

Computer Organization and Architecture: A Pedagogical Aspect
Prof. Jatindra Kr. Deka
Dr. Santosh Biswas
Dr. Arnab Sarkar
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 08
Main Memory

(Refer Slide Time: 00:35)

Units in the Module

- Components of Central Processing Unit (CPU) and External Interface
- **Main Memory**
- Instruction Execution
- Instruction Format
- Instruction Set
- Addressing Modes
- Flags and Conditional Instructions
- Instruction: Procedure CALL/RETURN

So, welcome to the second unit of the module on addressing mode, instruction set and instruction execution flow. So, in the last unit, we have seen; what are the basic components of the CPU, and the external interfaces, and the basic memory structure, and then how they are all integrated. Now, we will go to the in this module as we have said that we will cover the basic idea of how instruction set is executed, what are the instruction set formats etcetera. So, we are now going to the second part that is required to understand the execution of the memory, execution of the instructions and what are the different instruction modes that is the main memory. Because as you know that we are all studying about Von-Neumann architecture, so in that case all the data and the instructions are present in the memory.

So, even if you want to understand about instruction execution, the format, the sets and different addressing modes and so forth, you have to know the idea of how they are all organised in a memory. So, the second unit of this is the main memory which we are going to look today.

(Refer Slide Time: 01:26)

Unit Summary

- Digital computer works on stored programmed concept --Von Neumann. Memory to store the information, which includes both program and data.
- The memory of computer is broadly categorized into two categories: Internal and external. Internal memory is used by CPU to perform task and external memory is used to store bulk information, which includes large software and data.
- The memory hierarchy is given by:
 - Register: A Part of CPU, so they reside inside the CPU
 - Cache Memory: Storage device placed in between CPU and main memory
 - Main Memory: CPU can work with the information available in main memory only. We have to first bring the information (whether it is data or program), to main memory. Like cache memory, main memory is also semiconductor memory. But the main memory is relatively slower memory compared to register and cache.
 - External Memory (Hard Disk): This is bulk storage external device and is used because capacity of semiconductor memory is limited.

So, now as we have told that this course is delivered in a pedagogical manner or pedagogical paradigm, so we will see what we are going to see in this unit that is the unit summary. So, as I told you we are all looking at the von Neumann architecture concept which will consist of both the data and the program and the data.

So, you have to know about the idea of what is a memory what is the main memory, and what is a secondary memory etcetera, and how it is addressed, and what is in a nutshell how it the or in a black box how the CPU interfaces with the memory. There will be in fact a detailed whole module on the memory design in which you will know the more internals of the how the memory works, how it made up of, how the codes and data are organized there, so that is a dedicated module for that. But in this module we look only the black box level view of a memory which will be required to understand how work would executes.

So, basically if you look memories are divided into mainly two types internal memory and external memory. So, internal memory basically is the semiconductor kind of a memory in which case you have a register. So, register is a part of the CPU itself. So, as we discussed in the last units, so this something like if you want to add two numbers so basically they have stored in a memory in a memory which is called the register; that means, something called a cache memory and a main memory. So, actually main memory is the word we have always heard the word called RAM. So, in a lay man language RAM, there are lot of technicalities we

will come into, but in a lay man language what is known as a RAM is basically your main memory.

So, basically your CPU or your arithmetic logic unit of the main which is the computing unit of the CPU, basically it can talk only to the main memory that is it can generate the address and then it can read and write data from the main memory. But actually in between the main memory and the register, so you can think that the CPU or the computing unit is mostly closely attached to something which is called the register. Everything all the operations mainly they compute at the register level. Because if you add two numbers generally one the temporary data or the result to be first stored in the registers and then they will go to the main memory. So, basically it is the most near part of the CPU, but actually the cost of having too many large number of registers is very high. So, therefore, the bulk code of the data executes mainly in the main memory which can be addressed by the CPU.

But there is another memory which lies in between the CPU and the main memory is called the cache memory. So, we will learn in more details about cache memory when we will going to into the full module on memory design. But the basic idea is that whenever you want to refer to some data, generally the address is generated for the main memory and as main memory is much slower compared to a register there is something in between which is the cache. So, the cache will get some of the memory, some of the data from the main memory put it into the cache and they will be used.

