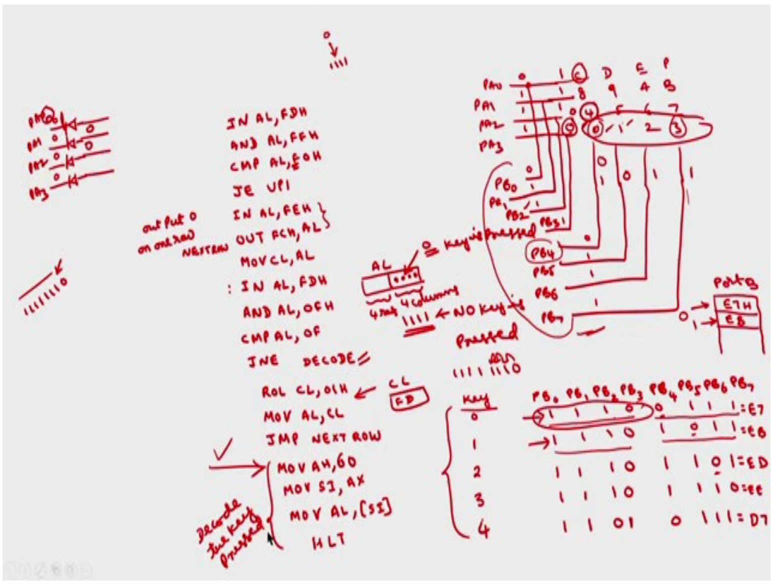
Microprocessors and Interfacing
Prof. Shaik Rafi Ahmed
Department of Electronics and Electrical Engineering
Indian Institute of Technology, Guwahati

Lecture – 20
7-segment Display Interface

So okay, in the last class we are discussing about the keyboard interface to 8086. So, in that we have discussed about the first two steps which is detect debounce and the third one is decode.

(Refer Slide Time: 00:42)



```
          IN AL,FDH
          AND AL,FFH
          CMP AL,FOH
          JE UPI
          IN AL,FEH
NEXTROW   OUT FCH,AL
          MOV CL,AL
        : IN AL,FDH
          AND AL,OFH
          CMP AL,OF
          JNE DECODE
          ROL CL,01H
          MOV AL,CL
          JMP NEXT ROW
          MOV AH,60
          MOV SI,AX
          MOV AL,[SI]
          HLT
```

Key	PB0	PB1	PB2	PB3	PB4	PB5	PB6	PB7	
0	1	1	1	0	0	1	1	1	= E7
1	1	1	1	0	1	0	1	1	= EB
2	1	1	1	0	1	1	0	1	= ED
3	1	1	1	0	1	1	1	0	= EE
4	1	1	0	1	0	1	1	1	= D7

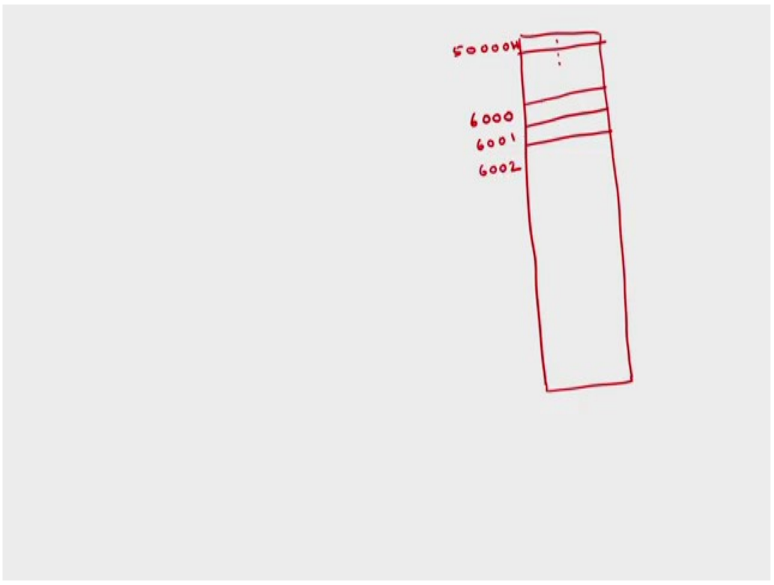
So, here we know that the rows are connected to PA 0, PA 1, PA 2, PA 3 and columns are connected to PB 0 to PB 7. So, in that the same row information is taken at PB naught to PB 3 and the column information is taken at PB 4 to PB 7 ok. So, the codes corresponding to key press 0 is so, when a 0 key is pressed in order to detect that key 0 key. So, when 0 key is pressed PA 0 to PA 3 will contain 1 1 1 0, but this is connected to here. So, PA 0 is PB 0; so, PB 0 is 1 1 only. So, for 0 press this key will be 1 1 1 0 because this 0 key is present in this row and column.

