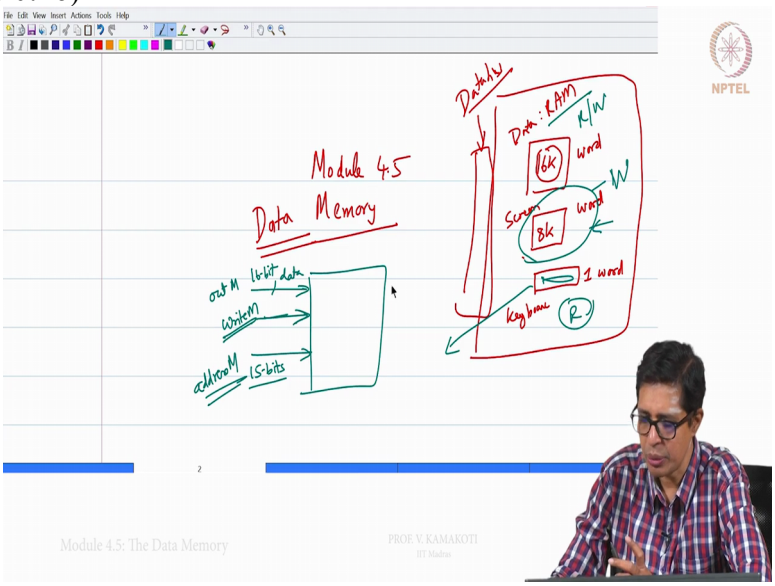
**Foundations to Computer Systems Design**
**Professor V. Kamakoti**
**Department of Computer Science and Engineering**
**Indian Institute of Technology Madras**
**Module 4.5**
**The Data Memory**

(Refer Slide Time: 0:16)




Now welcome to module 4.5 where we are going to talk about data memory. Now for us, data is not just a memory where we read and write data. Data also is something that we write on the screen and something that we read from the keyboard. Everything is viewed as memory operation and that is the concept of what we call as a memory mapped IO. So if you take a normal computer, there is a RAM which is a read access random access memory and that is what we call as data here. So we have a 16 kilo word RAM. Each word is 16 bits for us and there are 16 K such words.

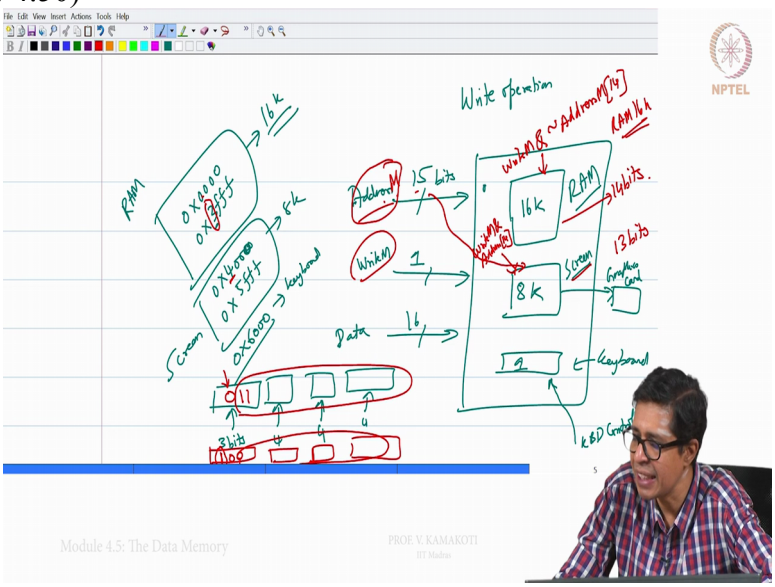
So we have RAM and then we have 1 IO device, 1 output device which is the screen which again I treat as data. And there is 8 kilobyte for that. And then there is an input device which again I treat it is a memory. All right? And that comes from the keyboard. And that is 1 word. So I have 16 kilo words of RAM, 8 kilo words that I have reserved for the screen and 1 word, just 1 word which I have reserved for the keyboard. So I type something in the keyboard, it comes to this 1 word. If I want to write something into the screen, I write into these words, the 8K words that you have seen and read and write from memory is always from the 16 K.

So your data memory encompasses these things, right? So essentially in a system, these 3 are connected by a data bus and depending on the address whether the bus will decide whether I am trying to write into the RAM or whether I am trying to write into the screen or I am trying to read from the keyboard or read from the RAM, right, et cetera. So this is the concept of memory mapped IO and this is how it is organised. Now we have to construct this, as a part of this module, we have to construct this data memory which comprises not just the RAM but the output device, namely the screen and the input device namely the keyboard.

