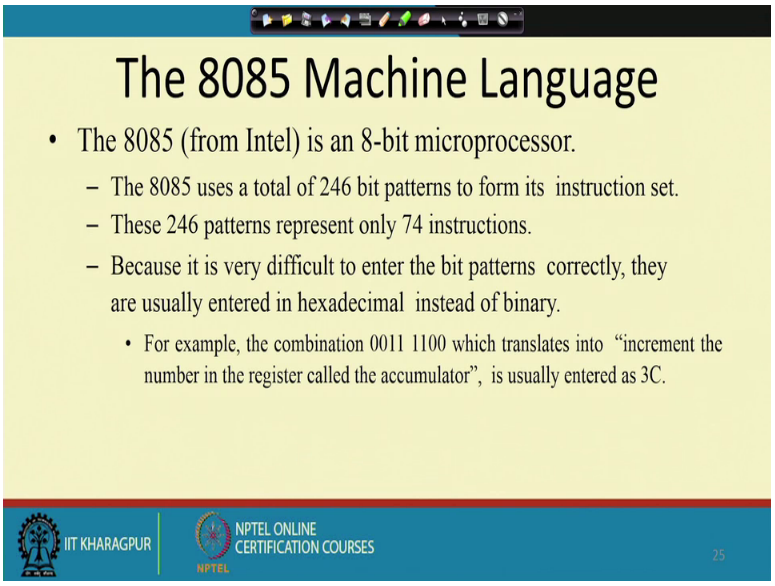
Microprocessors and Microcontrollers

Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 09
8085 Microprocessors (Contd.)

(Refer Slide Time: 00:18)



The 8085 Machine Language

- The 8085 (from Intel) is an 8-bit microprocessor.
  - The 8085 uses a total of 246 bit patterns to form its instruction set.
  - These 246 patterns represent only 74 instructions.
  - Because it is very difficult to enter the bit patterns correctly, they are usually entered in hexadecimal instead of binary.
    - For example, the combination 0011 1100 which translates into "increment the number in the register called the accumulator", is usually entered as 3C.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 25

For 8085 processors, if we try to look into the machine language so, it is an 8-bit microprocessor as we have said previously. So, this is basically so, the word side is 8 bit. So, that number of different combinations that it can understand these 2 power 8 that is 256. So, you can have at most 256 different instructions with eight-bit word size. Out of that 8085 uses only 246 different bit patterns. So, remaining 10 patterns are not used at all to form it is instruction set, and these 246 patterns. So, they define all together at 74 instructions.

So, when I say 74 instruction like add is an instruction, subtract is an instruction, move is an instruction, load is an instruction, like that. And depending upon the operand of the instruction the instructions may change. Like if I have got say 10 different registers in my processor, say R 1 to R 10. So, add R 1 comma R 2 so that the R 1 gets the content of R 1 and R some of R 1 and R 2 is different from the instruction at R 3 comma R 4, where 3 gets the content of R 3 and R 4, sum of the sum of R 3 and R 4.

So, that way these 2 are taken as different instructions. So, if we consider that way then there can be 246 different patterns, and that correspond to actually 74 different instructions. And it is very difficult to enter the bit patterns correctly. So, they are usually entered in a hexadecimal format instead of binary format. Like suppose in if you look into this microprocessor kits that are available for the development kits that are available.

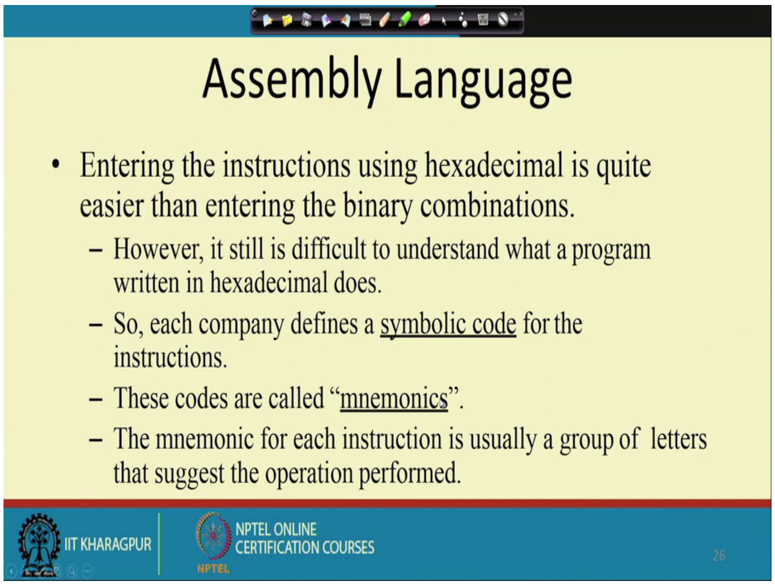
So, they are normally there are 2 ways in which you can load your program into that kit, one is by means of having some upload facility from some computer. So, that way the values may be uploaded directly into the ram of that processor of that kit. Or many times the kits they have got a hexadecimal hexadecimal keyboard on which is the. Numbers 0 to f they are given. And you can type in the values that you want to entered like, and they are entered in a hexadecimal format.

