilized by the processor. This way by splitting, instruction processor and data processor can work in parallel and more can be achieved. It is true to some extend. The moment this 2 processor have to communicate, that particular parallelism comes to a stop. Is it not? If one has to wait for the other one then both cannot go in parallel. This is the first step.

(Refer Slide time: 10:55)



What you think is possibly be a next step. May be before I proceed I will just review something which I mentioned earlier. So this way you can see that they can have 2 streams, call one as data stream and another one as instruction stream. That is we have course which either belong to instruction stream or to data stream.

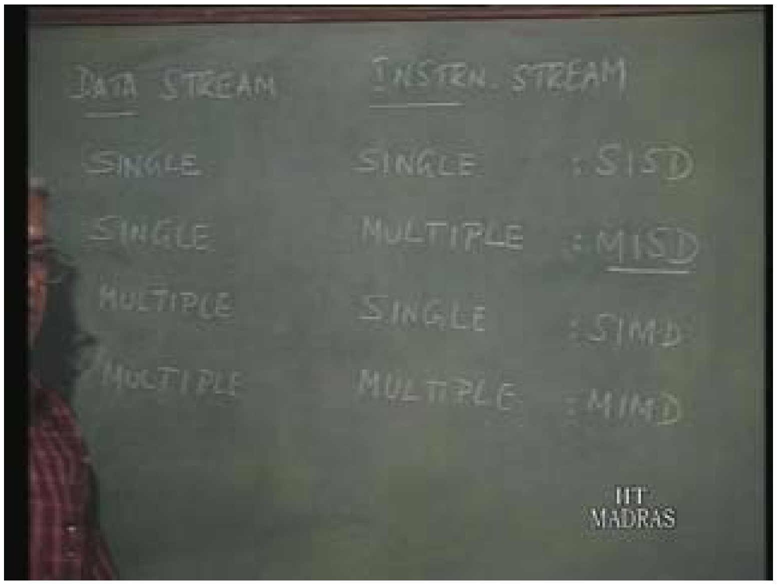
(Refer Slide time: 12:58)



Suppose the architecture is such that you can take only one instruction that is single instruction and then can deal only with one data that is single data.

These which we have seen earlier leads to what is known as SISD class of architecture. In fact this particular classification is due to one computer scientist known as FLYNN. So whatever I am going to talk about is known as Flynn's classification. There is only one processor and so it can take only one instruction and there is only one data processor and so the system can deal with only one data. Single instruction and single data which we generally have. You may have split memory or you may have a single monolithic memory but it just depends on the number of processes. On the other hand suppose you have many instruction processors. That is multiple instructions it can carry out and there is single data. Then you have a class of multiple instruction single data. Suppose you have only a single instruction processor but multiple data, then you have (SIMD) single instruction multiple data stream.

(Refer Slide time: 15:00)



DATA STREAM	INSTRN. STREAM	
SINGLE	SINGLE	: SISD
SINGLE	MULTIPLE	: MISD
MULTIPLE	SINGLE	: SIMD
MULTIPLE	MULTIPLE	: MIMD

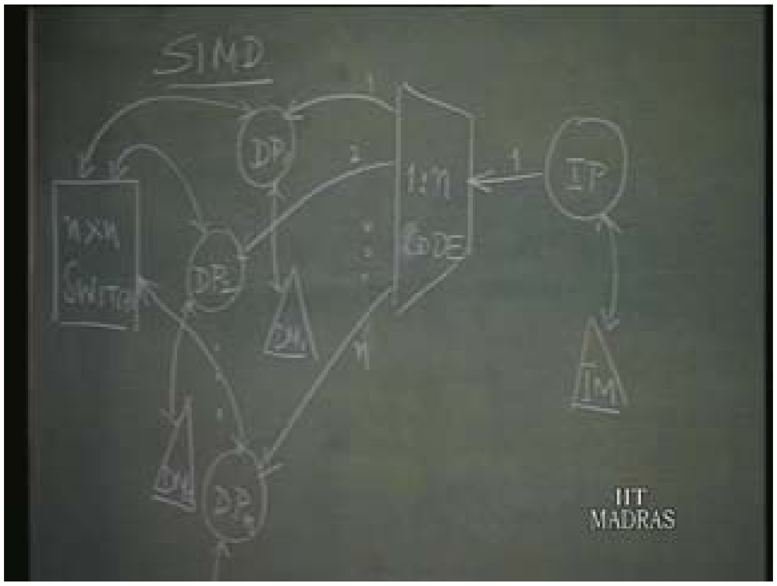
And the last is the multiple instructions and multiple data; this goes to the class of MIMD or multiple instructions multiple data stream. The SISD is the only one in which both are single that is the normal one or conventional one. Then in the other three we have something or another, either instruction or data multiple. Of these this SIMD and MIMD are just called sometimes as SIM D and MIM D. This particular one can just put it as parallel computing system. The parallel system of SIMD or MIMD type. Why this MISD is not included in SIMD and MIMD? You have a single instruction and then the multiple data let us say a same instruction is worked on many data. What will be the end result? Again you will be having a set of multiple data and the next instruction again can be single. There is no problem. But look at this MISD, suppose you have a multiple instruction and then single data. There are many instructions working on single data and so the result of that will be multiple data.