Now what do you mean by a output device here? So I keep writing into this part of the memory, this 8K memory and there will be a display controller which will be reading from this and blackening the screen or whitening the screen. Similarly, whenever I type something, there is a keyboard controller which will read that value and put it into this word so that I could read from this world. Similarly, I can write into this screen. I can read and write into this RAM. So RAM is a read/write memory, screen is only a right memory and keyboard is only a read memory.

So you are seeing variety of you know devices here right? That is part of this. Now how are we to construct this? Now for the data memory, as we had seen in the ALU slide right? So there is something called writeM and there is an address which is coming inside right which is 15 bits. Right? And then there is a 16 bit data that is coming here. This is outM. OutM comes from the CPU, writeM and there is a 15 bits which is address memory. Right? So a 15 bit address comes here. writeM is whether I want to write or not. Right? So when a 15 bit address, so now we will see, we have 2 operations that we need to do.

(Refer Slide Time: 4:50)




How do we do a write operation? First we will see how do we do a write operation. So I have this RAM which is 16 kilo word. I have this screen which is 8 kilo word and I have this keyboard which is 1 word. Now there is an address that is coming for 15 bits and then there is your white memory which is now becoming 1 because I want to write. This is address and there is of course data which is 16 bits. If it is write, then it will be either into the RAM here or the screen here.

We cannot actually write into the keyboard. Keyboard will be always from the CPU side I will be reading from the keyboard. The keyboard controller will be writing into the keyboard while the screen controller, so this is the keyboard controller which is connected to the which is part of your keyboard. This is the screen controller. We will call it as a graphics card. Normally you have a graphics card, that will be connected to the screen. So the graphics card reads from the screen while the keyboard controller writes into this information.

The CPU can write into the RAM and also these 3, the CPU can read from the keyboard. Are we clear? So when we talk of a write operation into this RAM, basically it is I am trying to write either into this RAM or I am trying to write into the screen. So how will I distinguish whether I am writing into RAM or screen? So the whole memory address is now partitioned and allocated to different parts. This is exactly how it happens even in large systems. So the address from 0 0X0000, this is hexadecimal to 0X3FFF, this is totally 16 K, 16 K words.

So the memory address starting from 0000 to 3FFF are allocated to the RAM. Right? The memory address from 0X4000 to 0X5FFF, this is 8 K, is allocated to the screen. The location 0X6000 is allocated to the keyboard. So this is screen. What I am getting is a 15 bit address. Right? So how will this 15 bit address look like? There are the first, these are, this is 4 bits. 4 bits and 4 bits. So this becomes 12 bits. Then I have the last 3 bits. Right? So these are the first 12 bits, least significant bits and the last 3 bits are here.

Now if I am trying to write into the RAM, the maximum address is the last 4 bits are given by these 3 in hexadecimal. So this is 3. So this will be 0011. so last 3 bits, so these will be 011. So if your last 3 bits are 011 or if your, this is the maximum you get. If I am writing into the screen, we start with 4000. So the last 3 bits will be as you see here, this can be anything. The last 3 bits will be 100 because this is 4. So if I am writing into the RAM, the maximum I can get here I can write from 0000 to 3FFF. So the maximum I will get in the last 3 bits is 011. So the last bit will never become 1 if I am writing into the RAM. The last bit will become 1 if I am writing into this screen.

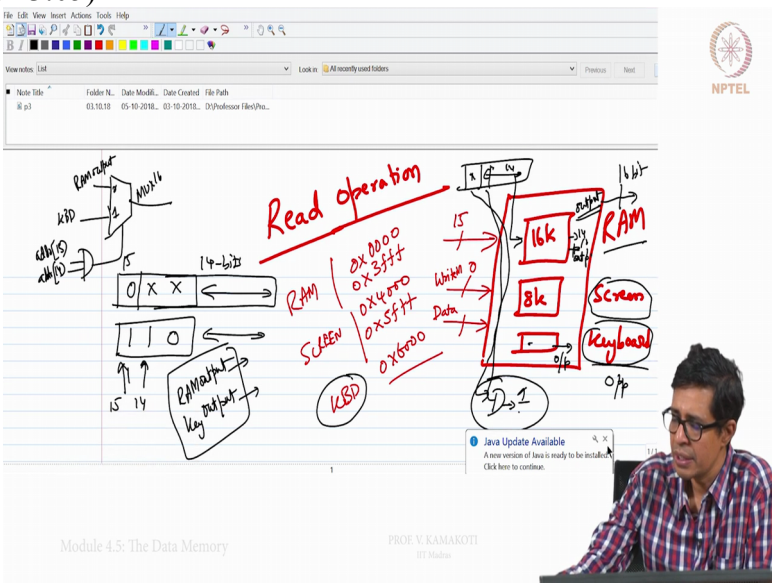
The minimum address for the screen is 4000. So the last bit will become 1 if I am writing into the screen. So for a write operation, I will use the $15^{th}$ address bit. Suppose I call addressM right? address 0 to address 14. addressM 14 which the $15^{th}$ bit, right? addressM0 to 14. So 14 will be the $15^{th}$ bit. I will use the address 14th bit to decide whether I am writing to the RAM or whether I am writing to the screen. Right? So if your writeM is 1 and your not of your addressM is 1, that is addressM is address 14. If your not of your addressM is 1 that is that bit, theni I am going to write into this 16 K RAM.