For example, if you are if you consider this particular bit combination like 0 0 1 1 1 1 0 0. So, this is this is actually the instruction that corresponds to the, that is the actually corresponding to the instruction which tells the processor to increment the number in the register called accumulator. There is a special register called accumulator in 8085. And this particular bit combination 0 0 1 1 1 1 0 0, if it is executed. So, it is telling the processor to implement the content of that accumulator by 1.

So, instead of entering 0 0 1 1 1 1 0 0 as a binary bit pattern; so if you are entering the value through hexadecimal keypad, then the value 3 and c they can be entered and the essentially translates to that. Of course, if you are doing it where are some computer, then it will happen automatically. So, automatically the values will be loaded there, but all these uploading tools. So, they have got a got an option which is to list the content of this uploaded file.

So, there it will show in the hexadecimal format only for the sake of compactness. And all the manuals and everywhere you will find that they are talking in terms of the hexadecimal values and not in terms of binary values. Because of the very simple reason that binary values remembering them or instructing them becomes difficult. And you see there are only 70, 246 different combinations. So, in older days people almost used to memorize like, what are those what are the codes for all these instructions. It was not that difficult like once a person is doing the 8085 programming for some days, it often becomes almost common to remember the instructions that are used very frequently the codes for them. So, that way it is it; that was the way it was done.

(Refer Slide Time: 04:44)



Assembly Language

- Entering the instructions using hexadecimal is quite easier than entering the binary combinations.
  - However, it still is difficult to understand what a program written in hexadecimal does.
  - So, each company defines a symbolic code for the instructions.
  - These codes are called "mnemonics".
  - The mnemonic for each instruction is usually a group of letters that suggest the operation performed.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

26

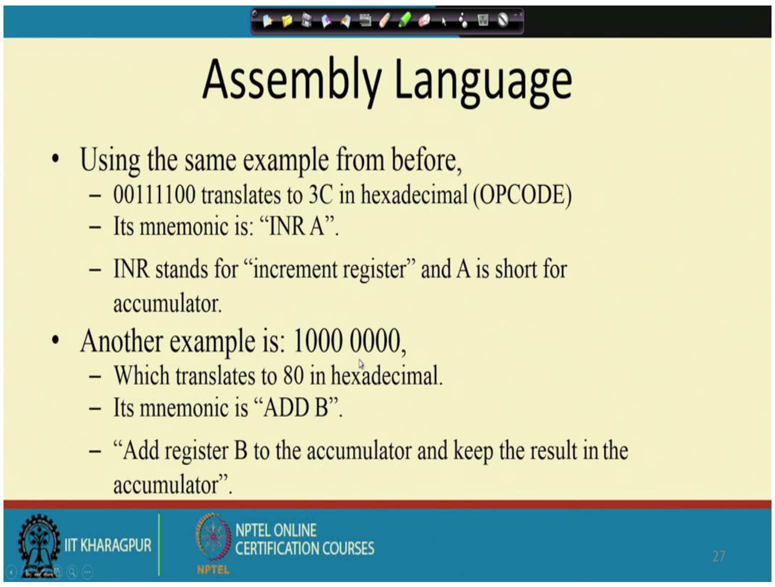
Now, while we are writing programs. So, one possibility is that we write the program in say high level language like c or c plus plus, and then we have got a translator that will translate it into the machine language of the processor. Now the difficulty is that there are many scope of unoptimized translation. So, that the translation which are not very optimized. So, and as you know that the human being so they are the best optimizers.