So only for one step you can talk about multiple instruction single data but in the next step itself the data will become multiple. So for only one step you can have the MISD and the next itself it is going to become MIMD. That is why we do not really see that this (MULTIPLE) is infact a practical system SD. We never consider this (MISD) and this is generally not a practical system. This (SISD) is okay, it is a normal conventional sequential system and SIMD, MIMD is the parallel architectural system. That is the first thing that you can notice from this simple diagram.

So let us try to develop further based on this as the core. What we had seen is actually single instruction single data because we have got 2 single processors. You can have single instruction multiple data or multiple instruction multiple data as I said earlier. That is obviously the next step we will take a look at that. Let us see how single instruction multiple data looks like. We have one processor that is the single instruction and multiple data processors that is we have data processor say $DP_{1,} DP_2 ... DP_n$. You have only one instruction. When you have multiple data, one possibility is the same instruction is going to be executed by all the data processors. This is one thing, other wise the same instruction can be split in to let us say n stages of processing and the n stages of processing will go on in parallel on these n multiple data processors. This is another way of looking at it but that particular single instruction must be executable in n individual steps or mutually exclusive steps, then only it is possible.

We have one instruction and then we split that in to n. There is only one instruction which goes and I am not showing the memory, just the processor alone we are taking. So one instruction goes and let us say an n copy of that instruction comes to DP. That is the first one whatever I was mentioning, not the single instruction which is split in to n stages. It is the same instruction which goes in to this DP. This must be 1: n switch. In fact we call it even as coder or some sort of code arrangement must be there. That is from a single instruction code, n codes have to be generated or n sub parts of the same code must be generated.

(Refer Slide time: 22:54)



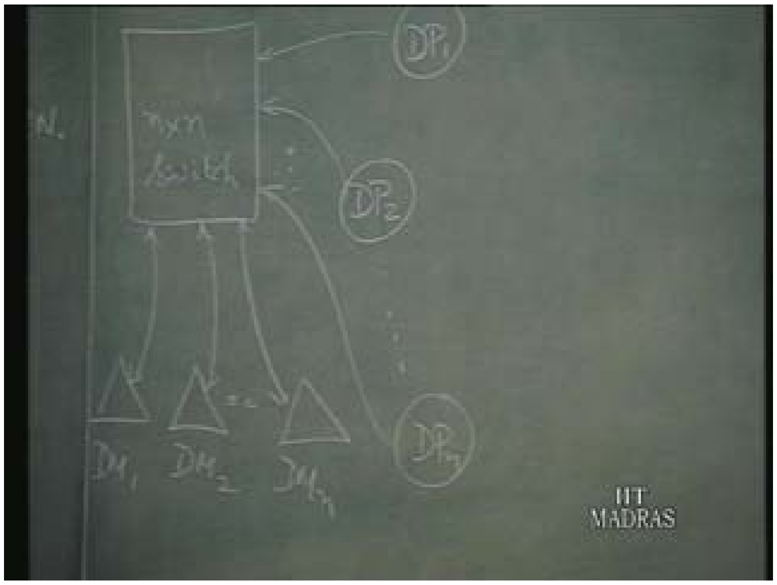
What about the memory part of this? There is only one instruction memory that is not a problem because there is only one instruction, one processor. What about these DP? So there are 2 ways. One is each of these data processors may have their own memories that is data memory. You have this data memory for first processor, the second processor and n^{th} processor. It means each processor has its own local memory. But what can be done if all these things are going to do it independently? At some point of time these processors must also communicate. You have got n of these, so you must have some kind of a $n \times n$ switch or something like that.