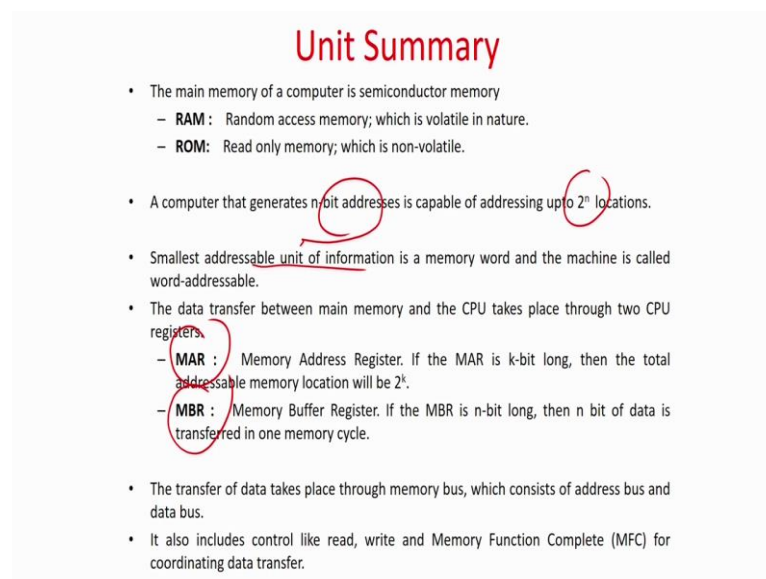
And again when the data which is in the cache has been exhausted or you require any more new data, it will coming from the main memory. So, in fact as of now if you understand the concept of a register and the concept of a main memory, it will be sufficing to understand the concept of this module and also the unit here. So, in fact, you have to understand that the CPU will generate some memory address which corresponds to the main memory. And the data will be coming from the main memory to the register, for the time being cache is transparent for our case and so forth.

And there is something called the last or the most top level in the memory which is called the hard disk that is your external memory. So, external memory can be further more if you go down the line there can be tape drives etcetera and there can be bulk SSD hard drives etcetera so, but for our case we are calling in a nutshell that is the hard disk or the external memory

where the whole data will be stored. So, basically this is not a semiconductor type of a memory generally a magnetic memory if you are in a broad way.

So, basically what happens is that whenever you want to execute a code we execute it from the memory locations on the main memory only. And whenever you want have something to be loaded which is not in the main memory then they are all copied from the main memory to the sorry from the hard disk or the external memory to the main memory and then the code executes. So, the idea is that if you if the CPU wants to generate some address, the addresses are generated mainly for the main memory. So, if data is not available in the main memory or some code or some instructions is not available over there, it is brought from the external memory, but external memory addresses are not generated as a part of the execution of the code. So, for us most important here are to understand the register and the main memory; others are actually in fact transparent to understand to this module.

(Refer Slide Time: 05:52)



The slide is titled "Unit Summary" in red. It contains a bulleted list of computer concepts. Handwritten red circles and lines highlight specific parts: "n-bit addresses" and "2ⁿ locations" in the second bullet; "unit of information" in the third bullet; "MAR" and "MBR" in the fourth bullet's sub-points; and "registers" in the fourth bullet's main text.

Unit Summary

- The main memory of a computer is semiconductor memory
 - **RAM** : Random access memory; which is volatile in nature.
 - **ROM**: Read only memory; which is non-volatile.
- A computer that generates n-bit addresses is capable of addressing upto 2ⁿ locations.
- Smallest addressable unit of information is a memory word and the machine is called word-addressable.
- The data transfer between main memory and the CPU takes place through two CPU registers.
 - **MAR** : Memory Address Register. If the MAR is k-bit long, then the total addressable memory location will be 2^k.
 - **MBR** : Memory Buffer Register. If the MBR is n-bit long, then n bit of data is transferred in one memory cycle.
- The transfer of data takes place through memory bus, which consists of address bus and data bus.
- It also includes control like read, write and Memory Function Complete (MFC) for coordinating data transfer.

So, now let us go into what will be next in the unit summary, which will be more important to us are basically the main memory. So, main memory is a semiconductor memory as I told you that there are two types basically one is RAM and one is ROM. So, the RAM is the random access memory, basically it is volatile; and ROM is the read only memory, but both RAM and ROM are basically random access only that is that is not a sequential access, you can access any unit or any word of that memory. But it is from the means the name has been carried over

as for ROM also can be access random manner, but read only memory means you cannot write anything over there and it is non-volatile.