What is hexadecimal equivalent of this one? So, if I receive this 0 1 1 here means there is a 0 key press. So, the code will be E 7 corresponding to 1 this will be E B okay. So corresponding to, so 2 is here. So, this should be 1 1 1 0 only for 2 also and 2 is here means this should be 0, 2 is in PB 6; PB 6 should be 0. So, 1 1 0 1 so, this will be D 13 is

D, E D. So, corresponding to 4 this will be same 1 1 1 0 corresponding to all these four keys this code is 1 1 1 0 only this will changes here this 0 will be shifted.

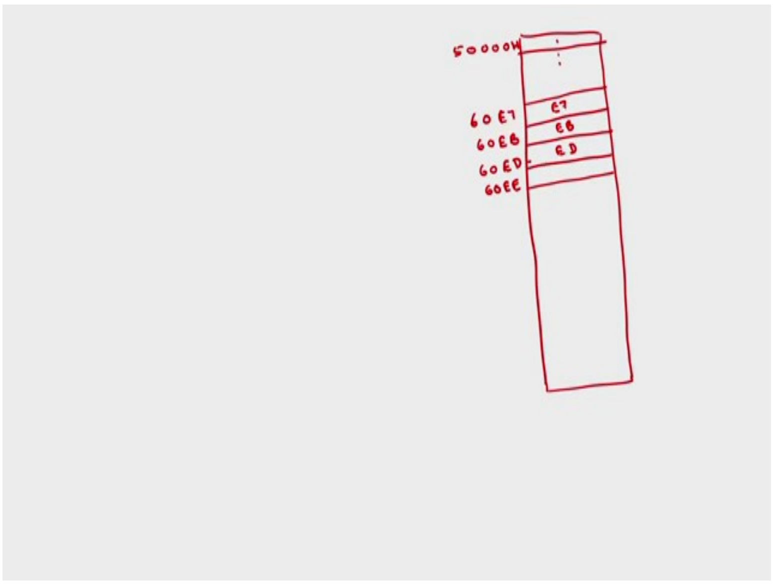
So, here 0 was here initially now here now here now this will be at last 1 shifted towards the right by 1-bit position 1 1 1 0 P 7 is this 3 is connected in P 7 line ok. So, this will be E E. Similarly, we can obtain the codes for 4 5 6 7 and so on. So, what I am going to do is here I am going to store these values in some location.

(Refer Slide Time: 03:19)



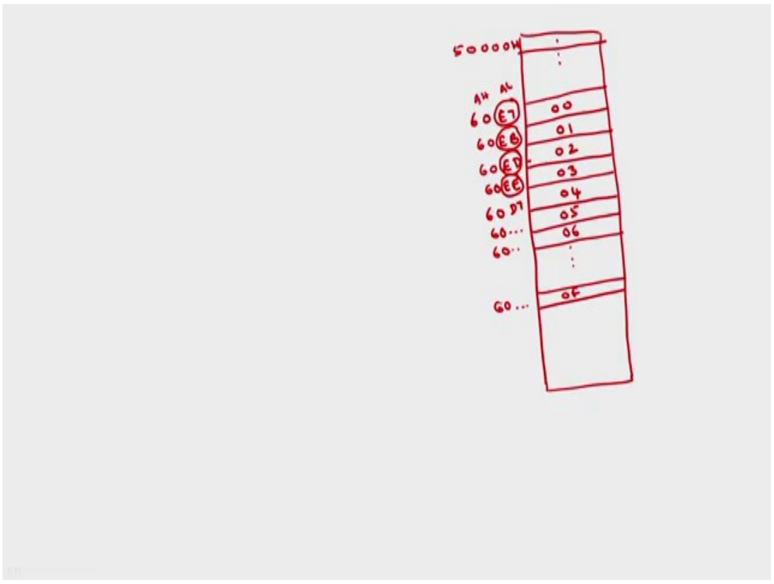
So, I want to store some location say 6000 H in data segment. The starting of this data segment is 50000 H, 60001, 6002. So, instead of this I will take this 0 0 and 0 1.

(Refer Slide Time: 03:52)



This last 8-bits, I will take as the code corresponding to 0, 1, 2 and so on. So, the codes are E7, EB, ED, EE, E7, EB, ED, EE. Here I will simply store 0; this is code corresponding to 0.

