Welcome students.

I'm Caje Francis Pinto assistant

professor from Department of Electronics,

St Xaviers College.

The course code is ELC 104 and the course

title is microprocessor and microcontrollers.

The unit which will be covering

is unit 6. 8051 programming and

the module number named is 8051.

Addressing modes and accessing memory

location using various addressing modes.

The outline of this module is as follows.

Different types of addressing modes

register addressing immediate addressing,

direct addressing, indirect addressing,

and index addressing.

Through this module,

the students will be able to explain the

different types of interesting modes.

In 8051 such as register addressing mode,

immediate addressing mode,

direct addressing mode,

indirect addressing mode,

and indexed address.

So now we'll have a look at the different

addressing modes present in 8051.

The CPU can access data in various ways and

we have basically 5 addressing modes in 8051,

which is a immediate register.

Direct, indirect, and index.

So we shall have a look at

immediate addressing mode.

In this,

the data is directly specified

in the instruction itself,

and the immediate data must be

proceeded with the hash sign.

An example is given as mov R0 comma,

hash 0F0H.

So the value of F0 is loaded

into the register R0.

The immediate value which is

loaded is a maximum of eight bits.

The next one is the DPTR which

is called as the data pointer,

which is a 16 bit register and

here the data pointer can also be

accessed as a 2 8 bit registers.

That is high byte DPH and the low byte DPL.

So we'll have an example of how

you load a 16 bit address into

a register mov DPTR.

2000 so the value of 2000 is copied

into the data pointer register.

The same thing can be also written

in the different way. Mov DPL,

comma hash zero and move DPH comma hash 20H.

The next addressing mode, what we'll

be discussing is register addressing.

This involves the use of registers

to all the data to be manipulated.

This is having a direct access to the

8 registers named as R0 through R7.

The source and the destination

resistors must match in size.

The movement of the data between

the registers is not allowed,

so we can move the contents of R0 into A.

You can move the contents

of accumulator into R2.

We can add the contents of R5

with the contents of accumulator.

Or you can also add the contents of

R7 with the contents of emulator.

And next whatever,

there in the accumulator can be saved

in the contents of R6 register.

With regards to DPTR,

the value 25F5 is loaded into the DPTR,

whatever's there in DPL value

can be stored in R7,

and whatever is there in

DPH can be saved in R6.

The next addressing mode what will

be discussing is direct addressing.

Direct addressing can access any

internal data memory between 30 to

7 FH all on chip memory locations

and registers have 8 bit addresses.

You can use the 8 bit address

in the instruction itself.

So over here the data is obtained

directly from the memory location.

So we have an example.

Move a 4H.

It means that whatever stored

in the memory location,

4H will be copied into the

contents of accumulator.

The second example is whatever is

there an R4 for this particular

value will be copied into A,

which is actually register 4,

which is just the same job of move a 4H.

The next third one is move the

contents of a into memory location.

56 H so whenever you want to

transfer data from the accumulator

to any manual location directly,

you can specify the location

in this particular format.

Also,

we use direct addressing for

special function registers,

so we have the address for

accumulator as E0H B register.

It is F0 for PSW.

The address is D0,

for stack pointers a 81.

We'll have a look for port zero.

It is 80, port one is 90,

port two is a zero and port three is B0.

So if you want to use.

To move the contents,

you can either use the address

or either use the mnemonics.

So how do we use direct addressing

for special function registers

and their addresses?

So special function register

can be accessed by their names

or by their addresses.

Between 80H and FFH.

So over here the value of 55 is

loaded into the accumulator,

so the address of accumulator is E0.

So you can either use a or you

could have used the address of a.

Also,

the value of 55 is loaded into a,

so the address of B register is F0 and

the contents of R0 can be moved into B.

So this is an alternative

of writing the program.

Either you load the value of 55 to A,

or you load the value of 55

into the address of the Same.

So advisable is to use the 2nd and 4th.

So next part is to write the code to

send 55 to port one and port 2 using

their names and using their addresses.

So the value of 55 is loaded into A

and the value of a is loaded into P1,

the same value which is there

in the is also loaded into P2,

so this is the program how we are using

with their names now as perform the table,

the port one address is 80

and the port to address is.

Is you so you can load the value of

55 into A and the same value of A can

be loaded into the port one of the

microcontroller 8051 which is written as 80.

If you want to load the contents of

accumulator into port1 using direct address,

you can use the direct address as a 80H.

Now with regards to

indirect addressing here,

a register is used as a pointer to the data,

so we have two 8 bit pointers.

One is R0 and one is R1.

16 bit pointer is the DPTR when R and

R1 hold the addresses of RAM location,

they must be preceded by the

@ sign,

so we'll have a look at an example of

how do you use indirect addressing.

The first thing is the value

of 40 is copied into R1,

so the starting location of the.

Ram is 40,

so R1 is pointed to the location 40H.

Then the contents pointed at 48th

will be copied and transferred

to the contents of accumulator.

The next one is the contents of

R1 can be copied to the contents

pointed at memory location or not.

So remember that for in target

addressing @ is used.

indirect addressing is also

used for accessing external memory,

so you can use R0 and R1 to point to

external memory location from 00 to FF.

So over here,

the difference is there's an additional

letter called X which symbolizes

that it is an external memory.

So MOVX A comma @ R1,

moves the constants of external memory

location whose address is in R1 into A.

If you are using a data pointer,

this data pointer can point to

64K memory of external memory.

So over here,

the contents of external memory,

that is data pointer will be moved

into the contents of accumulator.

The last one is the index addressing.

This is used in accessing data

elements of lookup table entries

located in the program ROM.

It uses a register called as PC or data

pointer for storing the base address,

and another register is used

for storing the offset address.

That is accumulator.

So we use the instruction MOVC,

C means code, so the contents of

A are added to the 16 bit register

data pointer to form the 16.

Address of the needed data.

So the effective address

is the sum of the two.

A is equal to pointer plus offset

pointer is obtained from PC or DPTR and

offset is obtained from accumulator.

So an example is given to

you for index addressing.

It is MOVC A @ DPTR,

so it moves the byte from the memory

located at DPTR plus A to the accumulator.

So we have an example of

addressing mode Immediate

direct register and index.

So in immediate it is new data is in the

form of the operand for direct.

The data is stored in the internal

data memory and when it comes to

register you can use R1 or DPTR

and this value can be internal

memory of external data memory.

When it comes to index,

it is a PC or a DPTR,

and the address can be internal

or external program memory.

So we will come to the summary of

addressing modes, so in register

addressing we use the contents.

We can move the contents from

accumulator to registers,

and if you're talking with respect

to indirect addressing modes,

we have to use 2 8 bit pointers

called as R0 and R1.

So you can move the contents of

accumulator comma at the @Ri

so this RI can be 01,

so whatever is.

Mounted in this area,

I will be moved to the contents of.

Also,

we can use direct addressing,

meaning the whatever the location is

specified in the internal RAM can be

moved to the contents of accumulator.

If you're using external memory,

if you want to transfer external memory,

you have to use instruction

call as MOVX space.

A comma @ Ri Ri can be 01,

so whatever is there in the external

memory location pointed by RR,

One will be transferred to the accumulator.

But if the external RAM is

a 16 bit memory address,

we can use our DPTR where the contents

of the external memory location can

be transferred to the accumulator.

The next one is the index addressing mode.

You can use the DPTR or you can use PC,

so over here it is MOVC

space A at the rate of a PC.

So over here the internal external form

the code can be transferred from the

memory location to the accumulator.

With this we come to the end of our module.

These are my references,

thank you.