So, generally the BIOS or some important parts of your code or which is required for the machine to boot up which cannot be changed or in fact you should say that even if the machine is in shutdown mode, when it comes back to power we will require those codes to be executed are on the read only memory. And mainly the random access memory is exactly what we understand by the term that which is used and reused of loading your code, loading the data, changing the data and again writing it back is the random access memory.

So, now if you look at how basically it works. So, generally CPU will generate an n bit address, because the memory will have some locations which can be done by the address. So, generally there are 2^n locations, where n is the number of address lines. So that means, if there is n lines in address bus, so the number of locations will be 2^n ; and by giving an address you can access any one of the data in that location.

And again the smallest unit of information is called a memory word that is a single bit may not be able to be accessed over memory, if a memory can be organized say for example, $2\text{GB} \times 8$, that is 2 gigabytes. So, into 8 means the memory is organized with each word is having 8 bits. So, at a time you can only read 8 bits out of the memory in one cycle. You cannot go for 4 or you cannot go for 16 so that is actually called a minimum smallest addressable information of a memory. It is not that in a word in a memory, which is organised as byte, you can access a single bit so that is the way how can you access it.

And in fact there are two very important registers one is the memory address register and one is the memory buffer register which will be used to access the memory. So, in fact whichever word you want to access that address will be given in the memory address register and whichever data you want to read or put it into the memory that is written in the or read into the memory buffer register. And in fact data transfer happens through the data bus and address is sent by the address bus. And there are some control signals like you want to read the memory, you want write the memory and then some synchronization that is read is over, write is over all those things. So, there are some control lines.

(Refer Slide Time: 08:35)

Unit Objectives

- **Application: Demonstrate:**--Demonstrate the use of semiconductor memories which are used for designing the main memory of a computer.
- **Comprehension: Describe:**--Describe how to address a particular memory location of the main memory.
- **Comprehension: Explain:**--Explain the connection of the main memory to the processor through system bus.
- **Analysis: Determine:**--Determine the size of a memory module. Explain the size of a memory location - byte addressable, word addressable, etc.
- **Comprehension: Explain:**--Explain the memory read and memory write operation.

So, in a nutshell this is how was the basic summarized notion of a memory which we are going to see in this unit. In fact the details of how a memory is constructed, how exactly data is organized, if there is some data which is absent in the main memory has been brought from the hard disk, if some memory is not available in the cache it has to be brought from the main memory all those details for that there is a dedicated module given on memory. We will study it later.

So, what is the objectives of this unit, after doing this unit or studying the unit you will be able to demonstrate the use of a semiconductor memory which is used in designing the main memory of a computer that is why it is required the basic idea you will be able to demonstrate. Demonstrate in the sense that if there is a code how it is stored in the memory, how it is accessed and in what manner the CPU can interface with the memory. Then next it will be a comprehension objective you can able to describe how a particular memory address is accessed. Then you will able to explain the connection of the main memory through the processor to the system bus that is data bus, address bus and control bus, then analyse you will be able to determine the size of a memory given its configuration that is byte addressable word addressable etcetera that if I tell you the memory is $1\text{ GB} \times 8$. So, what does it mean, so that configurations you will able to demonstrate. And finally, you will able to explain the read and write memory operation that how it is done. This unit is basically giving a very broad idea of a memory main memory that is used to understand how a code or a instruction is executed. More details will be in a separate module.

(Refer Slide Time: 10:01)

Semiconductor memories

- Main Memory is generally implemented in the form of RAM (Random Access Memory). RAM is volatile
- **SRAM (Static RAM):** Access time (i.e., read and write) is fastest in SRAM compared to other variations. However, it is expensive because it requires comparatively larger area (i.e., 6-transistor circuit for memory cell). Due to its cost it is not very suitable for main memory (and is generally used for cache memory).
- **DRAM (Dynamic RAM):** Access time is higher than SRAM, but is cheaper as it can be implemented using only a single capacitor and single transistor. There is another issue with this memory, because capacitors lose their charge and hence DRAM needs to be refreshed every few milliseconds. As of now, DRAM is the most widely used for main memory implementation.