(Refer Slide Time: 04:32)



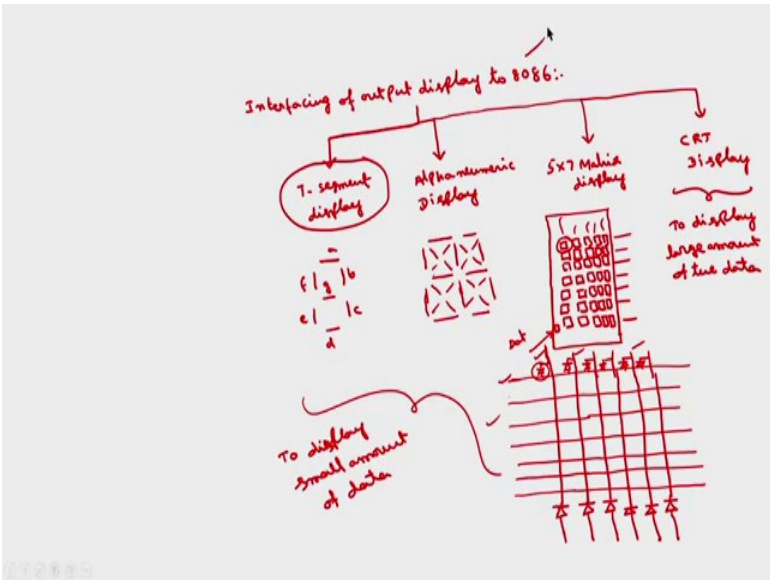
So, I will store 00, 01, 02, 03 because these are the codes corresponding to; these are the code corresponding to 0, this is the code corresponding to 1, this is the code corresponding to 2, this is the code corresponding to 3. So, I will store this data here finally, I want to read this data into accumulator AL. Similarly, 04 60 the code corresponding to 4; the code corresponding to 4 will be so, for 4 this sequence will be this PB 2 should be 0, remaining all should be 1s for 3 this is 3 for 4.

So, 4 is here. So, this bit should be 0, remaining all will be 1s. So, this bit is connected to PB PB 2; PB 2 means this is 0 0 1 0 0 1 and correspondingly 4 is there in this row. So, means a PB 4 should be 0, PB 4 should be 0 remaining all will be 1s. This will be now this is D7. So, we have to store D7; here the location D7 I am going to store 4. Similarly, 60 this will be code corresponding to 5. We can obtain this in a similar manner, this will be a code corresponding to 6 so on up to this will be code corresponding to F 0F. Totally we have 16 locations to read this value into AL.

So, at the end of this program, this AL will contain AL will contain the code corresponding to this 4 rows and 4 columns; this entire code this code will be contained in AL at the end this program. Now, to read that value into AL ok. So, what I will do is MOV AH comma 00 60 because I want to store this in a locations of 60 this value is there in AL, I am initiating to this one. So, that AX will be having this address that I will take into some register.

MOV BX comma or SI comma AX; then MOV AL comma contents of SI because in that particular location, we have the code corresponding to the key that is pressed. So, that I am to going to take into accumulator then halt. So, this process is to decode the program decode the key. This is how we can interface the keyboard to the 8086 micro-processor. Now, the next device that I am going to discuss which is an important output device which is output display; interfacing of output display to 8086.

(Refer Slide Time: 08:06)


Interfacing of output display to 8086:-
7-segment display
Alphanumeric display
5×7 Matrix display
CRT display
To display large amount of the data
To display small amount of data
dot

There are several types of the output displays. The simple output display is 7-segment display. As the name implies there are 7-segments. This is segment a, b, c, d, e, f, g. So, we can display all the digits from 0 to 9 ok. Then suppose if you want to display in addition to the digit digits the alphabets also, then we have to use alphanumeric displays alphanumeric display.

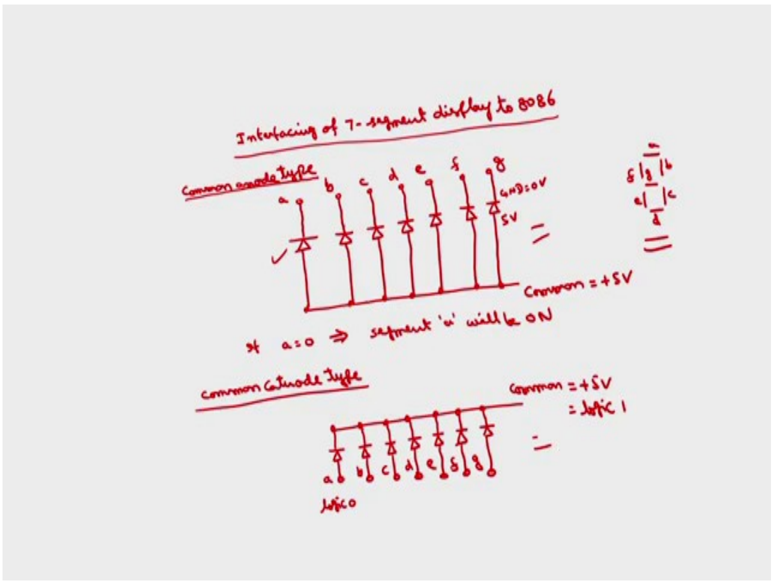
We have a 18 segment displays such as this is the example of alphanumeric display. Using this you can display any alphabets also. There is another type of display which is called matrix display. If it takes say 5 by 7 matrix display. There will be having IC; this segments are arranged in a form of a matrix. So, it will be having 5 by 7 means 5 in a row 7 such rows, we are appropriately selecting the segments we can display any alphabets on this display.