All these processors can be connected to the switch, so that there can be processor to processor communication. This processor DP_1 after it finishes on this (DM_1) data, can communicate to the processor DP_2. In fact it can be bidirectional. Basically this switch is required for communication among the multiple processors. What happens when we talk about multiple processors? We are entering in to the new dimension of the problem that is communication is a part of computation. We said for computing, efficient computing and essentially for fast computing we will go for multiple units, whether only multiple data or both multiple instruction multiple data. You may have multiple data because these things can come from multiple data memories but then at some point in time there must be communication among these processors. For the first time we are seeing that communication among the processor is needed otherwise meaningful processing can not go on. It is some what like this, a single command is given and multiple processors obey the command. It is very much like; see the leader of a battalion giving a command and all the people in that battalion obeying that which is some what like that. You can say that an array of soldiers marching along and obeying a single command.

This is one of the nice architecture for array processing. So essentially it is an array processing architecture. Wherever it is possible to carry out on multiple data and if communication between these or among these processor is not a problem then this array processing will be very efficient. The problem comes that moment the communication becomes the bottle neck because for computation we have taken care of, but what should be done for communication. If it can be predicted what is the amount of communication between these processors, it may only be the prediction it may not be strictly true. Sometime it may fail and sometimes it may be okay. So communication among processor is one major issue in multiple processing specifically array processing. Slightly let us alter this. Why the data memory has the local memories of the respective processors. Why not try a slightly different architecture? If you have like this what happens? We can just see this processor DP_1 and memory DM_1 together is like one system and these (DP_2, DM_2) is another system and this (DP_n, Dm_n) is another system. Though we say $n \times n$ switch, we can just see that these systems are something like loose systems and they are coupled through this switch.

Actually it is loosely coupled array processor architecture. Why it is loosely coupled? Mainly because each processor can carry out its own processing with its own memory and then they interact with other only for communication purpose so it is loose. Instead of having liked this, suppose you have these various data processors connected some what differently with their memories. This (DP_1, $DP_2 \ldots DP_n$) are connected through the $n \times n$ switch as before but then have these data memories ($DM_{1,} DM_2 ... DM_n$). And these data memories are connected to the switch.

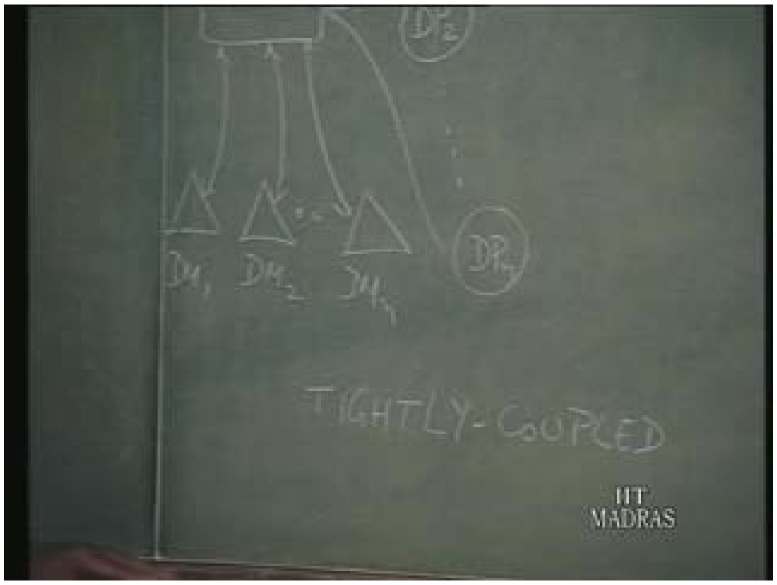
(Refer Slide time: 27:58)



The rest of it is as before. The processors are connected through the switch to the memory and unlike the previous one we can not say that this memory DM_1 is associated only with this processor DP_1. This memory DM_1 is not actually associated with this processor DP_1, any processor can access any memory. The coupling becomes different. It can be like this, for one instruction this processor may use this memory DM_1. For the next instruction the second processor may use the same memory.

