Computer Organization
Part – II
Memory

Prof. S. Raman
Department of Computer Science & Engineering
Indian Institute of Technology, Madras

Lecture – 15
Introduction to memory system

In this second part of the course, we will take a look at the memory part. For us to gain and understanding of how the memory should be functioning, we better go back a little and then see what CPU was doing with memory. Essentially, the CPU will be dealing with the rest of the components in the system, including memory. As memory is one of the components, the lower level or subsystem at slightly higher level, we will consider only the interaction between CPU and memory. Essentially, what goes on between these is part of the bus activity; let us remember that. So what was CPU doing? CPU was generating some address signal, that is, the address part, and then it was indicating the correct sequence in which it was working. CPU addresses the memory and then indicates the set of control signals it wants the memory to do. So maybe it is something like CPU indicates to the memory that it wants a read to be performed, or a write to be performed. These are actually control signals. Now depending on whether it is a read or a write control signal, appropriately the timing of the second part, namely, the data between the CPU and memory, will vary but nevertheless there is some flow between CPU and memory in terms of data also, just like address.

I was talking about the timing; why? It is like this: suppose CPU wants a read to be performed from the memory, essentially CPU tries to read something from the memory. So the data must flow from the memory to the CPU. On the other hand, if CPU wants to write something into the memory, then the CPU should not only place the address, but also the data and then only it has to generate the write signal. That is why I was saying the timing in which the data component part of the signal comes would depend on what exactly it wants. Now it is like this – CPU addresses the memory and, let us say, it gets the data – whether it gets the data or sends the data does not matter. Now you can see the way we have been talking about CPU essentially is that it is the master of the bus; it is the master of the whole situation, and memory responds depending on what is to be done. So we may say that memory is the slave in this situation. So CPU as a master demands memory, and memory responds like a slave; now this is one thing we must bear in mind.

So the CPU places the address and, whatever is to be done, read or write, is performed, and then the data flows between CPU and memory. So functionally now you know that CPU as a master will demand and memory as a slave, will respond. This is number one. We will be talking about the CPU: suppose when the CPU wants the memory to respond and memory is not ready. In other words, if the CPU is faster or find memory is slow, the memory will indicate that it is not as fast as CPU wants it to be, and then there is another signal called ready. Actually it means if the signal wants to indicate that it is slow, basically what it says is it is not ready. You can take that this ready input in the CPU is always one or up, and whenever memory wants, it is pulled down.

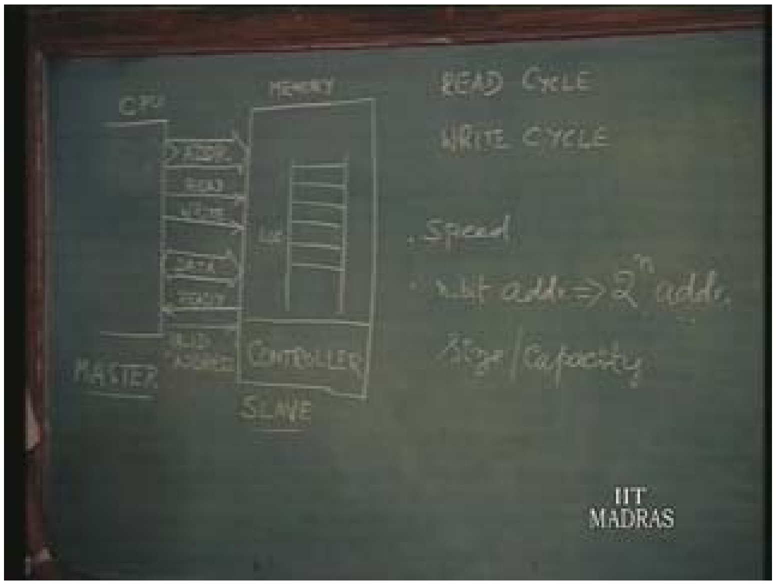
Until the memory indicates that the memory is ready with the data, in case it has generated data, and sends it over is or ready to accept the data in case it is a write cycle, it will first pull this ready up again and then indicate the CPU that now it is ready to proceed. So you can see that you have the address part of the bus; you have the data part of the bus; you have the control part of the bus. Now of the control thing we have already seen three: read, write, and then the ready. Read and write are the two control outputs from the CPU and ready is a control input to the CPU. So now you can see that memory as a subsystem should understand this address input; it should understand the address input so that it may prepare the data from that particular location and then it should understand the control signal so that it knows what it has to do.