So, totally we have 35 segments 7, 7 rows 1 2 3 4 5 6 7 and 5 columns and there will be one extra segment here to represent the dot. If I take inside this the circuitry will be something like. So, we have 6 rows totally connected through diodes that is light emitting diodes, LED. This will be connected to a port and then we have here along this 7 rows it is intersection, we have another segment.

Here at this we have diode. So, this is corresponding to this particular segment. This can be on or off by properly selecting this rows and columns of this particular LED. Similarly here we have LED. So, there are total 35 LEDs here. So, like that by properly selecting this rows and columns, we can select any of this segments thereby we can display any digit or any alphanumeric letter okay. This is about the 7-segment display.

Another is CRT display – cathode ray tube display. The first three these three are used for to display the small amount of the data to display small amount of the data. We can use either 7-segment display or alphanumeric display or 5 by 7 matrix display. Whereas this last one CRT display most of the computer uses CRT display whereas, the microprocessor kits are some instruments uses this LED displays. So, this I mean a CRT kit to display large amount of the data. First we will discuss how to interface a 7-segment display to the microprocessor 8086.

(Refer Slide Time: 14:36)



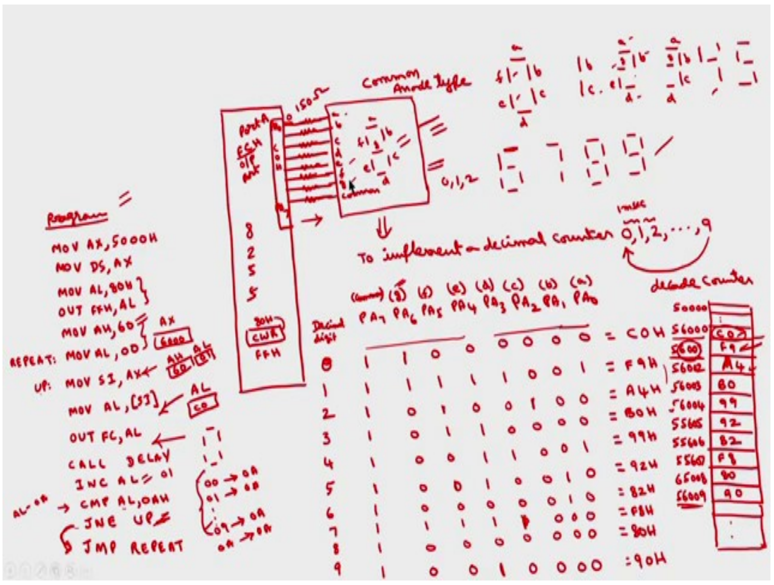
Interfacing of 7-segment display to 8086: There are two types of 7-segment displays – one is called common anode and another is called common cathode. Since we have discuss with the there are total 7 segments corresponding to each segment as in the case of a data matrix we have a LED inside that a b c d e f ok. So, corresponding to a, we have one LED, corresponding to b, corresponding to c, corresponding to d, corresponding e, corresponding to f, corresponding to g. So, each one will be having one anode and one cathode.

So, if it is common anode type as a name implies anode should be common to all the 7 LEDs and this will be connected to a common point. This is called common and each segment is connected to one particular pin this is a, this is b, this is c, this is d, this is e, this is f, this is g. Now, to glow any LED if I want to glow this, anode is connected to common which common has to be connected to plus 5 volts. To glow a LED this anode should be plus 5 volt this should be we have to ground or 0 volts ok.

So, if a is equal to 0 implies the segment a will be lit means ON ok. Similarly, if b is equal to 0, segment b will lit, c is equal to 0 c will lit depends upon the segment that is required we have to lit the corresponding segments, this is common anode type. In common cathode type reverse. The common point is connected to all the cathodes in case of common cathode type all the cathodes, it will be connected together. So, all this cathodes will be connected to common point which has to be grounded.

Then anodes you have to connect to glow a particular segment you have to connect logic 1 or plus 5 volts. This is common point which has to be connected to plus 5 volts or logic 1 and to display any particular segment this has to be connected to logic 0. So, that this a will glows, this is b c d e f g. This is about this a common anode type and common cathode type. So, inside this 7-segment display we have this circuitry, depends upon whether the display is common anode type or common cathode type ok.