So, if the a code fragment is reasonably small, and if you are an experienced programmer, then possibly you can write the best possible program in machine language or a so, that you can write a program in machine language which will be better than any other program which is written in c or c plus plus. Because you will be able to calculate like the time required for executing each and every instruction and things like that as we will see. So, that way it becomes easier or it becomes better if you write in a machine language.

But again, writing in machine language the difficulty is that code and all that. So, what is done an intermediary to these 2 the high level and the machine language is the assembly language? So, instead of entering instructions is a hexadecimal. So, you can you can do it using some symbolic codes. So, it is so, and also if the if you are if you if you are given a set of hexadecimal values. So, it is difficult to understand the meaning of the program. So, that is the other issue.

So, what is happening is that. So, the companies they have defined some symbolic code for the instructions. So, these coals codes are called mnemonics. A mnemonic for each instruction is usually group of letters that suggest the operation to be performed. So, the based on the move is a mnemonic add is a mnemonic. So, that way it is telling the operation that we want to perform; so this assembly language program or assembly language statements. So, they will consist of the mnemonic, and the operand that will be there or on which that operation will be done.

(Refer Slide Time: 07:05)



Assembly Language

- Using the same example from before,
  - 00111100 translates to 3C in hexadecimal (OPCODE)
  - Its mnemonic is: "INR A".
  - INR stands for "increment register" and A is short for accumulator.
- Another example is: 1000 0000,
  - Which translates to 80 in hexadecimal.
  - Its mnemonic is "ADD B".
  - "Add register B to the accumulator and keep the result in the accumulator".

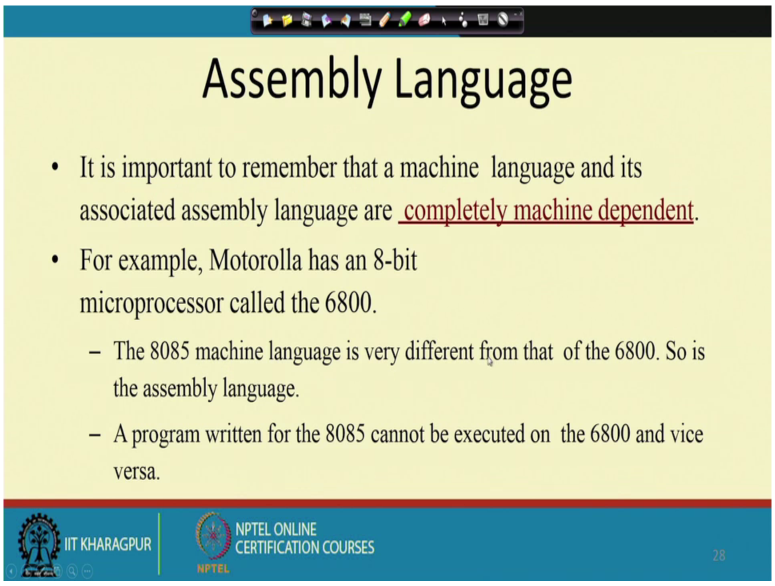
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
27

So, if we go back to the previous example. Like, this particular bit pattern 0 0 1 1 1 1 0 0. So, this is 3 c in hexadecimal. So, and it is corresponding mnemonic is INR A. So, INR is the increment register. So, I N is increment an R it registers increment register A. So, A is the is a special register that we have in 8085, which is the accumulator. So, increment register A; and this 3-c value. So, this is called the operation code. That is, it will tell the processor, what is the operation to be done, and that is why it is called opcode or operation code and this A. So, this is called the operand, and this this INR is the opcode.

Now so, for example, if you take say this instruction 1 0 this particular bit sequence, 1 0 0 0 0 0 0 0. So, this is 80 in hexadecimal. So, 80 is the mnemonic. So, 80 is the opcode you can say, and it means that corresponding mnemonic is ADD B. So, ADD B is the mnemonic, 80 is the opcode.

