Unit 2: Instruction set of 8051

2.1 Instruction set:

An **instruction** is a binary pattern designed inside a microcontroller to perform a specific function. The entire group of instructions is called the **instruction set** which determines what functions the microcontroller can perform. The complete instruction set of 8051 can be grouped in five different functional groups –

1. Data Transfer Group
2. Arithmetic Group
3. Logical Group
4. Bit Manipulation Group
5. Branching and Looping Group

2.1.1 Data Transfer Instructions

The microcontroller spends its most of the time in copying the data one to another. So 8051 instruction set is full of data movement instructions.

Mnemonic	Operation	Addressing mode	No.of bytes	Example	Description
MOV A ,Rn	Rn → A	Register	1	MOV A ,R1	Copy the data from R1 of the selected register bank and store it to register A
MOV A ,direct	Addr → A	Direct	2	MOV A , 60H	Copy the data from specified address i.e. 60H and store it to register A
MOV A ,@Rn	Addr(Rn) → A	Indirect	1	MOV A ,@R1	Copy the data from the address specified within R1 of the selected register bank and store it to register A
MOV A ,#data	Data → A	Immediate	2	MOV A ,#30H	Copy the data immediate data specified within instruction to register A
MOV Rn ,A	A → Rn	Register	1	MOV R7 ,A	Copy the data from register A to register R7 of the selected register bank
MOV Rn,,direct	Addr → Rn	Direct	2	MOV R1,30H	Copy the data from specified address i.e. 30H and store it to R1 of the selected register bank
MOV Rn,#data	Data → Rn	Immediate	1	MOV R5,#20H	Copy the data immediate data specified within instruction to register R5 of the selected register bank
MOV direct ,A	A → addr	Direct	2	MOV 70H ,A	Copy the content of the register A to the address specified within the instruction i.e. 70H
MOV direct,Rn	Addr → Rn	Direct	2	MOV 30H,R2	Copy the content of the register R2 of the selected register bank to the address specified within the instruction i.e. 30H

MOV direct,direct	Addr → Addr	Direct	3	MOV 30H,60H	Copy the content of the memory location 60H to the memory address 30H
MOV direct ,@Rn	Addr(Rn) → addr	Register indirect	2	MOV 30H ,@R0	Copy the content of the memory addresses specified in the register R0 of the selected register bank to the memory address 30H
MOV direct ,# data	Data → addr	Immediate	3	MOV 20H ,# 30H	Copy the immediate data 30H to the memory address 20H
MOV @Rn,A	A → Addr (Rn)	Register	1	MOV @R3,A	Copy the content of register A to the addresses specified within R3 of the selected register bank.
MOV @Rn,direct	Direct → Addr (Rn)	Direct	2	MOV @R7,30H	Copy the content of memory location 30H to the address specified within R7 of the selected register bank.
MOV @Rn,#data	Data → addr	Immediate	2	MOV @R5,#40H	Copy the immediate data to the memory location specified within register R5 of the selected register bank.
MOV DPTR,#data 16 bit	Data → DPTR	Immediate	3	MOV DPTR,#7000H	Copy the immediate data to the 7000h to the DPTR
MOVC A,@A+DPTR	(A+DPTR) → A	Indirect	1	MOVC A,@A+DPTR	Add the content of the register A and DPTR to generate address. copy the content of generated address to the register A
MOVX A,@DPTR	addr (DPTR) → A	Indirect	1	MOVX A,@DPTR	Copy the content of the address specified within DPTR to the register A
MOVX @DPTR,A	A → addr (DPTR)	Indirect	1	MOVX @DPTR,A	Copy the content of register A to the address specified in DPTR
MOVX @Rn,A	A → addr (Rn)	Indirect	1	MOVX @R0,A	Copy the content of register A to the address specified in R0 of the selected register bank
PUSH direct	addr → SP SP=SP+1	Direct	2	PUSH DPH	Copy the content of DPTR higher byte to the memory location pointed by the stack pointer
POP direct	SP → Direct	Direct	2	POP DPH	Copy the content of the memory location pointed by stack pointer to DPTR higher byte
XCH A ,Rn	A ⇄ Rn	Register	1	XCH A,R5	Exchange the data between register A and register R5 of the selected register bank

