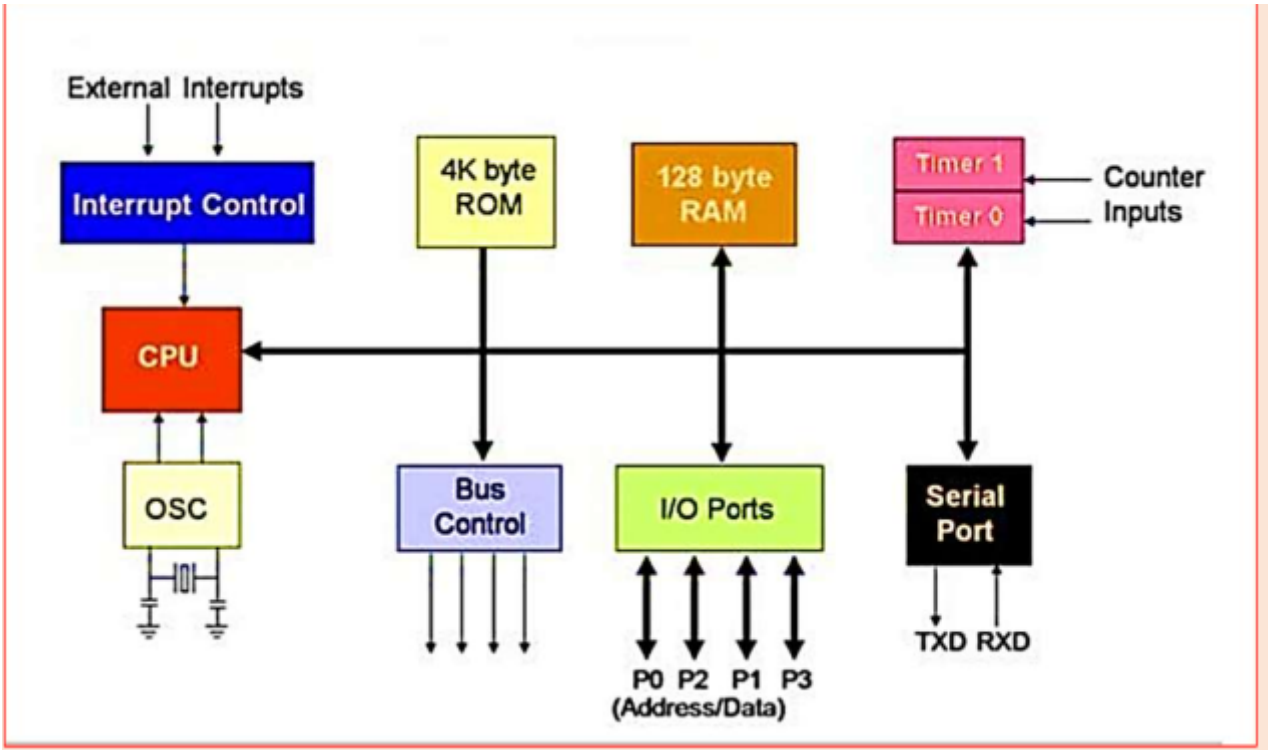
Microprocessors Module 5 Important Topics

8051 block diagram



The internal architecture of 8051 contains the following blocks

- Accumulator (A)
  - Saves operand for operations by ALU
- B Register(B)
  - Saves second Operand for ALU
- PSW (Process Status Word)
  - 8 bit register to save the status information
- Stack Pointer (SP)
  - 8 bit register is incremented before data is stored onto the stack using PUSH Instruction
- Data Pointer (DPTR)
  - Holds external data memory address of data being fetched or to be fetched
- Port P0
- Port P1
- Port P3
- Serial Data Buffer (SBUF)

    - Contains 2 independent registers
    - Transmit Buffer
        - Parallel in Serial out register
    - Receive buffer
        - Serial in Parallel out register
- Timer register
    - Two 16 bit register, can be accessed as their lower and upper bytes
        - TL0 -> Lower byte of timing register 0
        - TH0 -> Higher byte of timing register 0
        - TL1 ->Lower byte of timing register 1
        - TH1 -> Higher byte of timing register 1
- Control Registers
    - IP - interrupt Priority Register
    - IE - Interrupt Enable register
    - TMOD - Timer Mode Register
    - TCON - Timer Control Register
    - SCON - Serial Control Register
    - PCON - Power Control Register
- Timing and control unit
    - This unit derives all the necessary timing and control signals required for the internal operation of the circuit.
- Oscillator
    - This circuit generate the basic timing clock signals for the operation of the circuit using crystal oscillator
- Instruction Register:
    - This register decodes the opcode of an instruction to be executed
- RAM
    - RAM is 128 byte memory for the read and write and is indirectly and directly addressable.
- ROM
- ALU
- SFR Register Bank
    - 4 Register Banks each of 8 registers