So, it says that add register B, to what? So, it does not tell anything like that in this instruction, but implicitly it means that this B value there A value of register B, will be added to the value of register A. So, A is a special register, which is called accumulator registered in 8085, and whenever you have got any arithmetic logic operation, then one of the operand in most of the cases is the accumulator, and also that destination is the accumulator only. So, when you execute the instruction at B, what it will do it will add the content of register B with the content of register A, and the value will be stored in register A. So, that is the final value will be kept it registered a only or the accumulator only.

(Refer Slide Time: 09:10)



Assembly Language

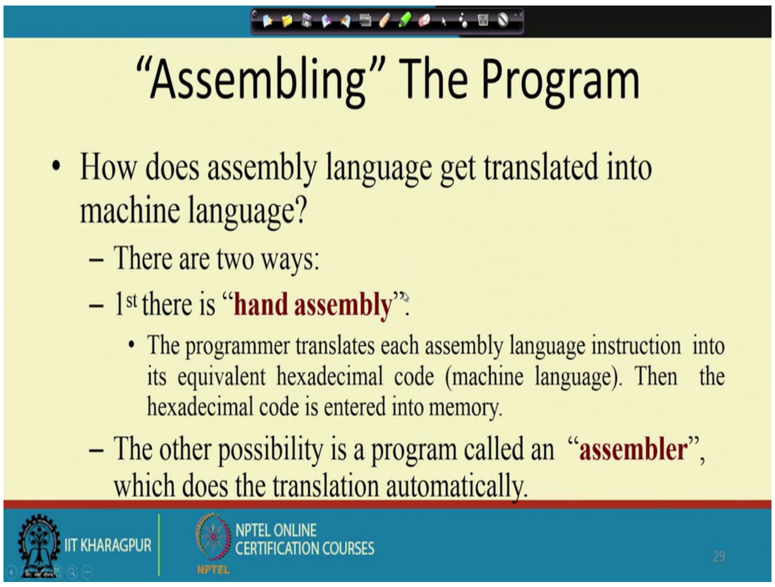
- It is important to remember that a machine language and its associated assembly language are completely machine dependent.
- For example, Motorolla has an 8-bit microprocessor called the 6800.
  - The 8085 machine language is very different from that of the 6800. So is the assembly language.
  - A program written for the 8085 cannot be executed on the 6800 and vice versa.

So, it is important to remember that a machine language, and it is associated assembly language are completely machine dependent. So, like when you are talking about the c language, a high-level language like c that is machine independent, you can take the same c program to a number of platforms where the underlying processors are different, and then you can again compile it there and get it compile program executed. But you cannot take the same compiled code from one machine to another machine, until and unless the environments in both the system both the computers are exactly same, including the processor that we have and particularly the processor that is exactly same.

Otherwise the machine language of one processor is different from the machine language of another, and assembly language is also different. The assembly language is nothing but a user stand able version of the machine language; so human understandable version of the machine language. So, it is going to be different for different processors. So, one program written in assembly language of processor a will not work on processor b.

So, this is a typical example like Motorola it has got an 8-bit microprocessor called 6800 and 8085 machine language is different from that or 6800. So, is the assembly language a program written for the 8285 cannot be executed on 6800 and vice versa. So, that is obvious.

(Refer Slide Time: 10:45)



"Assembling" The Program

- How does assembly language get translated into machine language?
  - There are two ways:
  - 1st there is "**hand assembly**".
    - The programmer translates each assembly language instruction into its equivalent hexadecimal code (machine language). Then the hexadecimal code is entered into memory.
  - The other possibility is a program called an "**assembler**", which does the translation automatically.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

29

So, for an for an assembly language program. So, see assembly language is also not understandable by the processor; so for assembly language program so, we need to convert it into machine language.