In case the memory is slow, it also will have to generate another input to the CPU to indicate whether it is ready or not, and then it sends or accepts the data depending on what the actual cycle is. What is the cycle? We talk about read cycle or a write cycle, depending on what the CPU wants to do. In the read cycle the CPU will place the address; it will generate the read and the memory will respond with the data and the CPU will read that data in – that constitutes a read cycle. In other words, with reference to the CPU, there is some data which is coming in. The CPU now reads with reference to CPU.

In the write cycle, the CPU as a master generates the address; it also prepares the data and puts it and then it indicates write and the memory will write that data into the specific location indicated by the address. So actually, in this particular case, CPU outputs the data. So read is nothing but inputting something to the CPU and write is nothing but outputting something from the CPU, nevertheless, the words input and output will be used not with reference to memory but only with input/output subsystem or IO, input by output units, IO subsystem or IO units. In that process, if the CPU wants to indicate it is ready or not ready it will also make use of this kind of a signal. In addition to this, there can be other sets of signals also. Ready is just one thing I have indicated; later on maybe we will see. I will just indicate right away – it is like this.

Earlier I think we had talked about something like an 8-bit address or a 16-bit address. An 8-bit address or a 16-bit address obviously comes over 8 lines or 16 lines in parallel. Each bit comes over one line, so for instance, if it is a 16-bit address over 16 signal lines, this particular 16-bit information will be coming. Logically this is okay; we would say CPU places the 16-bit address but physically speaking, each of those 16 bits may arrive at the memory end at different times – why? It is because of some delay in each of those things; in other words, each of that particular signal line will behave like a transmission line and you have what is known as a transmission line delay and all those problems.

(Refer Slide Time 19:11 min)



So physically speaking, there are a lot of problems to indicate that the entire 16-bit address is available at the memory end. For instance, there may be another signal – now if it is known that this particular part of the bus will delay, let us say, by some 20 nanoseconds maximum, the CPU may place the 16-bit address, and after 20 nanoseconds, it may indicate that the address is valid on the bus, which means it is guaranteed that when that particular signal comes from the CPU, at the memory end, the entire 16-bit address is available. So there can also be additional signal such as that, for instance, that may be indicated like this: a valid address and so on; we can go on building more and more complex interfaces between CPU and memory.

As an example, I am just giving you – and this is also part of the control bus, the CPU places the address, let us say, after 15 nanoseconds because it knows that the delay is not going to be more than 20; after 15 nanoseconds, the CPU may generate a valid address. When the valid address seen is at the memory end, memory knows that the entire 16 bit is available. So similarly, we can go on building extra things also and the same thing holds good the other way. For instance memory may place the data and then after some delay, it may indicate it to be valid data and so on and so forth. I have just given you a few samples here; all these things will constitute the bus. We will talk more about this later. So essentially, this is the functional aspect of the memory, that is, the memory in essence stores something, though we may use the term storage we know that it is usually used in different contexts. The memory stores some information whether it is an instruction or data, and each of these are called locations. Each of these locations will be uniquely identified by the address.

So functionally, memory stores some information in address locations and in each address location whatever it contains, it will pass over the data to the data interface between the CPU and memory, and then the actual flow, whether it is CPU to memory or memory to CPU, will be decided by what exactly the CPU wants. All these take place at the specific instance of what the master wants. I am going on repeating this point mainly to drive home the set of signals and the sequence in which the signals must come.