Where will I write, that address is going to be given by the remaining 14 bits. Because I need 14 bits to address this 16 K RAM. The remaining 14 bits will tell which location I am going to write, the $15^{th}$ bit will tell whether I am writing into the RAM or I am going to write into the screen. So if your writeM is 1, I am trying to write into the memory and your $15^{th}$ bit of your address is 0, that means you write into the 16 K RAM inside that memory. And which location of that 16 K I will write? I will write into that location given by these 14 bits. For 16 K, I need 14 bits. The remaining 14 bits in your addressM will tell you which location I need to write.

So if you look at the RAM that you have created, RAM 16 K, you have created 1 RAM 16 K. You actually need only a 14 bit address and that 14 bits will come from this address. The moment I see that the 3rd bit the 14th bit is 1 so I will write into the screen. When my writeM is 1, I want to write into the memory and my address 14 is also 1. My 14th bit. Then I have to write. And into which location will I write? For the 8K, I basically require 13 bits and those 13 bits are given by the remaining, the first 13 bits. I do not need the 14 bits for this 8K.

So if the 14 the 15th bit is 1, I surely know because I have allocated that address for the screen. If 15th bit is 1, I sure and if it is a write I know surely I am going to write into the screen. And so I will enable writing into the screen there and then which address? In this addressM the first 13 bits will tell me which address. In this case, I need to use the 14 bits but in this case I have to use the first 13 bits. And that will tell me where I have to write into the screen. Then okay this is about write operation.

(Refer Slide Time: 13:05)




So the last thing that we need to is about the that sorry the read operation. So we are getting the addressM, the writeM of course that is going to be 0, we do not care here and dataM. So we are going to basically give an out of this memory. So this is input to the memory. So we call it outM output from the CPU to the memory. This has inM, input to the CPU from the memory. So we have again a RAM, we have a we have the screen and we have the keyboard.

Now this is the read operation. So writeM will be 0 here. Since it is a read operation, writeM will be 0 here. The address 15 bits will come. So for this, we need only the 14 bits. For this you need only the 13 bits. Right? And this this is of course this is 0X6000. So the last 3 bits here will be 110. This is for the keyboard. 0X6000 right, the last 3 bits will be 110. If it is the screen, the last 3 bits should be 100. Right? If it is RAM, it will be 0 and it can be anything, 0XX. So while I am reading from the RAM, as mentioned earlier, the last bit is 0.

This is a 15 bit. But we need 14 bits for accessing this 16 K RAM. Those 14 bits, which address in the RAM I should access, is given by these 14 bits that are following this. This is the $15^{th}$ bit. And if I am accessing memory, I just want to confirm if I am accessing keyboard meaning I am trying to read from the keyboard location, we need to check whether these 2 bits are 1. The $15^{th}$ and $14^{th}$ bit are 1, then we basically read from the keyboard. We do not read from the screen. . So this is the basic status.

So when the $15^{th}$ bit address comes, the 14 bits, so this is a 15 bit address so this is the most significant bit and then there are 14 bits. 14 bits go here right? And so there will be an output generated by this RAM. There will be an output that is generated by this RAM. Similarly, we test the first 2 bits, the 15, 14, $15^{th}$ and $14^{th}$ bit right? Only the $15^{th}$ and $14^{th}$ bit here, we AND them and if that is 1, then we know that we are going to read from the keyboard. So the first 14 bits will be given to this RAM and some output will be coming out of this RAM.