(Refer Slide Time: 19:08)



Program

```
         MOV AX,5000H
         MOV DS,AX
         MOV AL,80H
         OUT FFH,AL
         MOV AH,60
REPEAT:  MOV AL,00
UP:      MOV SI,AX
         MOV AL,[SI]
         OUT FC,AL
         CALL DELAY
         INC AL
         CMP AL,0AH
         JNE UP
         JMP REPEAT
```

Decimal digit	PA7 (common)	PA6 (g)	PA5 (f)	PA4 (e)	PA3 (d)	PA2 (c)	PA1 (b)	PA0 (a)	
0	1	1	0	0	0	0	0	0	= C0H
1	1	1	1	1	1	0	0	1	= F9H
2	1	0	1	0	0	1	0	0	= A4H
3	1	0	1	1	0	0	0	0	= B0H
4	1	0	0	1	1	0	0	1	= 99H
5	1	0	0	1	0	0	1	0	= 92H
6	1	0	0	0	0	0	1	0	= 82H
7	1	1	1	1	1	0	0	0	= F8H
8	1	0	0	0	0	0	0	0	= 80H
9	1	0	0	1	0	0	0	0	= 90H

50000	
56000	C0
5600	F9
56002	A4
56003	B0
56004	99
55605	92
55606	82
55607	F8
65008	80
56009	90

Now, one way to interface this 7-segment display to 8086 is so, I will simply connect to one of the ports of 8 2 5 5. So, I will take one port which will be having 8 bits, this is 8255. So, I am not showing the connection with 8086 that we have already discussed. So, I will take one complete port of this one; say port A with a address of FCH and control word register is having address of FFH. This is PA 0, this is PA 1, PA 2, PA 3, PA 4, PA 5, PA 6, PA 7; this is PA 0 to PA 7. This I will connect to 7-segment display.

Of course, you require some current limiting resistors here because the current capacity of this segment is around some 20 milli amps whereas the microprocessor current is more. So, we have to use the current limiting resistors of the order of 150 ohms each. This a 7-segment display. So, we can find that for 7-segment display we have 7 segments a b c d e f g there will be one pin corresponding to common. So, depends upon whether this display is common anode type or common cathode type, this common has to be connected to either ground or plus 5 volts.

Here let us assume that this is common anode type common anode type. So, this common point has to be connected to plus 5 volts and if I send a logic 0 on any particular segment that segment will glows. This is a b c d e f g ok, this is interfacing. So, if I want to I mean implement a decimal counter on this particular 7-segment display to implement a decimal counter means first you display 0, then 1, then 2 so on up to you go up to 9, then after 9 again you come back to 0 continuously. And, you allow time between two consecutive counts as sometime say it can be 1 milli second 1 microsecond that is depends upon the application.

So, I want I mean program this as a decimal counter which counts from 0 to 9 repeatedly which is also called as a decade counter or this is because it counts from 0 to 9. This is also called as decade counter. Then, the first step is how to I mean display digits. 0, I want to display 0 like this I have only one option only. So, in order to display 0 so, these are the segments that has to be glown a b c d e f g has to be off. To display 1 we can select two ways – one is we can use only bc to display or we can use fe also if I select fe also this displays 1.

So, I want to display 2 like this. So, what are the segments? a b d e g. 3 I want to display like this, 4, 5, 6 this is depends upon your design. This can be excluded also without this also it will seems to be 6 only. 7 is this, 8 is all the segments, 9 also here this is also 9 or you can include this also ok. Depends upon your requirement so, I want to display the digits 0 to 9 like this ok. Now, what are the codes corresponding to this? I will store in a lookup table.

So, if I want to display 0, what is the code that is required? What should be the PA naught to PA 7? PA 7, PA 6, PA 5, PA 4, PA 3, PA 2, PA 1, PA naught ok. Decimal digit and then I will find out the corresponding code. So, decimal it is 0 say ok. So, in order to display 0 this is common anode type. This is PA naught is connected to a, this is connected to b, this is connected to c, d, e, f, g and this is connected to common point. So, in order to display any digit this should be always 1; common has to be connected to anode has to be connected to 1 and to glow any segment we have to send a logic 0 on that particular segment.

So, for 0 except g except g so, I have to glow all the remaining segments; so, g must be 1, all will be 0s. So, 0 means segment will be selected because this is common anode type. So, what is the hexadecimal code corresponding to 0; is 1 1 0 0 is 12 this is 0, 12 is nothing, but C 0 H in hexadecimal C 0 H ok. Similarly, you obtain from 1, 2s, so on up to 9. 1 this will be always 1 because this is common anode type. So, display 1 only I want to glow b and c remaining all has to be 1. So, this is 1 1 1 expect b and c this is b and c. So, both should be this is sorry b and c this is 1, this is d b and c should be 0, this will be 1.

What is the code? F 9 H corresponding to 2. So, I want to glow a b d e g a b d e g ok. So, what is the code? This will be 1. I have to display g also 0, f is not there so, 1. e – e is there so, 0; d also there 0; c is not there, 1; b is there, 0; a is there, 0. So, what will be the code? 1 0 1 0 is a 10; 0 1 0 0 is 4 A 4 H. Similarly, corresponding to 3 what is the code? This will be 1 always, 3 means I want only this a b c d g a b c d g. So, a b c d is a g should be 0 f is 1, e is also 1 a b c d remaining four are 0s. This will be B 0 H.