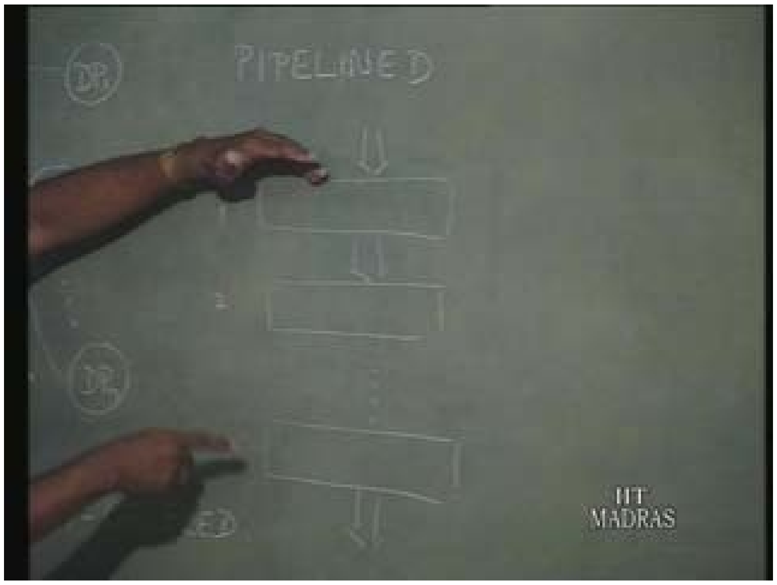
In other words we can consider this memory as shared between 2 processors. That possibility is there. What is the advantage of sharing? Advantage of sharing is the processor themselves do not have to communicate; it is enough to communicate through these memories. This is in fact shared memory architecture but then what happens is, in these particular one the system is loose and then they communicate. In other thing the coupling is rather tight, so this is a tightly coupled architecture. Otherwise this is also one form of array processor architecture. This is the kind of thing what you have in the SIMD, that is single instruction multiple data.

(Refer Slide time: 29:40)



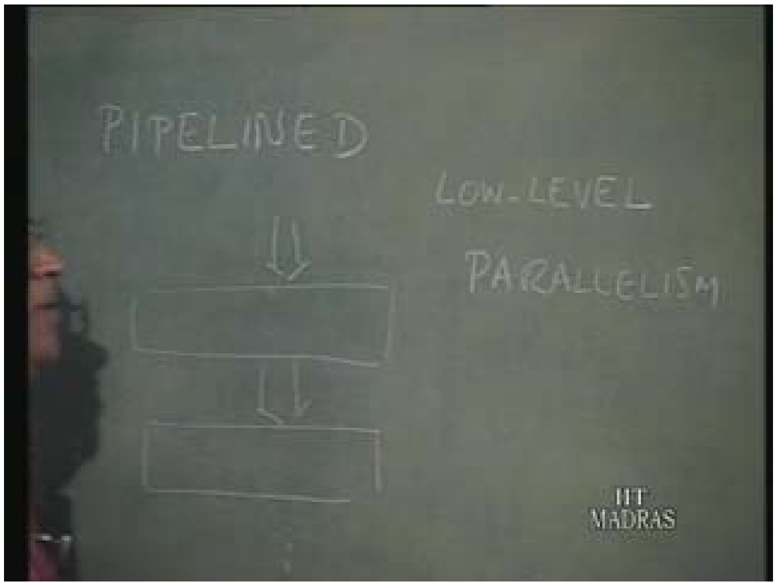
I said earlier these n may be just n copies of the same instruction and then there is another possibility that the same instruction split in to n stages and you have n stages of processing. The same instruction is executed through n stages. If that is possible which means the n stages of processing will go on in parallel. Then we talk about pipelining architecture. That is you think like as if one instruction is flowing and this particular instruction can flow through n stages of processing and then finally the output comes. The one part of the first instruction is getting executed here and one part of the n^{th} instruction will be getting executed there. (Refer Slide Time: 31:08) That is it is going on in parallel. Because of this kind of a structure it is called pipelined architecture.

(Refer Slide time: 30:48)



In fact all these n stages will correspond to one particular instruction. It may not always be possible to split them into n a stage, which is where the problem comes. In fact the pipeline architecture normally referred to as some low level of parallelism. It is parallel but it is at lower level because you split that.