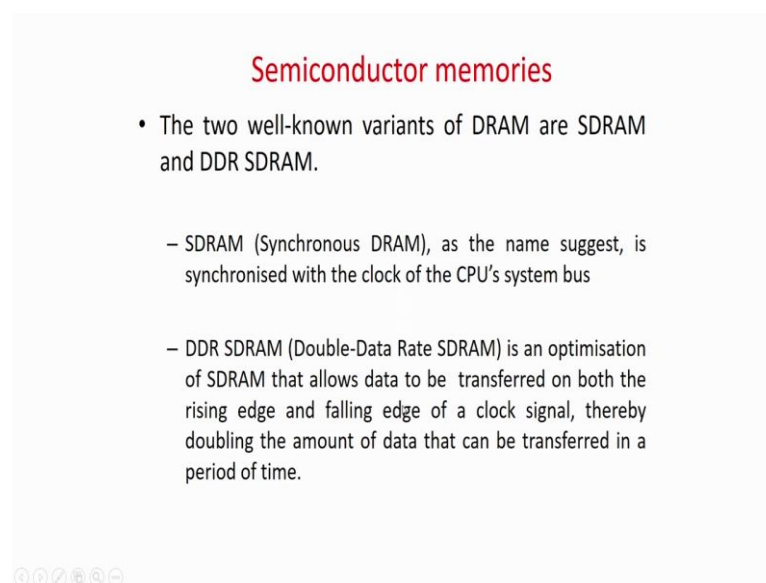
So, basically we start there are two types of memories, one is a semiconductor memory and one is a basically non semiconductor memory that may be a tape drive that can be a magnetic memory, which is generally consists of the external memory. But more important to us is something called the semiconductor memory because semiconductor memories are faster, you can build them on the chip, but again they are expensive. So, you cannot have a 1 terabyte RAM that is will be not possible by today's fabrication technology, but it will be extremely faster.

So, what we will do is that they actually gradually if you go down the memory hierarchy that is if you go from the hard disk to the RAM to the main memory then to the RAM and then to the register, the size will come down, speed will increase. And of course, the cost will increase that is what is the idea. So, basically the main memory which is mainly we are concerned about and the registers, registers are mainly made up of flip flops, but the main memory which we are mainly concerned about at present are basically of semiconductor.

So, basically there are two types of memory one is the static memory, and one is the dynamic RAM. So, static RAM the access time is faster in SRAM once you have to any other variation. Actually it's an extremely fast generally it may be used for your means register design but the size will be slightly larger because they are actually taking six transistor circuit design. As I told you at present in this unit we are not going to depth of how the six transistors are organized, you will be learning in details in later modules of the course.

There is something called the DRAM which is the dynamic RAM. Its access time is higher in fact that is more cheaper to implement than a static RAM, but the idea is that it actually is implemented by a single capacitor and some register size will be smaller than the static RAM. So, you therefore, you can build larger size of DRAMs in the same price compared to an SRAM. But time access time will be slightly higher, but still it is doable. So, in most of the modern days, you can understand DRAM is the most widely used technology for main memory implementation. Static memory size is bit high, so you can think that we can use it for designing of some high-speed memories may be in the registers or something like that.

(Refer Slide Time: 12:07)



Semiconductor memories

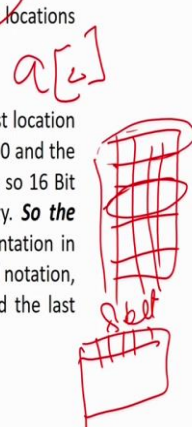
- The two well-known variants of DRAM are SDRAM and DDR SDRAM.
 - SDRAM (Synchronous DRAM), as the name suggest, is synchronised with the clock of the CPU's system bus
 - DDR SDRAM (Double-Data Rate SDRAM) is an optimisation of SDRAM that allows data to be transferred on both the rising edge and falling edge of a clock signal, thereby doubling the amount of data that can be transferred in a period of time.

As I told you most of the RAMs are implemented using dynamic ram. So, there are two type of dynamic RAMs, one is synchronized DRAM and one is DDR RAM. So, synchronized DDRAM is as the name suggests, you can read the memory location at every edge of the clock, it may be positive edge it may be negative edge buy in any one edge you can read or write it. And that is actually it is say that it is synchronous RAM which is synchronized with the operation of the clock. Double SDRAM is also synchronized operation, but it is faster or doubly faster than the SDRAM because in this case you can read and write the memory at both the edges of the clock that is you can also go for the rising edge as well as you can do the falling edge. So, the speed of this RAM will be higher.