Later on, we will define the protocol of the bus – the protocol of the bus will be decided by which is the master and which the slave is and so on and so forth, and the sequence in which it wants. So we will talk more about it when we come to the bus part. Now let us see the things that we have to note to proceed with further discussion on memory. One thing we know is that memory must consist of locations, which will hold a byte of data in each of these locations. If it is a 16-bit data, then we talk about 2 bytes corresponding to one location and so on and so forth. In addition to this, the memory subsystem must also have the necessary part of circuit, which will interpret whether read signal or write signal has come and then, in case other signals of valid address have come and also to indicate whether it has to generate the ready signal or not. In other words, apart from this storage aspect in the memory subsystem, there should also be a part of the memory subsystem consisting of what we may call the controller.

What exactly this controller will do will depend on this CPU–memory interface and the physical characteristics of the memory also; that is, logically we talk about CPU placing an address and memory respond with data and so on and so forth – this is the logical level. At the physical level, I have already indicated one point that not all the 16 bits arrive and so we need some extra signals and also one more thing: physically if CPU is fast and memory is slow for some reason there is a need for ready signal. So actually the physical characteristics of this particular one will also decide what this controller should do. So really speaking, depending on the technology based on which we do the storage in the memory, the controller also will have to respond appropriately. That is as far as the logical part and the physical part is concerned. Now let us see the CPU as a master; I have just indicated this. Now this brings us to this: why do not we just put down this the speed and let us say a 16-bit address. When there is a 16-bit address between CPU and memory, precisely I imply that the maximum memory capacity would be 2^{16}; that is in other words, if you have an n bit address, it implies that I can have as many as 2^n memory locations. I just put it as 2^n addresses, and assuming for each memory location there is a unique address, we have 2^n memory locations; we just assume that.

In other words, what we mean by this is the size or capacity of the memory – by size of the memory I don't mean the physical size, really speaking the storage capacity of the memory; that is, how many locations can be there and what exactly is the size of the location? That is another thing – it maybe an 8-bit location or it may be a 2-byte location and so on. We will assume just a byte location, in which case we can have as many as 2^n bites. If you assume this particular width of the data is 1 byte, since you can have as many as 2^n locations, the capacity of this memory becomes 2^n bytes. This capacity and speed capacity – these are the two things and the third one, related to these, will be the cost. It is obvious that if the size of the memory is high the cost will obviously go up. Similarly, if the speed of the memory is high, the cost will go up. So what is the cost? The speed, size and cost are the three factors, which we have to consider while discussing or designing memory subsystem. There are a few other things; may be I will indicate the bit later.

Let us just move on to see the types of memory that we have; in other words, the bases on which we can have some classification of the memory. There are many dimensions actually; the classification I am putting across is a very general way; as I said there are many dimensions. I am not really going to go in a very systematic way about this; there are a few things I would speak from the practical point of view.

The first thing that comes to mind is whether the memory holds the data all the time, whether the particular system is on or off. Accordingly we talk about a volatile memory versus a non-volatile one. A volatile memory essentially would mean that the contents of the memory will get erased if the power goes off, whereas a non-volatile essentially means that even if the power is not there, the memory will still hold the contents; that means, let us just loosely say, the memory holds the data. Of course, these are linked to the technology. Generally you would find this non-volatile or magnetic – you do not need power all the time for the information on the magnetic tape to be there. Even if the power to the computer system is off, the contents would still be available. But then, there are ways in which you can make use of volatile elements and then give a back-up, so that if the main power goes, a back-up power comes, nevertheless power is always needed. This as I said is linked very much to the technology also from the functional point of view because we will keep coming across this particular thing again and again. Generally the memory subsystem will consist of two types. One is called a ROM or an expansion of that is read only memory. From the term itself you can understand that from this memory, we can only read the contents.

The RAM, the other part, is essentially a read by writes memory; the expansion of this is actually random access memory; but really speaking, we have to call this read/write memory, when you compare with this memory. Why? For instance, if the memory consists of read only memory, what we mean by that is the CPU places the address and it can only read. So the contents from this will be passed on to the CPU, whereas in the case of read/write, either you can read as before from ROM or you can write. That is, the CPU can place data and write it. Actually the RAM or random access comes about mainly because any location in the memory can be accessed at the same rate. That is, whether you access location 0 or location 1000 or location 2000, the time taken for the CPU to access that and get the data, that is, the delay between the time the address is placed and the data generated is the same. So it can be accessed randomly. Strictly speaking, the counterpart of this, instead of random access, you can talk about a serial access, in which case, if it is serial access, I can access only the first block, and then only the second block, and then only the third block and so on, serially. So the time taken to access the first block will be much less compared with the time taken to access the tenth block in case it is serial access. Anyway, we will not bother very much about that unless, we come across that again.