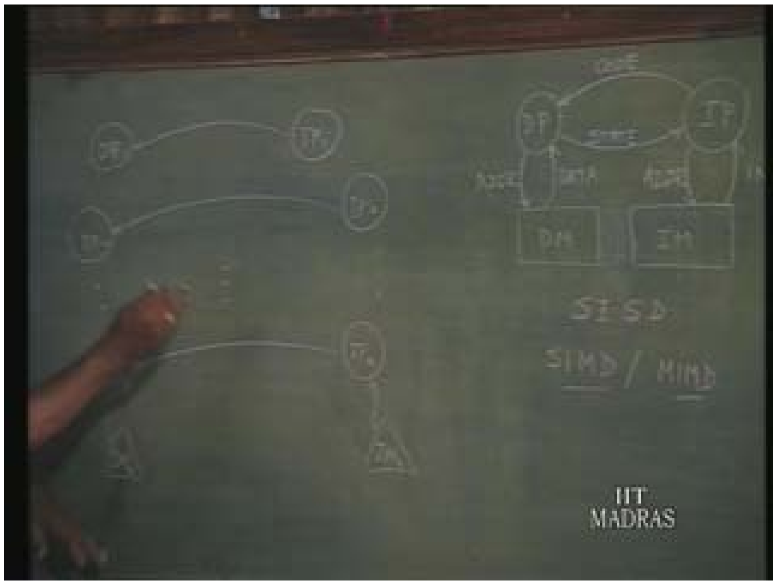
(Refer Slide time 31:53)



For instance, if an instruction can consist of 10 micro instructions. Then if all the 10 micro instructions can be carried out in parallel, then that is what you had in the above slide. But you must be able to execute them in parallel, which is the main problem. It may not always be. If it is possible then we say from instruction we have come down to micro instruction that is why it is called from high level you come to low level. It is a low level form of parallelism. That is in short about SIMD. We will take a look at MIMD which is a slight extension of what we have seen so for. Talking about the MIMD or multiple instructions multiple data, we have the multiple instruction processors and also we have the multiple data processors. Of course you also have the appropriate memories, you may have the monolithic or split and then you can have loose coupling or tight coupling. One simple thing is each instruction processor connected to these data processors. Just see how it really works out.

Is there any difference from the diagram which is given in the right? Just I will do the memory for one processor. We have the data memory and this is what we have in the other diagram also. And we are going to have multiple copies of that single system. That is how it comes for this particular connection. You have multiple instructions and also multiple data.

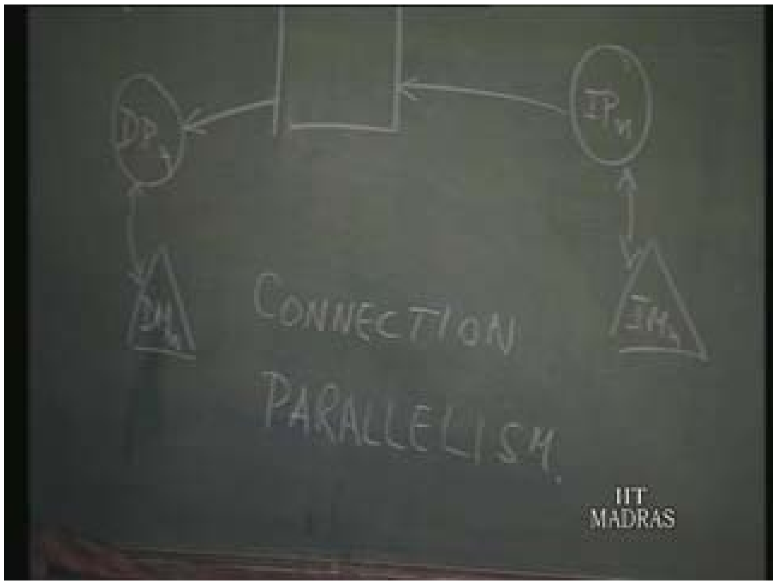
(Refer Slide time: 33:46)



In how many applications this would be possible? This is the main thing. Certain applications will be implemented making use of this architecture. For some application it may not really. For instance you can simultaneously think of multiple things, may be its possible for what we referred to as ashtavadhani, some one who keeps doing different things simultaneously. Normally you always think of one or may be two or three, certainly not more than that. It is also not possible for every body and here it is the same situation. (Refer Slide Time: 35:09) It all depends on the application. Incase the application supports then you can have these MIMD architecture also. As I said these memories may be connected as loose coupling or tight coupling. Instead of just having like this which as I said nothing but n copies of the same thing which is straight forward implementation. Instead of that suppose you introduce a switch in between so that these DP and IP are connected like this and we have $n \times n$ switch which means any instruction processor may be connected to any data processor. Then the strength of this particular system depends on this connection itself.