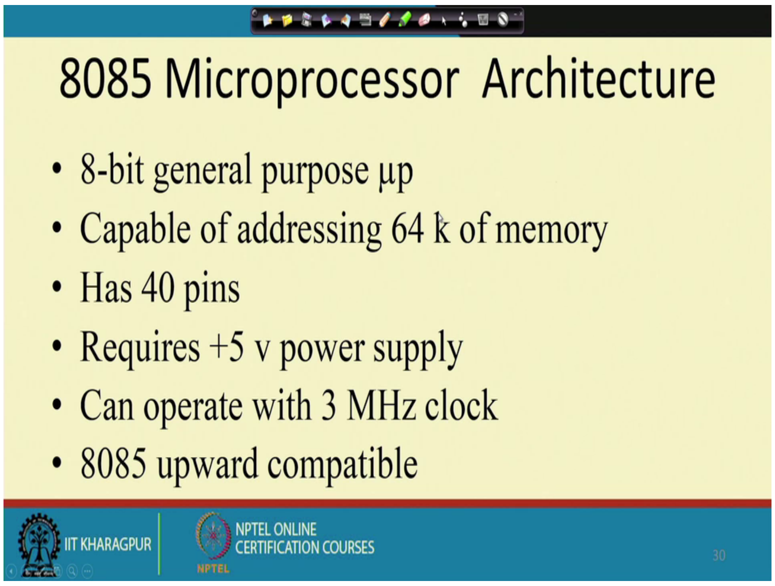
So, how do we do this? So, there can be 2 different ways by which it can be done. So, previously it used to be done by something called hand assembly. So, as I was telling so, older designers of 8085 based systems; so they remembered the code off for each and every instruction and also, if you look into the manual pages of 8085. So, that will tell you like what are the codes for various instruction. So, first you write down the program in assembly language, and then look into that chart where it has got for every instruction the corresponding machine the corresponding machines of instruction format or the opcode part opcod opcode forms. And using those opcode so, you can convert your program or it is in 8085 assembly language to the machine language program. So, that is not very difficult. So, this is done by the hand assembly this process is known as hand assembly.

So, the programmer translates each assembly language instruction into it is equivalent hexadecimal code or machine language, and then the hexadecimal code is entered into the memory. So, this is the way the hand assembly used to be done and naturally this is error prone like while doing this operation. So, I may be doing some mistake in selecting the in looking into the proper mnemonic and all, that and there may be a mistake and a possibility of error is more. And then those, entering that hexadecimal program into the computer system that becomes another headache so that way it is becoming it is error prone.

So, the other possibility is that a program if they use a program called assembler. So, this assembler programs just like the compiler. So, compiler they translate a high-level language program to machine language. And these assemblers they can convert the assembly language program to machine language. In fact, if you look into any compiler so, they will be having one stage called assembler. So, no compiler translates a program directly to machine language, it converts it into the assembler. And from the assembler the; it will be converted into machine language.

The reason is very simple like in a computer system if I have got if I support a number of high level languages, and then individual compilers they can convert to the assembly level language program for that for that processor. And in the system, I can have a single assembler, and that single assembler can now convert all these assembly language programs into machine language. So, there is no point in duplicating the effort of putting the assembler as a part of all these compilers. So, assembler part is made separate, and this compiler so, they generate code up to this assembly level up to this assembly level and from that point onwards assembler takes up and does the thing. So, this way we can have with there are assemblers that are available there are many assemblers, they can find they are available and they are used in the assembly process.

(Refer Slide Time: 14:16)



8085 Microprocessor Architecture

- 8-bit general purpose µp
- Capable of addressing 64 k of memory
- Has 40 pins
- Requires +5 v power supply
- Can operate with 3 MHz clock
- 8085 upward compatible

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
30

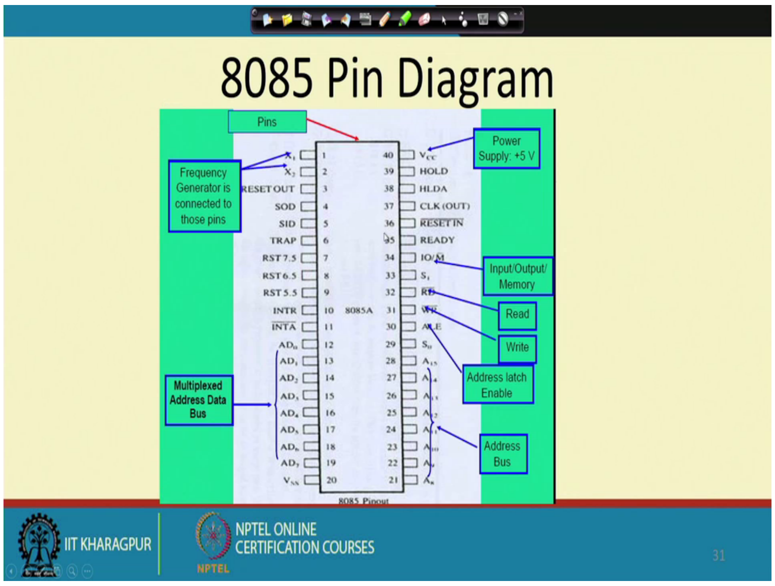
So, next we will go into that these are 8085 microprocessor architecture. So, it is A 8-bit general purpose microprocessor. So, let us try to understand the meaning 8 bit means. So, it can it does 8-bit operations at a time. So, most of the operations that it does is on 8-bit data, and even if it we operates on 16 bit. So, it will be handling them as 8-bit data only it is a general purpose. So, it is not a special purpose thing.