So RAM essentially is a read/write memory, that is, functionally speaking, and ROM is the read only memory – these are the two things you will come across. Another thing that we could find based on the technology is semiconductor memory; for instance, the RAM subsystem or the ROM subsystem will all be coming under this particular category because they all make use of the memory elements; they make use of appropriate semiconductor circuits. Now in contrast with semiconductor memory, we can talk about magnetic memory; but generally we do not say magnetic memory because essentially it is used for not the kind of memory which I mean; normally whatever CPU can access immediately is called memory, whereas in the case of magnetic, it is a little farther away as we had seen early in this series of lectures. We used magnetic things for storage purposes, magnetic tape, magnetic disks, disk subsystem, tape subsystem: these are all essentially for storage, which is further away from the CPU; that is, closer to CPU is the memory and then further away from memory, that is, furthest from CPU, we have the storage subsystem. Magnetic storage is also something.

Then depending on how exactly the data is stored in the semiconductor memory, we talk about static memory and dynamic memory. For instance, let us say we talk about RAM. You can talk about the RAM subsystem, RAM memory subsystem consisting of static RAM chips or dynamic RAM chips. What exactly is this particular thing about? Briefly, we know that binary data is going to be stored in each cell. Now let me go back and then try to define a little. Suppose this location is going to store a byte of information, then we talk about an 8-bit storage in each of these locations. So we say this memory cell consists of 8 storage cells, each storing 1 bit. Now when we come down to that 1-bit level, it consists of 8 things, each storing just 1 bit. That particular bit may be either 1 or a 0. That storage cell, the single bit storage cell, could hold that information in one or two forms. One is as long as the power is on, as long as memory chip or memory subsystem has been powered, there may be an element in which it always conducts and then it indicates that the information stored in it is 1 or 0.

(Refer Slide Time 29:49 min)

It is possible that when the power is removed, that information is lost. On the other hand I think may have to go a bit into some circuit details; only then can it be really explained. So I will do that. Suppose I have a storage cell consisting of, let us say, two transistor elements. Suppose the information 1 or 0 is available from this part of the circuitry, the way I have connected is like this: if this particular transistor is on, then this particular transistor is off. It is also possible to make this off transistor turn on and then thereby turn this particular thing off. So by having this transistor in on or off position, I would represent the information here. In other words this transistor when it conducts, or when it does not conduct, you can see that if this conduct, this one does not conduct. On the other hand, if this conducts, that does not conduct. So in other words, there is always a current flow either through this or through this of whatever information is stored in that. Now the power is removed; current flow goes with that; and information is lost. That, in a nutshell, explains the principles of the static RAM.

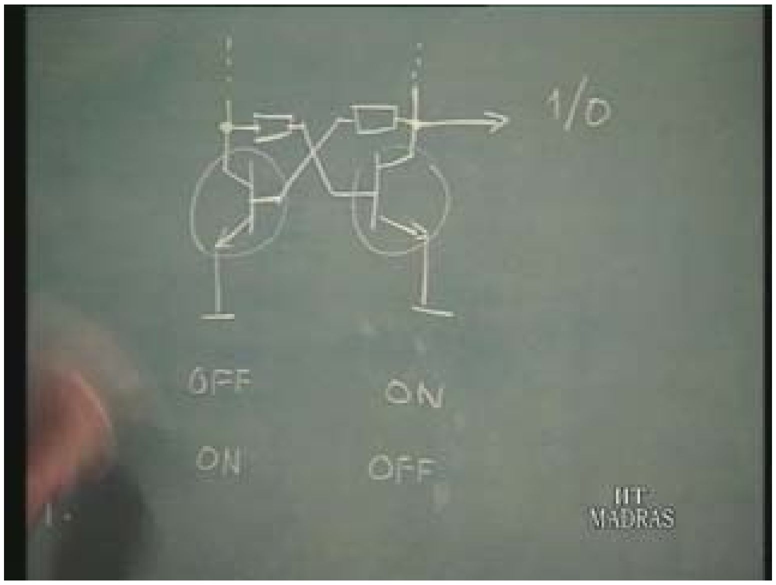
The current flow must always be through something, and when the power goes the information is lost, whereas in the case of dynamic RAM, what is exactly done is essentially again the transistor elements will be used but essentially what happens is it could amount not for an active element like a transistor, but a passive one such as a capacitor. That is, in other words, the capacitors that are available with the active element will be used and suppose this capacitor is being charged – I call it 1 – and later on when this is discharged, and then I may say the information that is stored in that is 0.