Memory organization of 8051

- In 8051 Microcontroller, memory is divided into program and data memory

- Program memory
  - Program memory means ROM
    - Read only
  - Used to store constant variables
  - Allows External program memory to be added
  - External ROM is access when
    - EA pin is 0
- Internal Data memory
  - 128 byte internal RAM is organized into three distinct areas.
    - 1. 32 Bytes
      - 00H to 1FH
      - 32 Working Registers
        - 4 banks of 8 registers each
        - These banks are number 0 to 3
        - Made of 8 registers named R0 to R7
        - To know which banks are currently in use
          - We use the bits RS0 and RS1 in PSW
            - PSW = Program Status Word
        - Bank 0 is select on Reset
    - 2. A Bit Addressable Area of 16 bytes
      - This Forms a total of 128 Addressable Bits
      - These bits are useful when Programs needs to only remember a binary event
        - Like Switch on, Light off
    - 3. A general purpose RAM area
- External Data Memory
  - Slower to access than internal data memory
  - Accessed thorugh indirect addressing mode
- I/O Addressing
  - 4 I/O ports of 8 bits
    - Each can be configured as input or output

Register organization Of 8051

- Internal memory of data of 8051 is divided to 2 groups
  - Set of eight registers (R0 to R7)
  - Scratchpad memory (High speed internal memory)

- 8051 Provides 4 Register banks, But only one register bank can be used at any point of time
- For selecting the register bank, 2 bits of PSW are used
  - PSW = Program Status Word

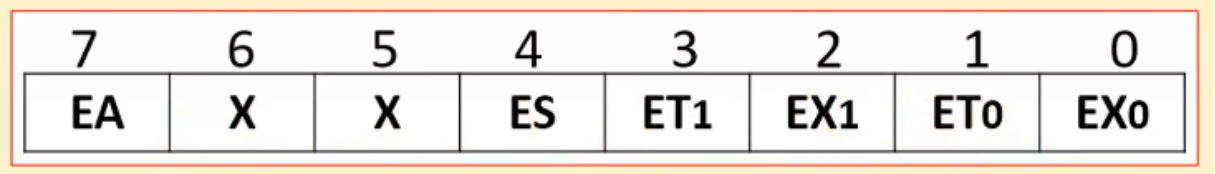
Address Range	Register Bank
00H to 07H	Register Bank 0
08H to 0FH	Register Bank 1
10H to 17H	Register Bank 2
18H to 1FH	Register Bank 3

Interrupt

- Interrupt is a signal sent by external device to the processor, to request the processor to perform a particular task or work.
- When a processor recognizes an interrupt, it saves processor status in stack
  - Then it calls and executes interrupt Service Routine (ISR)
- At the end of ISR, it restores processor status and program control is transferred to main program
- Types of interrupts are
  - Software and Hardware interrupt
  - Vectored and non vectored interrupt
  - Maskable and non maskable interrupt
- Types of interrupt in 8051
  - Timer 0 overflow interrupt TF0
  - Timer 1 overflow Interrupt TF1
  - External hardware interrupt INT0
  - Internal hardware interrupt INT1
  - Serial communication interrupt RI/TI
- There are 2 Special function registers (SFRs) for interrupt handling
  - Interrupt Enable Register
  - Interrupt Priority Register

Interrupt Enable Register

- Responsible for enabling and disabling interrupt

- Bit addressable register
  - 

7	6	5	4	3	2	1	0
EA	X	X	ES	ET1	EX1	ET0	EX0

  - Each of these determines Enable/Disable of the below registers
  - EX0 -> External Interrupt 0
  - ET0 -> Timer 0 overflow interrupt
  - EX1 -> External interrupt 1
  - ET1 -> Timer 1 overflow interrupt
  - ES -> Serial port interrupt
  - EA -> All interrupts
  - Here positions 6 and 5 are not used

Interrupt Priority Register

- Its possible of changing priority levels of interrupts by setting or clearing corresponding bits in the IP register