For example, if you are looking into a say digital signal processor. So, that is dedicated for signal processing application. So, it is not like that. So, it is a general-purpose processor and it is a microprocessor. So, it has got the CPU does it? It is actually a CPU consisting consisting a ALU, then you have got registers and all that. But it does not have memory and all that. With it is capable of addressing 64 kilo of memory. So, and each is and it is 8 bit so, whenever it is accessing memory. So, it will be in terms of 8 bits. So, that way you can say it is 64 kilo byte of memory can be addressed by the processor.

Now, whether the system will have 64 kilobyte or memory or not that depends on the designer of the system that uses is 8085 as the basic processor. Now if it is not using 64 kilobyte maybe the system has got only 32 kilobyte of memory in it. So, remaining spaces will be empty, and processor should not generate addresses in that range during execution of any program in that system.

Anyway so, the basic processor it has got the capability to address 64 kilobyte of memory. Physically it is a forty-pin chip. So, there are forty pins. And so, this is detailed pin layouts we will see, 5-volt power supply is required can operate at a clock frequency of 3 megahertz, and it is upward compatible. That is the previous version 8080 it is compatible with that in terms of pin layouts.

(Refer Slide Time: 16:24)



So, this diagram is actually a telling as the pin diagram of 8085. So, we have got this Vcc which is the power supply pin then we have got this x 1 and x 2. So, in which we have to connect a crystal externally to generate the clock signal. So, this is the frequency generator. So, this will be connected to this x 1 and x 2. Then we have got some special lines. So, some of the special lines that we have is as I said that it can address 64 kilobyte of memory. So, if it is accessing 64 kilobyte of memory, then I need 16 address bit lines to power 16.

So, you see these lines marked pin number 21 to 28 marked as 8 to A 15. So, these are the 8 bits. And this pin number 19 to this pin number 12. So, they are actually the other bits. So, this is AD 0 AD 1 up to AD 7. So, they are they are multiplexed address data was. So, will understand this multiplex term later, but this this will also provide the address line. So, address line the address bus is total 16 bit out of out of that this side I have got 8 bit, and this side I have got the 8 bit.

V ss is the ground, and then they are so, next after address bus. So, we have got the data bus. So, data bus is also given by these lines. So, AD 0 to AD 7. So, if you just ignore this later A. So, what you get is D 0 to D 7. So, this is actually the data bus. So, address and data they have been. So, they are mapped onto same pins. So, address 0 and data 0 they are put on the pin number 12 address 1 and data 1, they are put on pin number 13 etcetera. But the data actually reduces the size of the chip, because you cannot have any arbitrary sized pin package.