Now once charged, even if the power is removed, the charge would remain at least for sometime. So that in fact is the principle based on which the dynamic RAM, the particular circuit, works.
But the problem is when charged; the charge will not remain for long. So this could call for frequent charging. In other words what is required is that we have to keep refreshing the charge and thereby retaining the information we have stored in it. Suppose you charge and then you say by charging I am retaining some information, say 1, but then you can remove the power source, but then by its discharge normally the RC value associated with that, which is a time constant, as per that the charge will be lost. So after some time to ensure that information is not lost it will have to be recharged again this has to be done very frequently. Based on this, you can see really that the physical part influences some of this memory subsystem design. We cannot always keep talking about the logical aspect saying 1 and 0 and so forth.

Depending on the kind of memory chip you are using with which you are constructing the memory subsystem, the appropriate controller will have to take care of the aspect that whatever is stored must be retained. It would depend on the physical part of it. The cost of RAM is very much decided by these: the speed, the capacity, cost for each of these – these will be varying. Usually you would find the static RAM is quite fast but very costly. Dynamic RAM will be slow; you can have higher capacity, and the cost also is less. So you can see that the speed, the size of the capacity and the cost would depend on the technology of these chips, which could ultimately define the cells of this memory.

Now let us go to the functional part, back to functional part. Really speaking, when we discuss this controller, we have to take into account quite a few of these aspects also, but we will not do it in this particular course, not certainly in detail. Let us go back and see what we have over the address; the address, which is the input to the memory subsystem, and how exactly this data comes forth; we will go into details of this. Talking about the details, first we will take a look at this address part – that is the first input. Suppose you have an n bit address, you are always seeing that we have as many as 2^n addresses. That means we can define something that is called an address space, a total address space, consisting of 2^n, total address space is 2^n.

(Refer Slide Time 32:26 min)

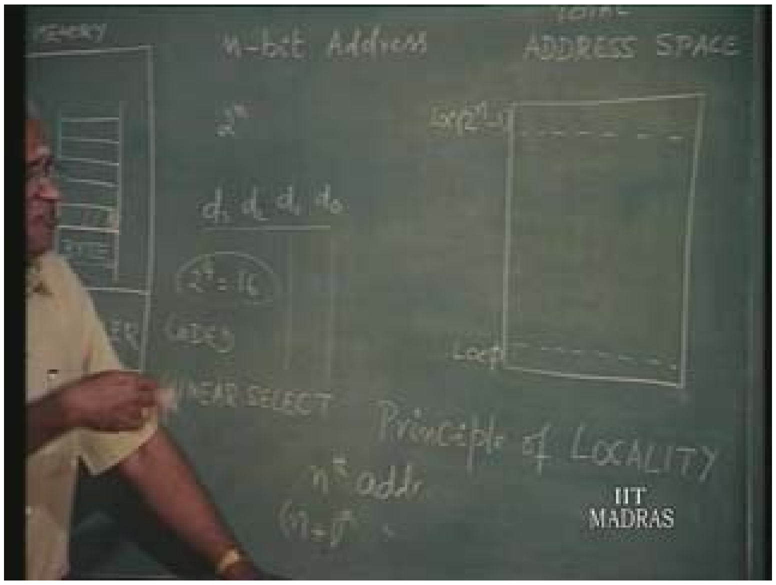


So starting from location, let us say, 0 to the top-most location, since we said 2^n, the top-most one will be 2^n minus 1. That comes because of what we may call the coding of this – we have taken an n bit address and by decoding we are getting 2^n. Now while discussing a few problems, we had come to this: we had talked about this earlier remember, that is, I was doing with the 4 bit. Suppose we have a 4-bit information, or 4-bit data, I will just call it d_0, d_1, d_2 and d_3. With this 4 bit, we can have as many as 2^4, that is, 16 combinations. On the other hand each of these bits can also be used independently and then only arrive at 4.