(Refer Slide Time: 12:47)

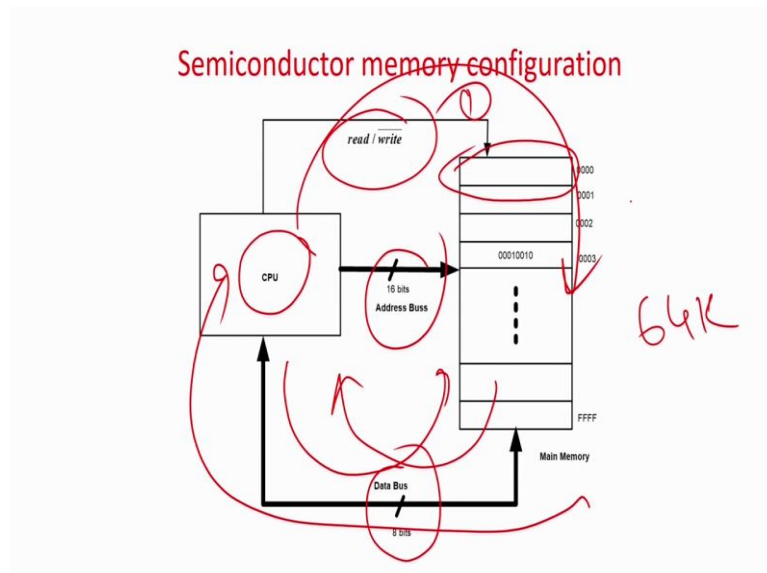
Semiconductor memory configuration

- The configuration of the main memory is $64K \times 8$ Bits. It implies that there are $64 \times 1024 (=65536)$ memory locations and each location (called word) has 8 bits.
- 65536 locations are referenced from 0 to 65535. First location is referenced by Binary number 0000 0000 0000 0000 and the last location is referenced as 1111 1111 1111 1111; so 16 Bit binary number is required to address this memory. **So the address bus is of size 16 Bits.** For ease of representation in Assemble level instructions we use in Hexadecimal notation, where we reference the first location as 0000_H and the last location as FFFF_H.



So, now we are going to look at what is a memory configuration, if somebody tells you that I have a generally memory configurations are written in a product type of manner. So, for example, they have written that $64k \times 8$ bits that means what, in this case I will show you the figure of a memory first.

(Refer Slide Time: 13:04)



So, the memory basically looks like this, this is an array, or you can just think that memory is basically is an array. So, let me first go back, memory actually looks like an array. For the very simple notion you can think it is an array, but there is only one difference is its two-dimensional

array I should say that it is basically a two-dimensional array. So, for example, we can access $a[2][3]$. So, you can this you can easily do in this c program, but here is slightly interesting that you cannot do this bitwise access. Because I told you in a memory the difference is that you have to access one word at time or in fact you can think that it's a 2d memory but to read and write you have to do one complete row at a time. So, the row is called the word in terms of memory technology.

So, in fact if I say in that way you have to actually drop this through so that means, you can access location number to 0 1, 2 then this whole row will be read in the memory or you can write it in the memory, individual bit access is not possible. So, you can remember whenever in this manner that is a 2d array is no problem about it, but only thing is that you cannot read a single bit out of it, you have to the read whole row at a time or write at a time.

So, here it is saying that it is $64k \times 8$ bit memory. So, what does it mean it means that the 2d array looks like this is 8 bit that is row size is 8 bit or 8 units are there in each each unit you will be able to write either a 0 or a 1. So, there are 8 bits over there and that is actually called the one word.

And then what is the number of rows in the memory, so that is actually given by the first. So, it is saying it is 64 k so that means, it is actually 64×2^{10} that is 64×1024 that is something like that is right. So; that means, we have 64×2^{10} that is 64 k that many memory location that means that many rows will be available. And each location has 8 bits that is actually called the word, but in fact the very similar we can write it as an array of size 65536 8, but only thing is that this 8 individual bits cannot be accessed you can address the whole 8 bit at a time. So, in fact this will be called as a byte accessible memory because one word or one byte can be accessed here.

And how do you read out the locations. So, how do you access the memory? So, in fact memory locations start from 0 to 65535, and the first location is all this and the last location is this. So, that is we say that whenever this is the number that is the number of rows is this one. So, it is actually 2^{16} .