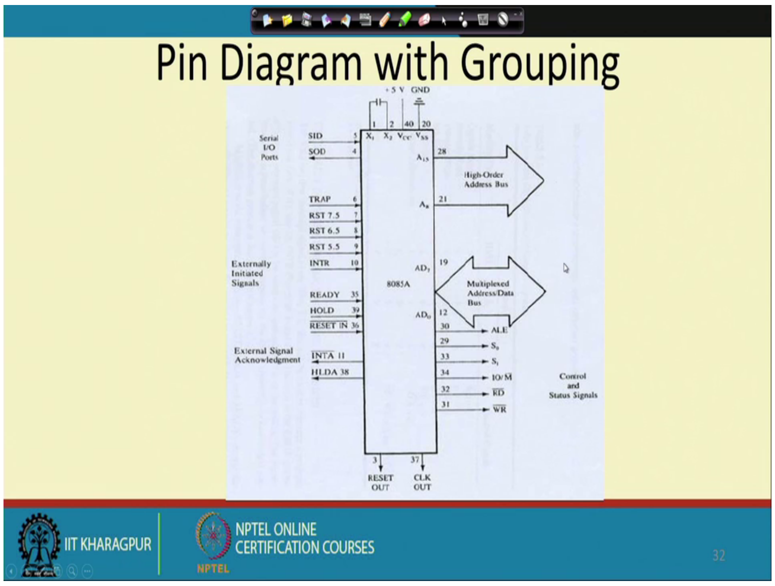
So, after so, this forty-pin package is a standard package; so after that the number of pins next one I think is 64. So, that way in between you do not have any other option. So, it is mandatory, that we keep the pin size compatible with the packaging. So, that is there now apart from this. So, there are some special signals like this, read bar and write bar which is for the read and write operations. So, they are to be connected. Then there is a there is there is there are a set of interrupt lines. So, if you want to interrupt the basic operation of this processor from the outside word you can use this line. So, trap rst 7.5, 6.5, 5.5 intr. So, these lines can be utilized to telling the processor, that something exceptional has happened in the outside word and it should take care of that.

Some for example, maybe this this line say this rst 7.5 lines is connected to a mouse, and whenever this mouse is clicked. So, it tells the processor that the user has please dip the mouse button once. So, that way it can the the microprocessor may do some special action corresponding to that.

So, we will come to that those things later. Then apart from that we have got this this ALE signal, this ALE signal is address latch enable. So, that will tell us like when these that will be useful for separating out this multiplexed address and data bus lines will see it slowly. Then there is another line which is I O stroke M bar; so I O M bar. So, it is input output or memory. So, when it is when this processor is connected to some memory device, it can also be connected in a similar fashion to some I O device.

Now, where it is processor is accessing the accessing outside word it is accessing for outside word for some data. So, whether the data is from memory or from I O device. So, if it is from I O device in that case this I O M bar line; so this value will become 1 and if it is from the memory then this I O M bar line will be equal to 0. So, accordingly we can design some decoder circuitry by which either the memory chip will get enabled or the I O devices they will get enabled; so they will. So, we will discuss slowly on all these concepts.

(Refer Slide Time: 21:06)



Pin Diagram with Grouping

So, this diagram gives a cleaner way of understanding like what are the things that we have. So, we have got this higher order address bus A 8 to A 15 in pins 21 to 28. Address and data bus so, they are multiplexed. So, this AD 0 through AD 7. So, they are multiplexed here (Refer Time: 21:27) this; so, this address and data bus. And we have got this so, your crystal is connected here between pin 1 and 2 pin forty is the plus supply voltage 5 plus 5-volt pin 20 is the V ss which is the ground line.

Then you can have some interrupts. So, like given by the pin number 6 7 8 9 and 10. So, they are the interrupt pins. Then you can have some interrupt acknowledge. So, this is this inta bar line. So, this is pin number 11. So, this is if this comes in conjunction this pin number 10 intr; so this intr so this when this interrupt is coming. So, this interrupt acknowledgment pin is activated by the processor. So, when we discuss when we will discuss on the interrupt. So, we will be talking about this interpreter of acknowledge pins.

Now so, in my board; so apart from the processor or the microprocessor there can be many other devices that also need a clock signal; so this clock out pin. So, this is actually giving the clock signal of the processor to the other components on the board. So, if you need any device to be driven by the same clock as the microprocessor. So, you can use this clock outline for that purpose. Similarly, there is a reset pin. So, this reset in bar so, if this reset in bar. So, this line is made 0; that means, is 8085 operation will be reset and as a result. So, processor will start operation from the beginning.

So, such some registers in the processor they will get some special values, that in some sense will mean that the processor is restarting the operation. And some and as similar to the clock this reset also we may need to give a reset out because when this reset pin is activated maybe we may have to reset many other devices as well. So, that is the reset-out pin. So, this reset out will be quick at this signal can be used to reset other devices. And we have got this SID and SOD; so these 2 lines. So, this is SID stands for serial input data, and a SOD stands for serial output data.