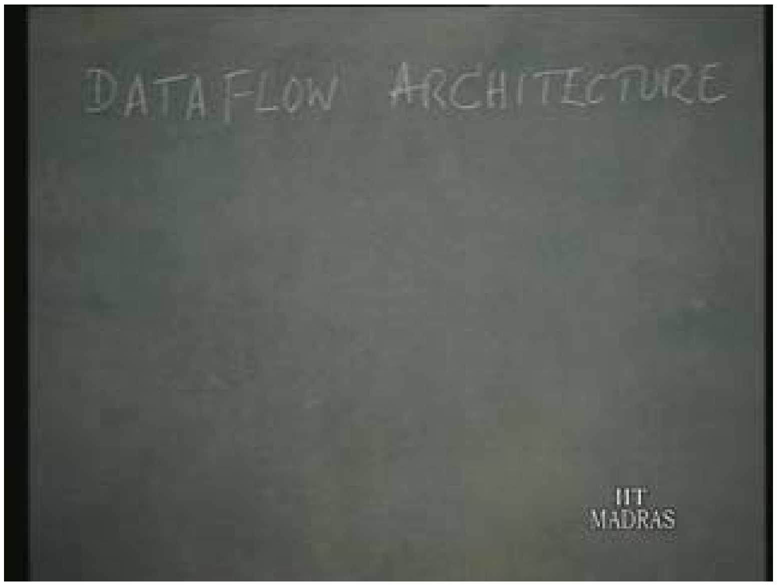
More than the processing capability, the way you can connect any data processor to any instruction processor, which would in fact be an additional factor. In fact systems which make use of this are generally referred to as connection machine and type is a connection parallelism type. It depends on that, sometime we may be able to program we may have a fixed connection and even during the processing you can keep changing that connection. In some applications it will be useful. In short this is about the MIMD architecture.

(Refer Slide time: 36:55)



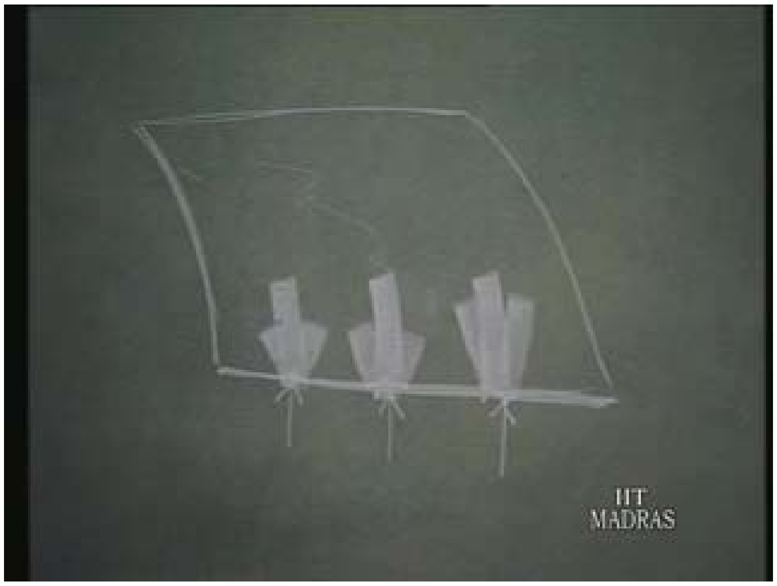
You can see that the all these thing had evolved from very simple concept. What is it? Computer as a state machine, a computer goes through state by state executing one instruction at a time. It fetches the instruction from the instruction memory, executes if necessary it fetches the data from the memory. On that particular data it executes and then it passes on some information so that the next instruction execution may be decided based on the previous instruction. And splitting the memory and then splitting the processor itself to start with and then going in for single instruction or multiple instruction, single data or multiple data. This is how it keeps going. In addition to these you also have other types of architecture. For instance very special type of architecture called data flow architecture.