A 16-bit output will be coming out of this RAM. Similarly, there will be an output automatically from the keyboard. So there is RAM output and there is a keyboard output. Now I need to decide between whether I should select the RAM output or the keyboard output. How will I decide it? If the last 2 bits are 1, that is the AND of address14 and address15 is 0 to 14, that is 15 bits. If the last 2 bits and this 1, then I know that I have to take it from the keyboard. If it is not, then I have to take it from the RAM.

So I have a Mux 16 here. I can instantiate a Mux 16 and I will give the AND of address 14 and the address 13, that is the $15^{th}$ and the $14^{th}$ bit. If it is 1, then I need to give the keyboard output. If it is 0, then I need to give the RAM output. So this is a RAM 16 K that we have used. So that is about the read operation. Right? So we have seen how so 1 of the things that we need to keep in mind is during the read operation, irrespective of whether I am going to read from the keyboard

or from the RAM, the RAM is accessed. That 14 bits go to the RAM, the RAM generates an output and that output is available.

Depending on whether I am going to read from thekeyboard or RAM, that output is utilised. Similarly, the keyboard output, of course it is there, it will be there. So but importantly note that the RAM is accessed even though even if I want to access the keyboard RAM is accessed, that output will be there but we do not select that out. Right? But when we are designing say microprocessors for say mobile phones et cetera where power becomes a very important consideration, in those cases, we cannot just keep accessing the RAM. If it is not necessary, we should not access the RAM.

So whenever there is a need, we access the RAM. So we can disable access to the RAM. So the RAM will be in some refresh mode. So it will be consuming less power. Right? So though this design will functionally work, for our processor, it is going to work but advanced processors specifically for mobile phone, et cetera, we will not be accessing a device or any part of the thing, we will not be energising it, we will not make it work, will not make it output, we will not make it function, so whatever word you call for it unless there is a need.

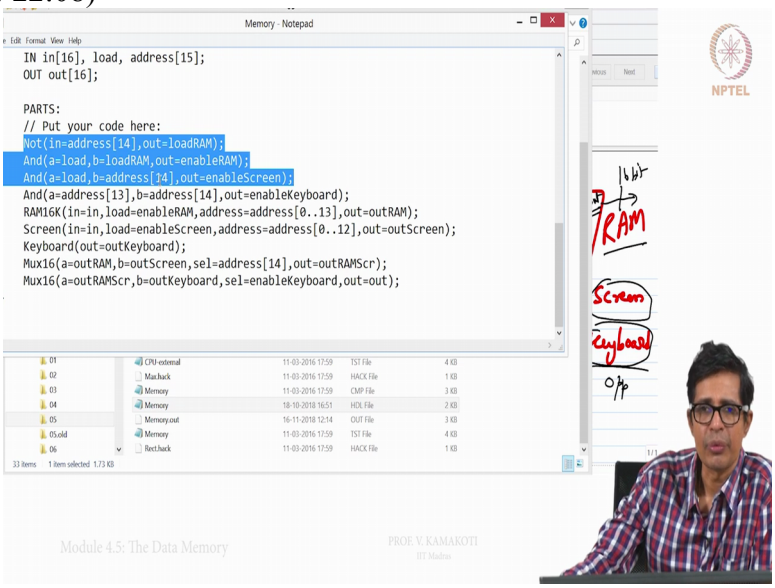
Please note that in our case, we are doing it, that we are making the RAM function even though we may not need the output. However, this is how we designed this. So what we take from this module is of course the most important part of what we call as the memory mapped IO wherein I have 3 devices which are part of the data memory. I read and write from that devices. I collect data from these devices. So we have our RAM, we have a screen and we have a keyboard. All of this, we access as if we are accessing memory and so this is what we call as a memory mapped IO.

So the data is basically transferred to this memory into this memory and out of this memory through a bus. So the bus has a resolution logic which basically finds out if I am going to write, whether I am going to write to the RAM or screen. That logic also we coded and when we are reading, whether I am reading from the RAM or the keyboard and that logic also we coded. Basically the bus essentially has certain multiplexers which will give us the desired output from this memory module.

So and then we also see that this is an example of how you know the even modern day peripherals work. So the screen which is treated as a memory, there is a screen controller like a graphics card as I mentioned which would be displaying whatever you are writing on to this, the corresponding thing on the screen. 0 means white, 1 means black in this case and similarly when you type something on the keyboard, the keyboard controller is responsible for bringing whatever you typed on the keyboard onto this location which we can read.

So this is how the read operation works, the memory mapped I/O works and this works. So now we will go and code this and see how the code for this is done.