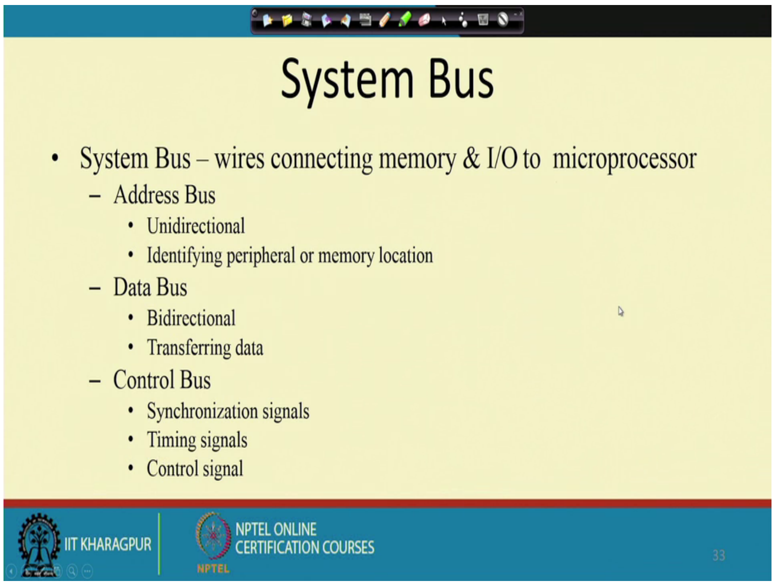
So, if you are feeding values to this microprocessor from the outside world. So, one possibility is that you put some I O device connected through this address bus data bus lines. That is one of one way of doing it. The other way of doing it is you have some serial you have some mechanism by which you can send the bits serially, and they are connected to be SID line. So, the bits are coming serially similarly this can serially output the data through the SOD line. So, if you are satisfied with low speed data transfer, you can go for the serial communication via this SID and SOD line.

Another important pins that we have this read bar and write bar. So, the this read bar and write bar. So, they are actually the read control and write control. So, this will tell whether the processor is trying to read from the outside world or it is trying to write to the outside world. So, when it is say really trying to read from the memory, then this read bar line should be made 0 that is it is active. So, it is read bar. So, 0 means it is active. So, this read bar signal is activated. And if it is trying to write something onto the memory in that case it is the write bar signal will be activated.

So, instead of memory it will be I O device also, and whether it is I O device or the memory. So, that is identified by this I O M bar line this I O M bar line. As I have said if this line is equal to 1; that means, the processor is expecting the value to come from some I O device, and if this value is 0, then it is expecting the value to come from the memory. Then these 2 pins S 0 and S 1. So, they actually tell the stage at which the microprocessor is at this point of time.

So, as you know that this processor executes the instruction by following this is fate decode execute cycle. So, in those phases this S 0 S 1 pin. So, they will have some appropriate values. So, by looking into these values for adding from the external world; so you can understand what the processor is doing now. So, this S 0 S 1 and this I O M bar read bar right bar this signal so, they tell about the status of the processor. And they are in general known as the control and status signals, because that talked about the status of the microprocessor.

(Refer Slide Time: 26:34)



System Bus

- System Bus – wires connecting memory & I/O to microprocessor
  - Address Bus
    - Unidirectional
    - Identifying peripheral or memory location
  - Data Bus
    - Bidirectional
    - Transferring data
  - Control Bus
    - Synchronization signals
    - Timing signals
    - Control signal

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
33

So, if we are looking into the wires that we have in the system. So, that gives rise to something called system bus. So, this is the wires that connect the memory and the I O devices to the microprocessor. So, address bus is unidirectional, and it identifies peripheral and memory locations. So, if we look into this address bus. So, this address bus you can see the A 8 to A 15 it is going out from the processor. And similarly, here also this when this multiplexed address data bus acts as address bus, then it is going outside going out of the processor.

So, this address bus is unidirectional, and it is going to the outside the output from the processor. And it is used for identifying the memory location from where the value has to be read or where the value has to be written. Or the peripheral device that has to be accessed for doing the operation. Then data bus the data bus is bi directional because you may need to transfer the value from the microprocessor to some memory location or I O device, or you may have to get the value from the memory location or I O device to the processor.

So, as a result this data bus is going to be bi directional in nature. And there is control bus; so they are for synchronization of synchronization signal, timing signal, control signals etcetera. So they are, actually constituting the control bus. So, the system bus it is consisting of all those wires, which constitute the address bus, data bus and control bus.