(Refer Slide time: 38:26)


DATA FLOW ARCHITECTURE
IIT MADRAS

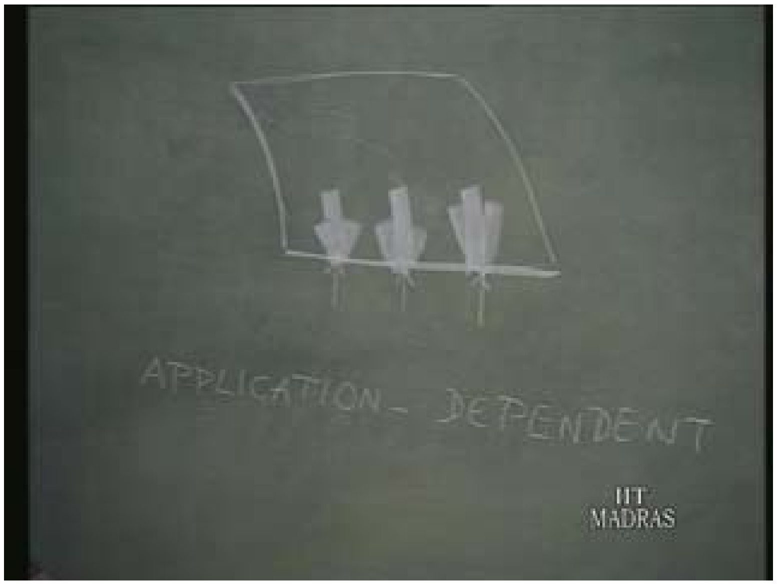
Basically we are taking a look at how different types of architectures have come about and for all these things the Von Neumann Architecture that is the stored programmed concept is the starting point. They all have logically developed by making one in to n and then by changing the connection. You direct connections; you provide connections through loose coupling, tight coupling and so on. The data flow architecture is another one which is a special type of architecture. What is it? For instance say a plane; this may be just a sheet of metal which is bent like this. Suppose you heat along the edges or at specific points. How exactly the heats keeps flowing across these surface may be in any direction? How exactly this heat flows over this? Then it is a very practical problem.

(Refer Slide time: 39:44)



The same way instead of heat you just think of some data that is given at some point. How this data keeps flowing? Here the temperature or the amount of heat that is being given is like data. How exactly this data flows? For instance as I said you heat these thing and then how exactly the heat flow over these surface. Many times you will not be able to give a perfect mathematical model of this surface. If you have the perfect mathematical model it may be possible to compute. How exactly this data flows over this? For this kind of special applications you can think of architecture. So that will also be a special type of architecture again it is parallel but then it is too special. As we go on from this SISD to SIMD, MIMD architecture, you can go on thinking of these special things. But then as I said they will all be application dependent. Necessarily everything will have to be like that mainly because after all it is the user who wants a system. From the user point of view what you have is essentially some application or the other. He is the one who is going to pay.

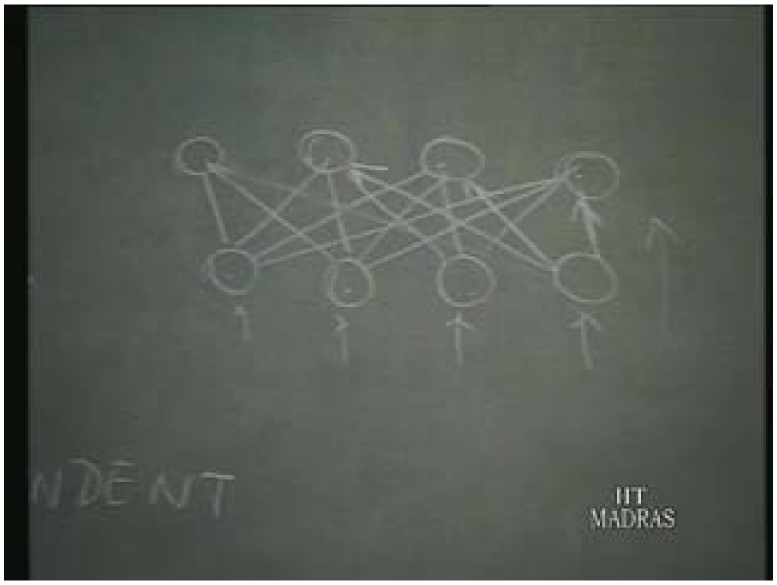
(Refer Slide time: 41:21)



For that particular application we just have to develop the system. It is not that we have a system with the specific architecture and then he will have to use it. If he is willing to pay money, fine come up with another architecture which is meaningful. For different types of applications, different types of architecture will keep coming but then the fundamental thing is a set of instructions working on a set of data. In fact I would say it is not all that easy to imagine what actually happens. Because it may be that we should have a network of computers.