We are calling this particular scheme as making use of the code, or coded scheme, in the other one, if you take just each of these bits, then we will call it a linear select scheme. In fact this linear select scheme is very relevant when we discuss the memory because with an n bit address, either you can have a total address space like this or we can have only n locations, each of them identified by one of these bits. Now in this particular example, with 4 bits we can have as many as 60 of this total address space or only 4 locations which could be as per the linear select.

We may use combinations of this later on; we will come across that a little later. For the present let us note that the total address space is much. Where is this n bit address coming from? It is coming from the CPU. In case we have a 16-bit address, then 2^{16} could correspond to say 65 K, normally we say 64 K. Should the entire 64 K address space be filled in by some costly RAM? It is not necessary because it is after all the CPU, which is addressing the memory and the memory in our case essentially consists of programs and data, and the program execution is instruction by instruction, which means when one instruction from a location is taken, it is very likely that the next instruction is taken from the next location; in other words, there is something called a principle of locality.

(Refer Slide Time 42:56 min)



That is something which is local and it is likely to be used more immediately and often that is what the principle of locality states. That is, if an instruction is taken from nth address, then it is possible that the next instruction may be taken from the neighboring one – that is n + 1 address and so on; that is what we mean by locality, something which is local; there is a high probability of this. So the entire address space need not be – from physically implementation point of view – made use of through a physical implementation of having let us say costly RAMs. Why are we discussing about the cost part of it here?

We had noted that CPU interacts with the RAM, and that is the memory we said. The CPU interacts with the memory and the memory would consist of, let us say RAM. It can also be read only memory. Then we are also mentioning about the speed difference between these two; that is, when the CPU addresses and the memory is not immediately responding then the memory is going to issue a signal saying wait and thereby it will delay the CPU's processing; in other words the processor utilization will come down.

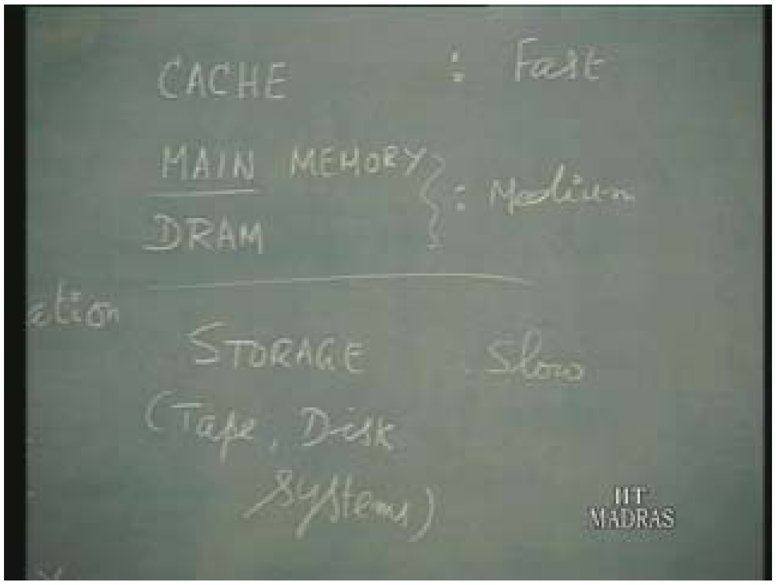
The processor will not be fully utilized; so to see that the processor is fully utilized, in other words, to see that the processor utilization is high, it is necessary for us that efficiency of the system will be high, the throughput of the system will be high and so on. We would like to make sure that the memory that the CPU interacts with is as fast as a CPU itself. To ensure that particular thing and also taking into account the principle of locality, we need not have the entire address space populated by fast memory because the entire thing will not be used. We can talk about the next 100 addresses and next 1000 addresses and so on.