(Refer Slide Time: 22:08)

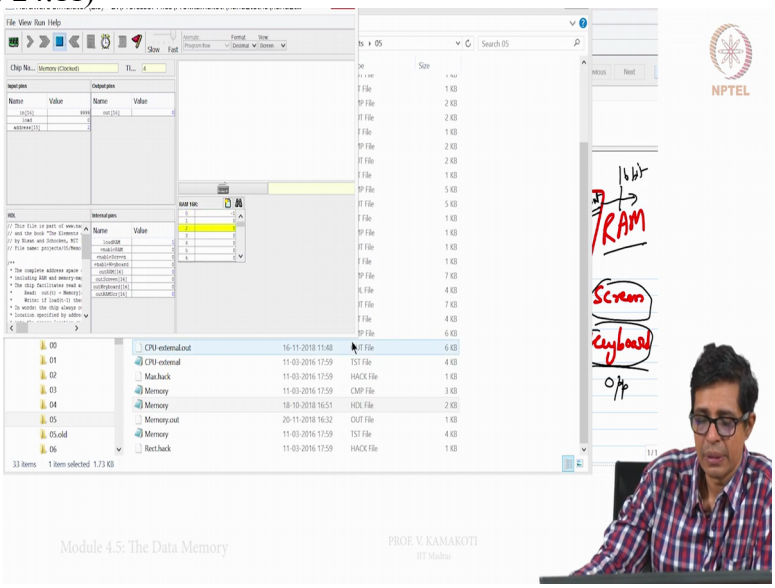



So we will go and load the memory from project 5. The memory is loaded. Let us see how what is there in that memory very quickly. So note that there is a RAM, there is a screen, there is a keyboard right? This Mux is basically used for you know selecting between keyboard and the RAM which would be the output depending on the address and these are basically used for whether we would like to for the write operation. So whether I need to write into the RAM or whether write into the screen. Right?

So I have enabled RAM or enabled screen. Depending on whether I want to actually write, that is given by this input load signal. The load signal is 1 means I want to actually write and this enabled RAM signal says I want to write into the RAM. Enable screen tells that I want to write it

to the screen. So that control logic is basically coded. Other than that, we have instant instantiated a RAM 16 kbits we have coded already as a part of the previous projects. Screen and keyboard are basically built in chips that are made available. Right?

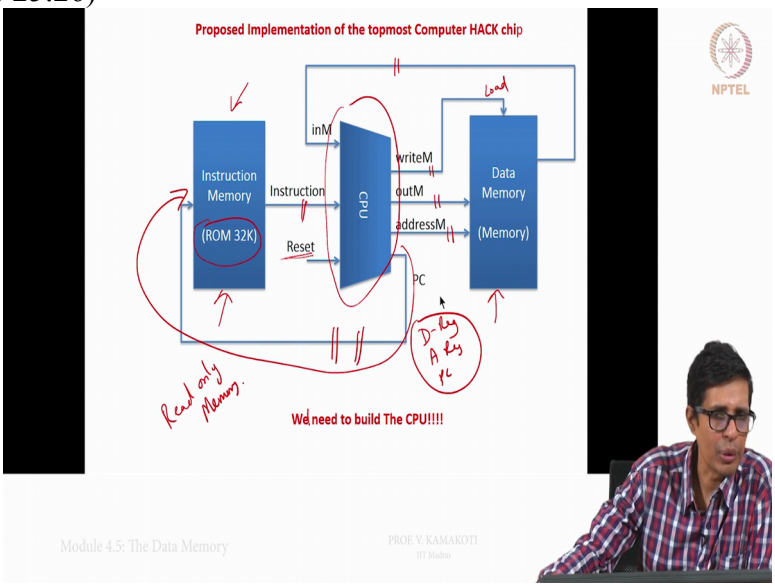
(Refer Slide Time: 24:11)




So now we will run this here for the memory. That is 1 memory test. We will load the script and we will run this memory test. Now we may, I have to quit the keyboard icon in this case. We are in that view screen mode. Click the keyboard icon and hold the K key uppercase. Yes. Make sure you see only 2 horizontal lines. There are 2 horizontal lines here. Hold down Y uppercase. Yes. So we have seen this and so this is the end of script. So is basically tests whether the memory is written properly, screen is written properly, et cetera.

Okay? So this has compare so basically these are some 10 lines of code. Exactly 10 lines of code that we need to write for this memory.

(Refer Slide Time: 25:26)




So in the next module, we will see the organization of the computer, we will connect these through, so we have seen how the CPU works, we have seen how the data memory, ROM 32K is an inbuilt chip. Now we will connect these 3 things together and then run certain test cases and that we will see in the coming module. Thank you.