And the whole things in the above diagram have to be simulated with a network of systems in which all of them are interconnected. Then we will have to keep studying the communication among these. We make an assumption that the data comes here in and each one is an independent system. It is just based on this connection. It is possible that the amount of heat that is given here is sort of locally going to spread and not really reach this point in which case there is no need for the connection which is very far. (Refer Slide Time: 43:41) We can always constraint the problem saying these particular process data only flows to maximum of one to the near neighbor and then two to far neighbors and not more than that.

(Refer Slide time: 44:01)



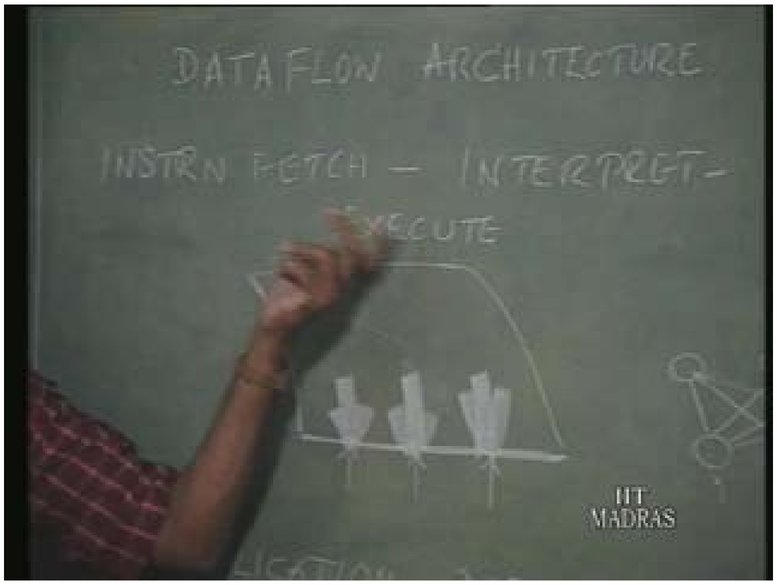
What is the assumption we have made? We have made that the data flows only in the upward direction. It may not always be correct; it all depends on where exactly we are doing. One simple thing I am just showing, you can see that these data flow architecture is very complicated one but then it is a very specialized thing. Quite often practical constraints will have to be brought in. But then what goes on in one point is just again what we are quite familiar with. And then the connection between these is something like networking of the systems and you restrict the communication. Here you can see that these computation at one point and communication with the neighboring points.

So the computation and communication, these are the things which we may call them as high end systems. If you want call them as low end system, the high end systems very much dependent on these. So that in short is about these courses on computer organization. We started saying how exactly a computer can be looked at as consisting of cpu, memory and I/O. Then I said I/O itself is an extended memory and essentially its CPU and memory. The CPU started with Von Neumann Architecture in which the concept is stored program which means the memory holds both instruction and data. So we think of instruction processor and data processor and a single memory. Then in Harvard architecture that also be split in the instruction memory or data memory and this is the basic thing about the architecture that is for the single instruction and single data.

We also saw that duplicating this processor and duplicating these memories and providing the different types of interconnection, we can go on for SIMD and MIMD and so on. So before that in the different section we took a look at how exactly a processor is design what goes on in the processor and then also the hierarchies of memories talking about. Then the device then the interconnection between memory and device, memory and CPU, of course CPU and device if there is a direct communication and so on.

Also we took a look at the bus, a bit of information on different types of buses essentially it is CPU memory I/O bus and then going in for these special arrangements. For all these things the starting point as I had already said is the simple instruction cycle that is fetch an instruction that is instruction fetch then interpret and the last is execution.

(Refer Slide time: 47:32)



Of course instruction fetch we say but then after interpretation, execute will also involve the data fetch. And that is what this organization course is about. Instruction cycle remember that everything will work out from the details about that.
Thank you all.