(Refer Slide Time: 15:41)

Semiconductor memory configuration

- The configuration of the main memory is 64K x 8 Bits. It implies that there are $64 \times 1024 (=65536)$ memory locations and each location (called word) has 8 bits.
- 65536 locations are referenced from 0 to 65535. First location is referenced by Binary number 0000 0000 0000 0000 and the last location is referenced as 1111 1111 1111 1111; so 16 Bit binary number is required to address this memory. **So the address bus is of size 16 Bits.** For ease of representation in Assemble level instructions we use in Hexadecimal notation, where we reference the first location as 0000_H and the last location as FFFF_H.

Handwritten notes: $2^{16} - 1$, 0000_H, and a red arrow pointing right.

So, if you go for 2^{16} then you are going to get this number minus 1 of course, then actually your bits I mean bus address bus size is 16 bits that means, you have to access words from 0 to 65535. So, you need a binary numbers which can access numbers from all 0s to 65535 that is the value. So, it's a 16 bit number all zeros means the first memory word and all ones means the last memory word. And therefore, the address bus from where the address we will be giving will be a 16 bit bus which actually comes from this number.

And again, but generally it is very cumbersome to write these values. So, we write in a hexadecimal notion that is 0000 hex that is the first memory location and FFFF is the last memory location. So, any value if you give, so if I give 0001 hex that means, I am accessing the first memory location 0 is this one first memory location I am going to access. So, if I write 0001 hex as the first memory location is addressed, and depending on the control value of read or write, you will be able to either read all the 8 bits that is very important to note compared to a 2d array, all the 8 bits will be either read or written at that memory location ok.

(Refer Slide Time: 16:55)

Semiconductor memory configuration

- As each location has 8 bits (i.e., data) the data bus is 8 bits.
- Also, there is a single bit control signal to configure the memory as read (i.e., transfer data from a memory location, whose address is given in address bus, to data bus) or write (i.e., transfer data from data bus to a memory location, whose address is given in address bus).
- The figure given below illustrates the details of the hypothetical CPU connected to the main memory.

So as I told you, each location has 8 bits. So, the data bus will be 8 bit bus. So, now when you address so you are referring one memory location, now you have to read or write data from that. So, we require another bus which is actually called the data bus. Now, what is the length of that bus or the width of that bus sorry the width of that bus, it will be the second term in this case you can access word whose size is 8 bits. So, the word size will be 8 bits. So, if the memory is $64\text{ k} \times 32$ that means, it will be four byte addressable memory. So, the address bus size is 32 bytes bits.

And again very important basically we are doing two stuff at a time either we are reading the memory or we are writing the memory. Reading the memory means we are taking whatever the value is in the corresponding memory location the address to the CPU and write is the reverse direction. So, there should be a control line to tell whether it is a read or a write memory. So, there will be a single control signal which will tell you whether to read and write.

And the figure below actually this one is our memory, I will come to that again I am saying that how the memory cells are organized so that a single word you can single control line will make it read or a single control line value change will make it write depends on the how the memory is implemented. So, all the details of the internal memory structure we will be looking at in the future, but for the time being the knowledge we are giving for the memory in a very nutshell will be enough to understand how a code executes, so as I told you this is the CPU it will generate 16 bit address.

Now, why a 16 bit address because the memory size is 64 k which is nothing but 0000 to FFFF; and we know that this is actually 8 bits or byte addressable so the data bus which is a bidirectional bus will be 8 bits. Writing will be from this, reading will be from this side and this is a control signal. So, if it is read-write bar that means, if this signal value is 1, then the memory will be in a read mode, this way; and if you make this as 0 then it will be a write mode so data from here will be coming over there. So, in a nutshell a memory looks like this.

(Refer Slide Time: 18:50)

Semiconductor memory configuration

Total size of the data is $2\text{GB} = 2^{31}$ Bytes (as 1 G Byte = 2^{30} Bytes)
 So, $2\text{GB} = 8 * 2^{31}$ Bits = 2^{34} Bits.

a) Bit Organized:

Size of data bus is 1 (only one bit at every location in the memory)
 No. of addresses = $2^{34}/1 = 2^{34}$.
 Therefore size of address bus = 34 Bits

b) Byte Organized:

Size of data bus = $1 * 8 = 8$ bits (8 bits at every location in the memory)
 No. of addresses = $(2^{34})/8 = 2^{31}$
 Size of address bus = 31 Bits