TCON Register

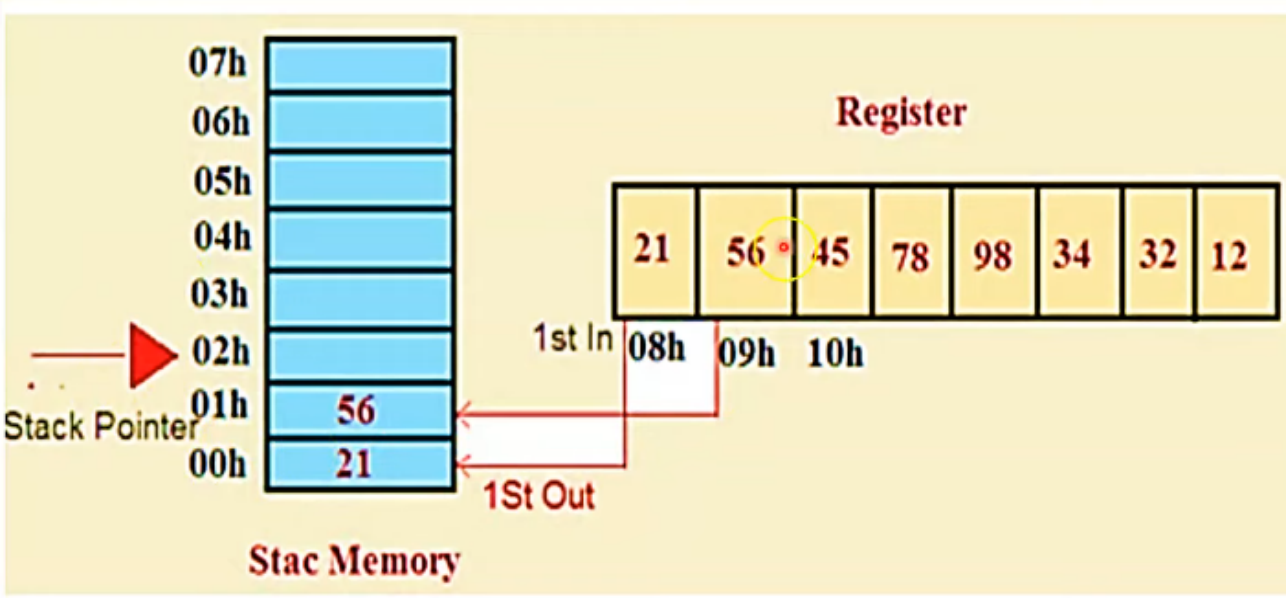
- Specifies the type of external interrupt to 8051 microcontroller

Stack

- Stack is a area of RAM allocated to hold temporarily all parameters of the variable
- Whenever the function is called parameters and local variables are added to stack
- When Function returns, Parameters and variables are removed from stack
- Register used to access the stack is called stack pointer register
- When we push something into stack memory, the stack pointer increases

PUSH

- Used to take values from any register and storing in starting address of stack pointer



PUSH operation of Stack

- Consider this code

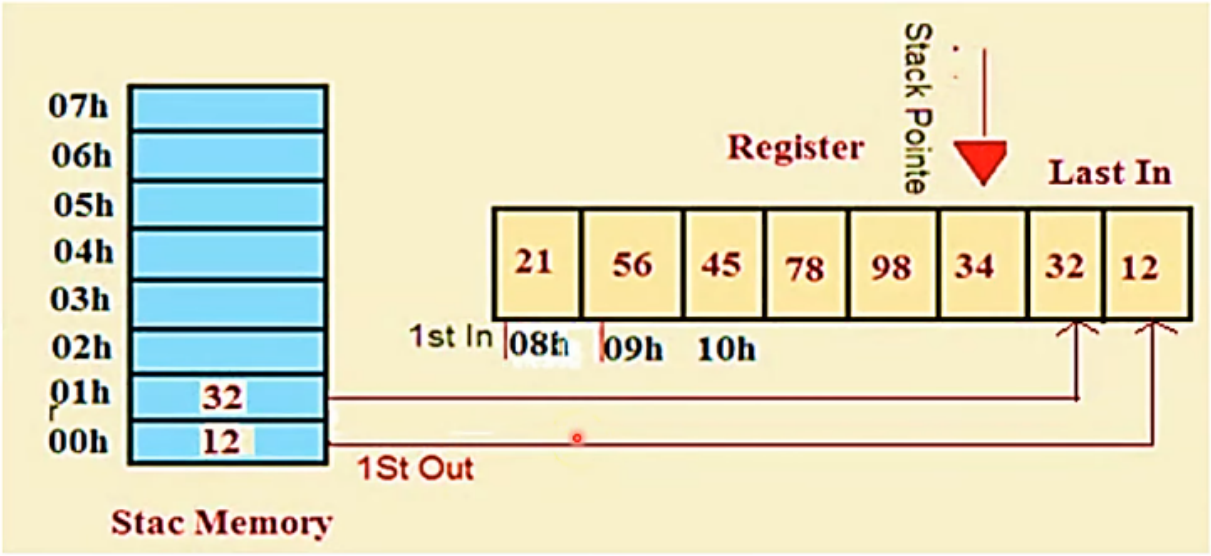
```
Eg:  0000H
     MOV  08H,  #21H
     MOV  09H,  #56H
     PUSH 00H
     PUSH 01H
     END
```