Corresponding to 4, so, I want this 4. So, what are the segments? Segments are b c f and g b c f and g this will be 1 b c f and g should be 0 f and g b and c should be 0 remaining all are 1s. So, what will be the code? 99H corresponding to 5 this will be 1, 5 means it will be a except f and e f and e will be 1s; f and e will be 1s remaining all will be 0s. So, this will be B0H this is B0H. 1 0 1 1 0 0 0, 5 is B0H 5 will be 1 2 3 except b and e b and e b and e. So, b should be 1. So, this will be 92H because this should be all distinct.

Similarly, 7 6 7 8 9; so, this will be always 1 only 6. 6 is except b remaining all has to be 0s. So, this will be 82H. 7 is only a b c has to be 1 remaining all are 1s. Here only a b c is 0 a b c should be this is also d 1 a b c should be 0 0 0. So, what will be value F? 8 H 8 is all must be 0 except the common pin. This is nothing but 80H. 9 is all except e, this is 0, this is 0, e should be 1 remaining all will be 0s this will be 90H. So, these are the code corresponding to this.

So, I am going to store these values into some memory location starting with some offset say some 6000 offset in a data segment whose starting address is say 50000. So, this will be 56000 offset is 6000, then 56001, 56002, 56003, 56004, 56005, 56006, 56007, 56008 and 56009. So, here I am going to store C0, F9, A4 this will be B0, 99, 92, 82, F8, 80 then 90. There this is you are not using ok. Now, I want to display these values through this first to display this C0 value then you display F9 value, you display A4 value and so on and in between you call a delay subroutine of 1 milliseconds.

So, what will be the program now? The program will be MOV AX comma initialize the data segment 5000H MOV DS comma DS comma AX and initially you have to only use this port A as output port because we are going to send the data from 8255 to output device and the remaining ports we are not using. And, we have already generated in the earlier classes to program all the ports in mode 0 as output ports 80H is the control word register. We have you have to load 80H inside this one, then this will program all the ports as output ports in mode 0 ok.

So, for that the instructions are MOV AL comma 80H OUT control word register address is FFH comma AL. So, these two instructions will program all the ports as output ports in mode 0 ok. A 0 I have to display 0 on here. So, what is the code corresponding to 0? Is C0 which is there in the location of 56000H. So, for that I am taking same logic I am going to use that I have used in the key board interface for decoding MOV AH comma 60 because this offset is 6000. So, in that 60 I have taking AH MOV AL comma 00 so that AX points to 6000. So, in the location with 6000 offset we have the code corresponding to C0 which is code corresponding to 0. So, we take MOV SI comma AX ok.

So, we are taking this SI with 60000H where 0 code is available ok. Then MOV AL comma contents of SI. So, after this instruction AL will contain C0H which is the code corresponding to 0 C0H. So, if a output here C0H from here a 0 will displayed on the 7-segment display. So, for that what is the instruction AL is having this one? OUT the port a address is FCH the contents of AL which is C0H, so that after this instruction a 0 will be displayed on this 7-segment display. Then you call a delay.

You can set any value of the delay depends upon your application ok. We can set 1 milli second, 2 milli seconds we know how to write the delay program then after that we have to display logic 1 ok. So, to display logic 1 I mean digit 1 whose code is present in 56001H. So, we have to just simply increment AL. INC AL INC AL compare AL with 09. So, up to 09 we have to increment this, we will compare with otherwise A because 9 also you have to display right. So, you take 1 above this 0A. Jump not equal UP ok.

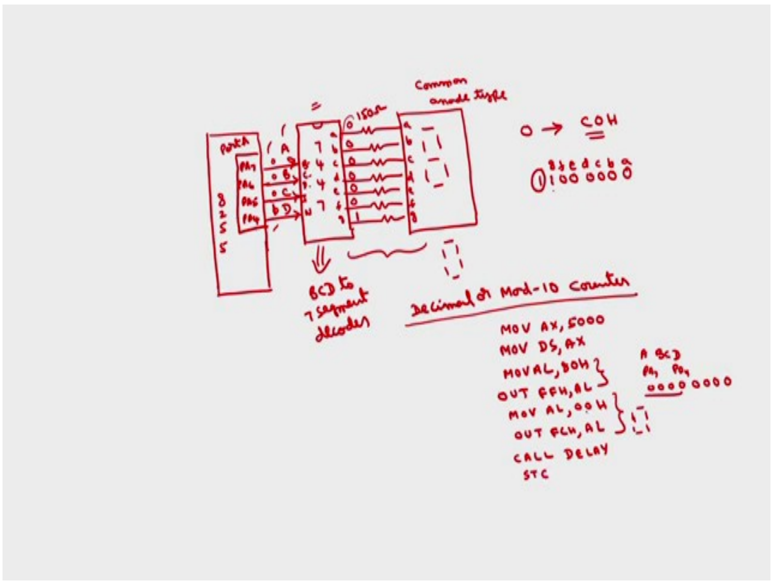
For 00 0A, if you compare it will be not equal 01 with 0A not equal. So, after 9, 9 also we are going to compare with A not equal this will be this particular instruction will be false, it will come out of the loop ok. So, if it is not equal you have to go to UP true; if this is 0A after displaying all the digits 0 to 9 then 0A with 0A they are equal. So, this is falls this will come out of the loop. So, from 0 to 9 this will go to the up where should be this up now? This up should be here. In AH permanently 60 is there, in AL we have this 0A. So, that is to be taken into this contents of SI. So, SI will be pointed to this I mean 0 601.