XCH A ,direct	addr ⇄ Rn	Direct	2	XCH A,50H	Exchange the data between register A and the specified address within the instruction i.e. 50H
XCH A ,@Rn	A ⇄ addr(Rn)	Register indirect	1	XCH A,@R5H	Exchange the data between register A and the address specified within R5 register of the selected register bank.
XCHD A ,@Rn	A(3-0) ⇄ addr(Rn)(3-0)	Register indirect	1	XCHD A,@R5H	Exchange the lower nibble of data between register A and the address specified within R5 register of

2.1.2 Arithmetic Instructions

Some of the microcontroller application such as weighing scale, speedometer, flow meter, control system etc requires the mathematical calculations. The 8051 microcontroller has 24 different opcodes for arithmetic operations and they can be classified as follows

1. Addition
2. Subtraction
3. Increment
4. Decrement
5. Multiplication
6. Division
7. Decimal Addition

2.1.2.1 Addition:

The instruction ADD is used for the addition of the two operands. For addition, one of the operand should be present within register A. The result of addition is always stored in register A. Flags are modified according to the result.

Mnemonic	Operation	Addressing mode	No. of Bytes	Example	Description
ADD A,Rn	A+Rn → A	Register	1	ADD A,R5	The content of register A added with content of register R5 of the selected register bank and result is stored in register A
ADD A, direct	A+ addr → A	Direct	2	ADD A,20H	The content of register A is added with content of memory location specified within instruction i.e. 20H and result is stored in register A
ADD A, @Rn	A+ addr(Rn) → A	indirect	1	ADD A,@R0	The content of register A is added with content of memory location specified within register R0 of the selected register bank and result is stored in register A
ADD A, #data	A+ data → A	Immediate	2	ADD A,#30H	The content of register A is added with immediate data and result is stored in register A
ADDC A,Rn	A+Rn+CY → A	Register	1	ADDC A,R5	The content of register A is added with content of register R5 of the selected

					register bank and a carry bit. The result is stored in register A
ADDC A, direct	A+ addr+CY → A	Direct	2	ADDC A,20H	The content of register A is added with content of memory location specified within instruction i.e. 20H and a carry bit. The result is stored in register A
ADDC A, @Rn	A+ addr(Rn)+CY → A	indirect	1	ADDC A,@R0	The content of register A is added with content of memory location specified within register R0 of the selected register bank and a carry bit. The result is stored in register A
ADD A, #data	A+ data → A	Immediate	2	ADDC A,#30H	The content of register A is added with immediate data and a carry bit. The result is stored in register A

2.1.2.2 Subtraction: The instruction SUB is used for the subtraction of the two operands. For subtraction , one of the operand should be present within register A. The subtraction is performed using 2's compliment method. The result of subtraction is always stored in register A. Flags are modified according to the result.

Mnemonic	Operation	Addressing mode	Number of Bytes	Example	Description
SUBB A, Rn	A-Rn-CY → A	Register	1	SUBB A,R5	The content of register R5 and a carry flag are subtracted from the content of register A. The result is stored in register A
SUBB A, direct	A- addr-CY → A	Direct	2	SUBB A,20H	The content of memory location and a carry flag are subtracted from the content of register A. The result is stored in register A
SUBB A, @Rn	A- addr(Rn)-CY → A	Indirect	1	SUBB A,@R0	The content of memory location whose address is stored in R0 and a carry flag are subtracted from the content of register A. The result is stored in register A
SUBB A, #data	A- data-CY → A	Immediate	2	SUBB A,#30H	The immediate data and a carry flag are subtracted from the content of register A. The result is stored in register A

2.1.2.3 Increment : The increment is performed using instruction INC. The operand value is incremented by one. flags are not affected upon the execution of these instructions.