- 21 and 56 and moved to located 08 and 09 in the register
- PUSH 00H
  - Stack pointer is initially at 08H
  - The value at stack pointer is pushed to Stack
    - Here the value at stackpointer (08H) is 21
    - 21 is pushed to stack
  - After pushing, stack pointer is incremented by 1
    - Now its value is 09H
- PUSH 01H
  - Stack pointer is now at 09H
  - The value at stack pointer is pushed to Stack

- Here the value at stackpointer (09H) is 56
- 56 is pushed to stack

POP

- Used for placing values from stack pointers maximum address to other register address
- It decrements the stack pointer by 1



POP Operation in Stack

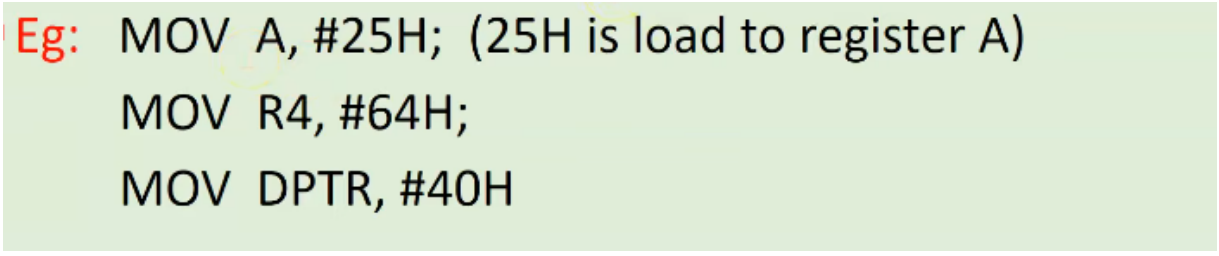
-
```
Eg:  0000H
     MOV  00H, #12H
     MOV  01H, #32H
     POP 1FH
     POP 0EH
     END
```
- In the above code
- First we move 12 and 32 to the stack memory locations 00H and 01H
- POP 1FH
  - Value at top of the stack is popped
    - Here its 12, located at 00H
  - This popped value is stored at 1FH
  - Stack pointer is decremented

- POP 0EH
  - Value at top of the stack is popped
    - Here its 32, located at 01H
  - This popped value is stored at 0EH

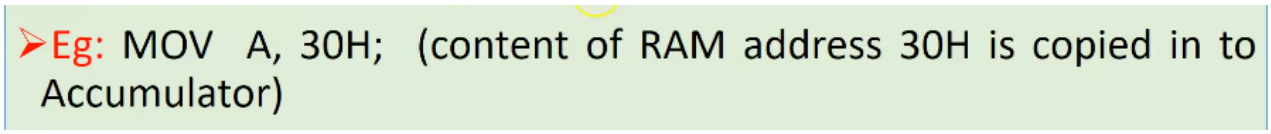
Addressing modes

- Its the way in which instruction is specified or different ways that are used by processor to access or store data
- 8051 Provides 5 Addressing modes
  - Immediate
  - Direct
  - Register
  - Register Indirect
  - Indexed

Immediate Addressing mode

- Here source operand is constant
- Immediate data is preceded by sign #
- Example
  - 

Direct Addressing mode

- Direct address of memory location is provided in this instruction
- Only RAM and SFR (Special function Register) address can be used here
  - 

Register Addressing Mode

- Operand is contained in the specific register of microcontroller
  - So the User must provide the name of register from where data needs to be fetched

- Eg: MOV A, R0 ; (content of R0 register is copied in to Accumulator)