So, after this instruction AL minus 0A is performed, but the result is discarded. AL contents are here after the first iteration 01 that 01 remains in AL. So, when this is not true here it will go. So, in this AX, AH will be having 60 only; now in the second iteration AL will be having 01. So, the offset of 60 01 means which is corresponding to a code corresponding to digit 1 is stored in that particular location ok. Is it clear now?

Now, it will display here 0, 1, 2 with the delay of the delay that is going to be decided by the application so, it will go up to 9. After 9 what happens this will come out of the loop, but, I want to repeat the process. So, you write unconditional jump to up 2 or unconditional jump to repeat we can call. Where should be this repeat? This repeat should be again this. So that again AL will be taken with 00. So, the code corresponding to 0 will be displayed. So, like that this loop will continues. This is how we can interface a 7-segment display to the micro-processor 8086 and then we can program this 7-segment display as you wish.

So, here I have I mean explained this program to display a mod 10 counter using 8086 ok. So, this is one way to interface the common cathode display directly you can connect this port to the this 7 segments directly. If I want to connect multiple 7-segment displays in some applications we need multiple 7-segment displays if it is in a single 7-segment display I can do in the manner that I have explained now ok. So, in order to connect multiple 7-segment displays normally we will use a BCD to 7-segment decoder which is called 7447.

(Refer Slide Time: 39:52)



```
MOV AX, 5000
MOV DS, AX
MOVAL, 80H
OUT FFH, AL
MOV AL, 00 H
OUT FCH, AL
CALL DELAY
STC
```

There is a IC 7447 so, which will be having four B C D inputs B C D input and 7-segment output and the output of the 7-segment display we are going to connect to through this current limiting register to display 7-segment display. There are some additional pins for this 7447 that I am not explaining these are the important pins. So, I will connect to this corresponding display a to g this is a to a, b to b, c to c, d to d, e to e, f to f, g to g.

Now, this requires only four B C D inputs ok. So, this B C D I am going to give through the 8255. Instead of using a complete port here now you can use only four pins of any port and this is PA 3, PA 2 or you can use the higher order port is up to you PA 7, PA 6, PA 5, PA 4 only four pins of a port A remaining four pins I am not using this is port A. I am assuming the same addresses F C H is the address of the port A, F F H is the address of the control word register.

Now, you have to send this B C D code. If I send 0 0 0 0 then what is the code required for this if it is a common anode type only? If I call this inputs as A B C D. A B C D, this

A B C D is going to be send by 8255. This A B C D to common anode type as you have discussed in the previous slide. To display 0, what is the code? C0H ok. So, of course, this common point has to be connected to this this C0H in that this is 1 1 0 0 0 0 0 0 this is of course, common point, but we are not considering here.

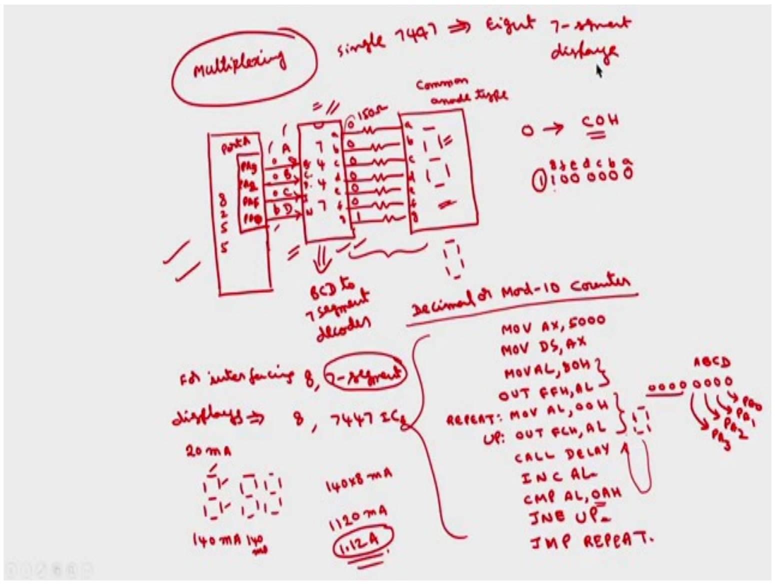
So, a to a b c is in this manner a b c d e f g. So, this 1 0 0 0 this will be generated here this will be 0 0 0 0 0 0 1. So, if I display 0 0 0 0 here output here, then corresponding 0 0 0 0 0 0 1 will be displayed there by here it will display 0 ok. Similarly, if I use 0 0 0 1 here a code corresponding to 1 display will be appeared here. So, this job will be performed by this BCD to 7-segment decoder. So, the job of the BCD to 7-segment decoder is as the name implies you give the input as BCD output is 7-segment display code ok. Depends upon the BCD 0, 0 code will be generated, 1 code will be generated and so on ok.