(Refer Slide Time 45:48 min)

It is certainly not likely more than next 2000 addresses, in which case, it is enough if the 2000 locations are accessible by the CPU immediately at the same speed in which the processor does. That is, whenever the CPU accesses the memory, within those 2000 locations there will not be any slowing of the speed, the CPU will not be held up because of want of data because the memory is not able to respond and so on. So the address space will subsequently be divided into different chunks. We can make use of fast memory; now remember what we were talking about earlier. We were saying that the speed, the size of the capacity and the cost – these are three factors which are interlinked. So when we say fast obviously the cost is will go up and because the cost goes up we would possibly restrict the size. So we may make use of fast memory and at the other end, we may make use of slow memory and possibly in-between we will have some medium fast memory.

In other words, we are getting into what is known as memory hierarchy, that is, layers and layers of memory, different types of memory. For instance, we could ensure that the CPU interacts more often with this fast memory, so that the processor utilization will not come down. Whenever the CPU does not find the necessary data in this part, then from the medium fast memory, the information, which will be the data, will be brought into this part so that CPU can access. Even then, if the information is not available, then, from the slow memory it is brought it out. So this is what we mean by having a hierarchy of memory; essentially it is from the point of view of increasing the processor utilization.

(Refer Slide Time 49:49 min)



CACHE : Fast

MAIN MEMORY
DRAM } : Medium

ation

STORAGE Slow
(Tape, Disk
Systems)

Based on this what I have written is the fast memory generally is called a cache memory. Cache is essentially the term used to store something in temporary thing for easy and quick access. So that is the fast part of it; the medium part of it essentially consists of RAMs; that one is generally called the main memory, which would consist of RAM – the cache also consists of RAM actually – but generally this main memory consists of dynamic RAM chips.

You see we are talking about dynamic RAM and static RAM. So either it is called a DRAM memory, main memory or just DRAM. Let me write down: cache is the first part of it, the medium fast part of it is the one consisting of DRAM, and the CPU may not directly access this slow part of it because it is too slow. If the CPU has to access, this utilization will come down very low. So this slow part is generally reserved for this storage subsystem; that is where you find tape, tape disks and so on and so forth, disk systems, tape systems and so on. The CPU will not usually directly have access to this; the CPU will always have access to this fast one only, which means, in case the CPU does not find what it wants here, there must be some other part of the CPU or a different circuitry, which gets the data from here and puts it here so that the CPU may access it.

So by making sure the CPU always accesses only the fast memory, we are all just making sure the processor utilization is kept as high as possible. So in the memory there are three things: the cache memory, which incidentally will make use of static RAM chips like we had DRAM chips. This particular one will make use of static RAM; it is very fast actually. Static RAM chips will be made use of in designing the cache memory subsystem and DRAM chips are made use of in main memory. This is just called DRAM; and then we have the storage. This is too slow and the cost part of it is very costly. This is medium, relatively cheaper than that, and inexpensive.

Generally we talk about so many dollars for bit when referring to the cost part of it, and also whatever is the size part of it, being the fastest and the costliest, this will be less in size. This will certainly be bigger in size and this somewhat medium – just to give a rough figure; Let us say if you have say 1 K cache, we may talk about one K or 16 K in the RAM and this will go into many megabytes or gigabytes. So you can see that the CPU essentially accesses the cache, so the entire address space may not be used by the same kind of memory in different types of memory systems.

Before we close, we could make note of just one small point. We said n bit address is generated by the CPU and the address space is 2^n. It is possible that the address may be 2^n, but we may be having the memory size of more than 2^n. But physically, with n bit of only 2^n, addresses are possible; so the memory can be only 2^n, but when you have more than 2^n, we talk about a virtual memory system. That is, physically it does not exist as far as the CPU is concerned. A CPU can have access only through its n bit address vector, 2^n, but the memory size can be more than 2^n, and that comes under the virtual memory system. We will talk about it later on.