Mnemonic	Operation	Addressing mode	Number of Bytes	Example	Description
INC Rn	Rn+1 → Rn	Register	1	INC R6	The content of register R6 is incremented by one
INC direct	addr +1 → addr	Direct	2	INC 30H	The content of memory location is incremented by one
INC @Rn	addr(Rn)+1 → addr(Rn)	Indirect	1	INC @R0	The content of memory location whose address is stored in R0 is incremented by one
INC DPTR	DPTR+1 → DPTR	Register	1	INC DPTR	The content of register DPTR is incremented by one

2.1.2.4 Decrement : The decrement is performed using instruction DEC. The operand value is incremented by one. flags are not affected upon the execution of these instructions.

Mnemonic	Operation	Add. mode	No. of Bytes	Example	Description
DEC Rn	Rn-1 → Rn	Register	1	DEC R6	The content of register R6 is decremented by one
DEC direct	addr -1 → addr	Direct	2	DEC 30H	The content of memory location is decremented by one
DEC @Rn	addr(Rn)-1 → addr(Rn)	Indirect	1	DEC @R0	The content of memory location whose address is stored in R0 is decremented by one
DEC DPTR	DPTR-1 → DPTR	Register	1	DEC DPTR	The content of register DPTR is decremented by one

2.1.2.5 Multiplication:

The 8051 microcontroller has a capability of 8 bit multiplication using register A and B. For multiplication register A and B act as source as well as destination. In 8 bit multiplication 16 bit result is generated. The higher byte is stored in register B and lower byte is stored in register A. The over flow flag (OV) flag is set if result of the grater than FFh.

Mnemonic	Operation	Addressing mode	Number of Bytes	Example	Description
MUL AB	AXB → A(0-7)B(0-7)	Register	1	MUL AB	The content with in register A and B are multiplied ad the result is stored in A and B registers.

2.11.2.6 Division :

8051 microcontroller can perform 8 bit division. the content of register A is divided by the content of register B. After division quotient is stored in register A and remainder stored in register B. over flow (OV) flag is set if the content is divided by zero.

Mnemonic	Operation	Addressing mode	Number of Bytes	Example	Description
DIV AB	A ÷ B → A(quotient) B (remainder)	Register	1	DIV AB	The content of register A is divided by content of register B and the result is stored in A and B registers.

2.11.2.7 Decimal Arithmetic:

8051 can perform the BCD arithmetic addition. Using DA A instruction the result within register A is converted into BCD numbers.

Mnemonic	Operation	Addressing mode	Number of Bytes	Example	Description
DA A	A → A(BCD)	Register specific	1	DA A	The content of register A is converted into equivalent BCD numbers

2.11.3 Logical Instructions :

Logical instructions are required for the decision implementation for machine control. for most of the industrial control takes the input switches or sensors depending on which the decision has to implemented on machine. to implement such decisions logical instructions are required. The 8051 instruction set contain logical instructions like ANDing , ORing, XORing, complimenting and compare .