Register Indirect Addressing Mode

- Address of a memory location is indirectly provided by a register
- @ Sign will indicate that the register holds the address of the memory location
  - Fetch the content of memory location whose address is provided in the register
- Eg: MOV A, @R0 ; (copy the content of memory location whose address is given in R0 register)

Indexed Addressing Mode

- Used for accessing data from Look up table
  - Look up table is located in Program code ROM of 8051
- Eg: MOVC A, @A+ DPTR (Here C means code. The content of A register is added with content of DPTR and the result is copied to A register)
- Because we are taking from ROM, we use MOVC instead of MOV

Instruction sets

- An 8051 Instruction consists of an Opcode Followed by operands
- Opcode part of instruction contains the Mnemonic
  - This Specifies the type of operation to be performed
  - All mnemonics are of one byte size
- Operand part of the instruction defines the data being processed by the instructions
  - The operand can be any of the below
    - Data value
    - I/O Port
    - Memory Location
    - CPU Register
  - There can be multiple operands
  - MNEMONIC DESTINATION OPERAND, SOURCE OPERAND

Based on operation performed, all instructions in 8051 instruction set are divided into 5 groups

1 .Data Transfer Instructions

- Associated with transfer of data between registers or external program/data memory

Sl. No	Instruction Format	Function Performed
1.	MOV dest, src	Copy the Content of Source to Destination
2.	MOVX dest, src	Used to move to/from external data memory only
3.	MOVC dest, src	Used to move from program memory (ROM) Only
4.	PUSH src	Copies one byte from source to stack top
5.	POP dest	Copies one byte from stack top to destination
6.	XCH	Exchanges data between two sources
7.	SWAP A	Exchanges upper and lower nibbles of A

2. Arithmetic Instructions

- Performs basic operations like add, sub, division, multiplication etc
- After execution, result is stored in first operand

Sl. No	Instruction Format	Function Performed
1.	ADD A ,Rn	Adds the register to the accumulator
2.	ADDC A ,Rn	Adds the register to the accumulator with a carry flag
3.	SUBB A ,Rn	Subtracts the register from the accumulator with a borrow
4.	INC A	Increments the accumulator by 1
5.	DEC A	Decrements the accumulator by 1
6.	MUL AB	Multiplies A and B
7.	DIV AB	Divides A by B
8.	DA A	Decimal adjustment of the accumulator according to BCD code
9.	CLR A	Clear the value in A(a=0)
10.	CJNE dest,src,target	compare the Source and Destination, and jump to target if they are not equal

3. Logical Instructions

- Logical instructions perform logical operations

Sl. No	Instruction Format	Function Performed
1.	ANL dest,src	Logically AND the source and the Destination
2.	ORL dest,src	Logically OR the source and the Destination
3.	CPL dest	Complement- Logically NOT the destination
4.	XRL dest,src	Logically XOR the source and the Destination

Boolean or Bit Manipulation Instructions

- Deals with bit variables
- The Boolean/Bit Manipulation instructions are
  - Data Manipulation Instructions
    - **a. Data Manipulation Instructions**

Sl. No	Instruction Format	Function Performed
1.	SETB Bit	Set the indicated Bit
2.	CLR Bit	Clear the indicated Bit
3.	CPL Bit	Complement the Indicated Bit
4.	MOV C, Bit	Move the indicated bit to C(Carry Flag)
5.	MOV Bit, C	Move the C(Carry Flag) to indicated bit
6.	ANL C, Bit	Move to C(Carry Flag) the logical AND of C and the indicated bit
7.	ANL Bit, C	Move to the bit , the logical AND of C and the indicated bit
8.	ORL C, Bit	Move to C(Carry Flag) the logical OR of C and the indicated bit
9.	ORL Bit, C	Move to the bit , the logical OR of C and the indicated bit

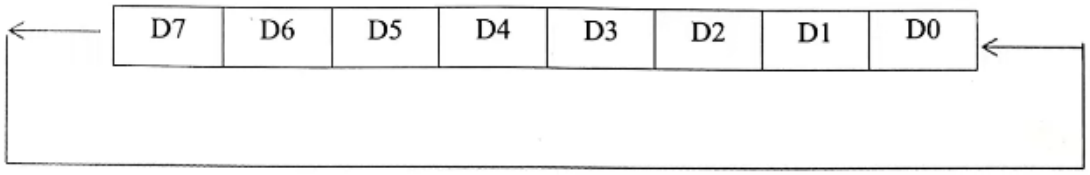
  - Port Instructions
    - Can either take data in or put data out
    - Uses MOV
      - **MOV A, P1** (copy the content of port1 to accumulator)
      - **MOV P2, R1** (copy the content of Register R1 to port2)
  - Complement Instructions
    - Only A register and RAM address can be used as detination