Handwritten note: $2 \times 2^{10} \times 2^{10} \times 2^{10}$

So, now we will study some of the basic semiconductor memory configuration by examples. So, sometimes let me first take the example of 2GB into that is 2GB. So, what is 2 GB that is $2 \times \text{GB}$ means megabytes sorry it will be kilobytes, megabytes, gigabytes and terabytes something like that right. So, 2^{10} that is your kilobytes megabytes and giga bytes so 2^{30} , so that is 2^{30} so that is nothing but 1GB is 2^{32} GB is 2^{31} . So, as you can see. So, byte bit. So, it is you can see that is 2 GB nothing but 2^{31} 2 GB is 2^{31} bytes that is 2 to that is one because 1 byte as I told you 2^{30} bytes and 2 GB is 8×2^{31} bits because 1 byte is 8 bits. So, the whole memory you can tell as a 2^{34} bits.

But again I told you that is generally we write a memory in terms of $x \times y$ that is x is the number of memory locations and this is the width.

(Refer Slide Time: 19:52)

Semiconductor memory configuration

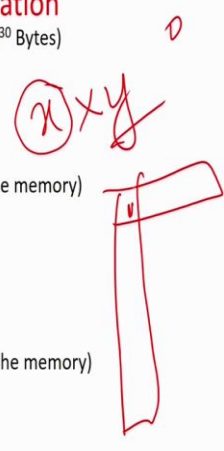
Total size of the data is $2\text{GB} = 2^{31}$ Bytes (as 1 G Byte = 2^{30} Bytes)
So, $2\text{GB} = 8 \times 2^{31}$ Bits = 2^{34} Bits.

a) Bit Organized:

Size of data bus is 1 (only one bit at every location in the memory)
No. of addresses = $2^{34}/1 = 2^{34}$.
Therefore size of address bus = 34 Bits

b) Byte Organized:

Size of data bus = $1 \times 8 = 8$ bits (8 bits at every location in the memory)
No. of addresses = $(2^{34})/8 = 2^{31}$
Size of address bus = 31 Bits



So, if we are writing 2^{34} bits; that means, I am not telling you what is the length of this one and what is the width of the memory. So, you have to tell in which way. So, in this case it is saying the whole size is 2^{34} bits that is again another way of describing a memory, but immediately you have to tell that it is bit organized. So, what do you mean by bit organized; that means, there is only one bit in the row. So, it is a very non practical kind of a memory, just to show the example either you write $x \times y$ or you can tell the total size of the memory that is in this case 2^{34} bits. How 1 byte is equal to 2^{30} bytes 2 bytes 2 GB = 2×2^{31} bits bytes sorry and then 1 byte = 8 bits. So, the whole size is 2^{34} bits, but I do not know what is the size and what is the size over here. So, in this case it is said as bit organized bit organized means there is only one bit at a time.

(Refer Slide Time: 20:49)

Semiconductor memory configuration

Total size of the data is $2\text{GB} = 2^{31}$ Bytes (as $1\text{ G Byte} = 2^{30}$ Bytes)
So, $2\text{GB} = 8 * 2^{31}$ Bits = 2^{34} Bits.

a) Bit Organized:

Size of data bus is 1 (only one bit at every location in the memory)
No. of addresses = $2^{34}/1 = 2^{34}$.
Therefore size of address bus = 34 Bits

b) Byte Organized:

Size of data bus = $1 * 8 = 8$ bits (8 bits at every location in the memory)
No. of addresses = $(2^{34})/8 = 2^{31}$
Size of address bus = 31 Bits

So, in this case; obviously, the whole length is 2^{34} that is 2^{34} memory locations will be there because only one bit at a time very impractical kind of a memory only the idea is given to you. Byte organised this is more or less practical so it is says that this length size is 8 bits, that is an in a byte in bite organized individual bits can be accessed, but as I told you this is a non very not a practical way of implementing because generally means we represent a data in terms of 1 byte, 2 bytes because that is more meaningful.

So, you want to access one meaningful word at a time in this case you have to add may be 8 bits one at a time you have to read eight locations and then merge it and then the understanding the meaning of it. So, it is not a very good way of doing it that is more better that you put 8 bits or 16 bits in one row, get the value and make a meaning directly out of it rather than going in a sequential manner. So, this byte organized.

So, what do you mean by byte organized, there are 8 bits over here. So, what will be the length, it will be $2^{34}/8$ that is the 2^{31} . So, this size is 2^{31} bits that is 2^{31} locations are there. So, in this case what will be the size of the address bus 31 bits, because you need to access all the 2^{31} bits very simple.