Mnemonic	Operation	Addressing mode	No. of Bytes	Example	Description
ANL A, Rn	A(ANDed)Rn → A	Register	1	ANL A, R4	The content of register R4 is ANDed with the content of register A. The result is stored in register A
ANL A, direct	A(ANDed)addr → A	Direct	2	ANL A, 20H	The content of memory location 20H ANDed with content of register A. The result is stored in register A
ANL A, @Rn	A (ANDed)addr(Rn) → A	Indirect	1	ANL A, @R5	The content of memory location whose address is stored in R5 ANDed with the content of register A. The result is stored in register A
ANL A, #data	A (ANDed) data → A	Immediate	2	ANL A, #10H	The immediate data ANDed with the content of register A. The result is stored in register A
ANL direct,A	A(ANDed)addr → addr	Direct	2	ANL 20H,A	The content of memory location 20H ANDed with content of register A. The result is stored at memory location 20H
ANL direct,#data	A(ANDed)data → addr	Immediate	3	ANL 20H,#40H	The content of memory location 20H ANDed with data 40H. The result is stored at memory location 20H
ORL A, Rn	A(ORed)Rn → A	Register	1	ORL A, R4	The content of register R4 is ORed with the content of register A. The result is stored in register A
ORL A, direct	A(ORed)addr → A	Direct	2	ORL A, 20H	The content of memory location 20H ORed with content of register A. The result is stored in register A
ORL A, @Rn	A (ORed)addr(Rn) → A	Indirect	1	ORL A, @R5	The content of memory location whose address is stored in R5 ORed with the content of register A. The result is stored in register A
ORL A, #data	A (ORed) data → A	Immediate	2	ORL A, #10H	The immediate data 10H ORed with the content of register A. The result is stored in register A
ORL direct,A	A(ORed)addr → addr	Direct	2	ORL 20H,A	The content of memory location 20H ORed with content of register A. The result is stored at memory location 20H
ORL direct,#data	A(ORed)data → addr	Immediate	3	ORL 20H,#40H	The content of memory location 20H ORed with data 40H. The result is stored at memory location 20H
XRL A, Rn	A(XORed)Rn → A	Register	1	XRL A, R4	The content of register R4 is XORed with the content of register A. The result is stored in register A

XRL A, direct	A(XORed)addr → A	Direct	2	XRL A, 20H	The content of memory location 20H XORed with content of register A. The result is stored in register A
XRL A, @Rn	A (XORed)addr(Rn) → A	Indirect	1	XRL A, @R5	The content of memory location whose address is stored in R5 XORed with the content of register A. The result is stored in register A
XRL A, #data	A (XORed) data → A	Immediate	2	XRL A, #10H	The immediate data 10H XORed with the content of register A. The result is stored in register A
XRL direct,A	A(XORed)addr → addr	Direct	2	XRL 20H,A	The content of memory location 20H XORed with content of register A. The result is stored at memory location 20H
XRL direct,#data	A(XORed)data → addr	Immediate	3	XRL 20H,#40H	The content of memory location 20H XORed with data 40H. The result is stored at memory location 20H
CLR A	A=0	Register specific	1	CLR A	This instruction clears the content of register A
CPL A	$A=\overline{A}$	Register specific	1	CPL A	This instruction will compliments the content of register A
RL A	$A_{n+1}=A_n$	Register specific	1	RL A	This instruction will shift the content of register A to left by one bit
RLC A	$A_{n+1} \rightarrow A_n$ $A_0 \rightarrow CY$ $A_7 \rightarrow CY$	Register specific	1	RL A	This instruction will shift the content of register A to left by one bit through carry.
RR A	$A_n \rightarrow A_{n+1}$	Register specific	1	RL A	This instruction will shift the content of register A to left by one bit
RLC A	$A_n \rightarrow A_{n+1}$ $CY \rightarrow A_0$ $CY \rightarrow A_7$	Register specific	1	RL A	This instruction will shift the content of register A to left by one bit through carry.
SWAP A	A(0-3) ⇄ A(4-7)	Register specific	1	SWAP A	This instruction exchange the lower nibble and higher nibble of register A

2.11.4 Bit Manipulation Instructions

Some variables content the single bit information like switches and LEDs. 8051 microcontroller contain the few bit addressable register and a bit addressable RAM. Using bit manipulation single bit can be modify or processed.