- Ex: **CPL A** (Complement the contents of A)
  CPL 43H (Complement the contents of RAM address 43H)

- Rotate Instructions
  - There are 4 rotate and 1 swap instruction
  - Only A Register can be used as destination

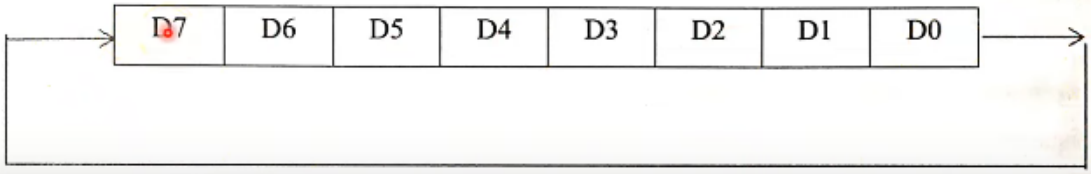
i) RL A ; Rotate left the content of A

The A register is rotated left and D7 appears in the D0 position after rotation.



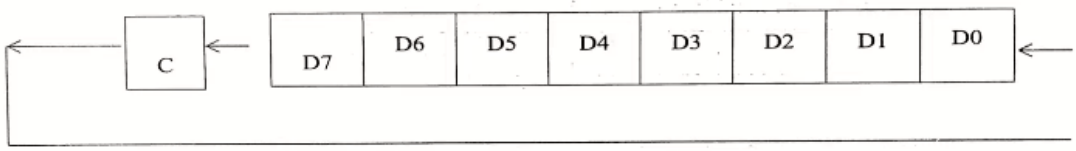
ii) RR A ; Rotate right the content of A

The A register is rotated right and the bit D0 is moved to D7



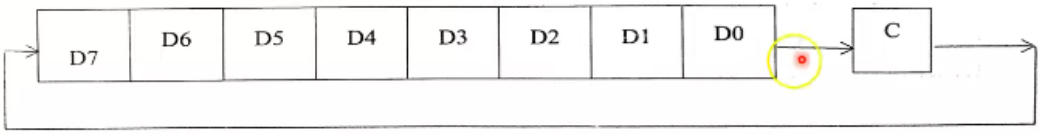
iii) RLC A ; Rotate left through carry

The A register is shifted left and bit D7 is moved to carry, while the carry bit is moved to D0 of A

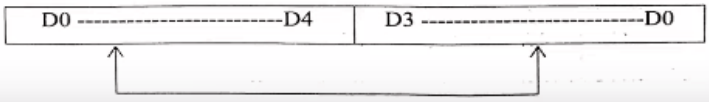


iv) RRC A ; Rotate RIGHT through carry

The A register is shifted right and bit is moved to D7, while the D0 moves into the carry.



v) SWAP A ; This is a special instruction where the lower and upper nibbles of A are swapped.



5. Program Branching Instruction (Flow Control Instruction)

- Flow Control instructions are used to divert flow of program
- These instructionsa are used to implement loops and subroutine calls
- Branch instructions change flow of a program by modifying PC value (Program counter)

Unconditional Branch Instructions

1. SJMP Target

- SJMP stands for Short Jump
  - Destination of jumping is expressed as relative number, 8 bits
  - 8 bit number represetns distance between current PC value and target Address
- If number is negative, its backward jump
- If number is positive, its forward jump

2. LJMP Target

- Long Jump, Not relative
- 3 byte instruction
- When this instruction is executed
  - PC Value is replaced by 16 bit number
  - This value ranges from 0000 to FFFF

3. AJMP Target

- Absolute Jump, also a relative jump

Conditional Branch Instructions

- These are branching based on conditions

- All conditional jumps are short jumps

Sl. No	Instruction Format	Function Performed
1.	JC target	Jump if CY=1
2.	JNC target	Jump if CY=0
3.	JZ target	Jump if the register A=0
4.	JNZ target	Jump if the register A is not zero
5.	JB bit, target	Jump if the bit=1
6.	JNB bit, target	Jump if the bit=0
7.	JBC bit, target	Jump if the bit=1. Then clear bit
8.	DJNZ byte, target	Decrement the byte, and jump if the byte is zero