Now, how to write the program? Again to a implement the same decimal counter or mod-10 counter. Here basically you have to give only 4 bit BCD ok. So, assume the same locations MOV AX comma 5000 MOV DX comma AX, then this connections are we are assuming only we have to give output the BCD okay. MOV here also you have to out I mean program, all as output ports in mode 0 so, OUT FFH comma AL. So, these two instructions will program all the ports as output ports in mode 0 ok. Now, we have to display 0 0 0 0 on 7447 then 0 1 0 2 and so on.

So, I am taking MOV AL comma 00H out this is port A address is FCH FCH comma AL. With these two instructions what happens? This 00H means 0 0 0 0 0 0 0 0 this four 0 0 the last 0s are this first 4 PA 7 to PA 4 these are connected to this is connected to A B C D ok. So, A B C D is 0 0 0 0 means so, a 0 will be displayed. So, with these two instructions A 0 will be displayed on 7-segment display. Then you call a DELAY of any time that is required by the application then I have to increment this.

So, we have to increment this 0 to 0 1 ok. So, for that you set the carry set the carry this will be more convenient if I use the lower order 4-bits instead of the higher order 4 bits, let us assume that this is PB 3, 2, 1 and 0 ok.

(Refer Slide Time: 46:37)


Multiplexing
single 7447 ⇒ Eight 7-segment displays
Common anode type
BCD to 7 segment decoder
For interfacing 8, 7-segment displays ⇒ 8, 7447 ICs
20 mA
140 mA
140×8 mA
1120 mA
1.12 A
Decimal or Mod-10 Counter
ABCD
PA0
PA1
PA2
PA3

```
         MOV AX, 5000
         MOV DS, AX
         MOV AL, 80H
         OUT FFH, AL
REPEAT:  MOV AL, 00H
UP:      OUT FCH, AL
         CALL DELAY
         INC AL
         CMP AL, 0AH
         JNE UP
         JMP REPEAT.
```

So, this will be last 4 bits A B C D will be last 4-bits for the previous one also we can write the program, but it will be somewhat complicated. This I am connecting to PA naught, this to PA 1, this to PA 2, this to PA 3 ok. So, after that I will write INC AL it will be 0. I will compare with AL comma 0AH because I want to display up to 9 so, I am comparing with the 10. So, whenever this AL becomes 10 so, this I mean this two will be equal it will come out of the loop a 10 will not be displayed it will display up to only 9 only.

I will write the same instruction JMP not equal UP. So, this UP is because you have incremented only outputting this is the UP. So, this will display 1 now, then in the next loop it will display 2, next 3, next 4, next 5, up to 9. After 9 AL becomes 0A and we are comparing this 0A with 0A. So, this will be equal this will be false this will come out of the loop. So, you have to write unconditional JMP to REPEAT. So, where should be this repeat again you have to take AL with 0 0 0. So, this will continuously display 0 to 9 on this particular 7-segment display.

So, this program is relatively simpler because we are using BCD to 7-segment decoder ok. So, if I want to connect a multiple 7-segment displays so, suppose if I want to connect eight 7-segment displays then we have to use 8, 7447s ok; if you want to directly use ok. So, for interfacing 8, 7-segment displays we have to use 8, 7447 ICs and when a particular segment is on it draws a current of 20 milli amps.

And if I want to display 8 on each segment on each display so, if you have single display to display 8 this will draw 20 milli amps this will draw 20 milli amps. So, total this will be 140 milli amps. If I use 8 such displays what is the total current this is 140, this is 140 milli amps and so on. So, total will be 140 into 8 milli amps. This will be of the order of 1120 milli amps or 1.12 amps, this is a large current. So, this will draw more curve. This is one of the drawback of using this type of configuration for displaying multiple 7-segment displays multiple 7-segment displays.

This is one drawback is it draws more current thereby more power consumption. In addition we have to use 7, 7447s ok. Instead of using this 7, 7447s we can use a technique called multiplexing technique multiplexing technique. If you use the multiplexing technique using one single 7447 itself we can display eight 7 BCD to 7-segment displays or eight 7-segment displays ok. So, how to I mean connect eight 7-segment display using only single 7447 in a multiplexing mode that we will discuss in the next class ok.

Thank you.