Mnemonic	Operation	Addressing mode	No. of Bytes	Example	Description
CLR Bit	Bit=0	Register/ direct if operated on carry bit	1or 2	CLR P2.3	Clear the bit P2.3
SETB Bit	Bit=1	Register/ direct if operated on carry bit	1or 2	SETB P1.0	Set the bit P1.0
CPL Bit	Bit=$\overline{Bit}$	Register/ direct if operated on carry bit	1 or 2	CPL 3.1	Compliment the bit P3.1
ANL C,Bit	CY(ANDed)Bit → CY	Direct	2	ANL C,ACC.3	This instruction will perform logically ANDing of carry bit and the bit specified within instruction. The result is stored in the carry bit
ORL C,Bit	CY(ORed)Bit → CY	Direct	2	ORL C,ACC.7	This instruction will perform logically ORing of carry bit and the bit specified within instruction. The result is stored in the carry bit
MOV Bit,C	CY → Bit	Direct	2	MOV ACC.2,C	The carry bit is copied on register A bit number two
MOV C,Bit	Bit → CY	Direct	2	MOV C,ACC.1	The register A bit number 1 copied to the carry bit

Single bit Jump Instructions

Mnemonics	operation	addressing mode	No.of bytes	Example	Description
JB bit, rel address	Jump to target	Direct	3	JB P1.5, HERE	jump if bit P1.5 is set
JNB bit,rel address	jump to target	Direct	3	JNB ACC.0, NEXT	jump if bit ACC.0 not set
JBC bit, rel address	jump to target	Direct	3	JBC ACC.7, NEXT	jump if bit ACC.7is set and clear bit
JC, rel address	jump to target	Direct	3	JC NEXT	Jump if Carry is set
JNC, rel address	jump to target	Direct	3	JNC NEXT	Jump if Carry not set

2.11.5 Branching and Looping Instructions:

Branching instructions in 8051

Program branching plays a very important role in software development. we know that a microcontroller is a device with the duty of continuously fetching and executing instructions. This may be designated as a sequential execution of instructions. However, many times the processor is to branch from one sequence to another, known as conditional branching. There may be some cases of unconditional jumps also. There are two types of JUMP instructions:

i. Unconditional JUMP instructions
ii. conditional JUMP instructions

Unconditional JUMP instructions

Mnemonic	Operation	No. of Bytes	Range of address	Example	Description
SJMP rel	PC=PC + 2 PC=PC + rel	2	+127 to -128 bytes	SJMP LCD	The program is unconditionally to the address indicated with relative jump bytes.
AJMP 11 bit addr	PC=PC+2 PC= 11 bit addr	2	2KB	AJMP ADD	Upon execution of this instruction the program counter is incremented by two and PC loaded with new address i.e. with label ADD
LJMP 16 bit addr	PC= 16 bit addr	3	64KB	LJMP DELAY	Upon execution of this instruction the program counter loaded with n16 bit address i.e. with label DELAY
JMP @A+DPTR	PC=A+DPTR	1	64KB	LJMP DELAY	The content of register A and content of DPTR are added to generate new address. PC is loaded unconditionally with this address.

Unconditional JUMP instructions

Mnemonic	Operation	No. of Bytes	Range of address	Example	Description
JZ rel	If A=0 or Z=1 PC=PC + rel Else PC=PC+2	2	+127 to -128 bytes	JZ SUB	If the zero flag is set then the program counter loaded with new address calculated relatively. otherwise the program counter(PC) starts executing next instruction.
JNZ rel	If A=non zero or Z=0 PC=PC + rel Else PC=PC+2	2	+127 to -128 bytes	JZ L1	If the zero flag is reset then the program counter loaded with new address calculated relatively. otherwise the program counter(PC) starts executing next instruction.
JC rel	If CY=1 PC=PC + rel Else PC=PC+2	2	+127 to -128 bytes	JC SUB	If the carry flag is set then the program counter loaded with new address calculated relatively. otherwise the program counter(PC) starts executing next instruction.

JNC rel	If CY=0 PC=PC + rel Else PC=PC+2	2	+127 to -128 bytes	JC L1	If the carry flag is reset then the program counter loaded with new address calculated relatively. otherwise the program counter(PC) starts executing next instruction.
CJNE A, direct, rel	If A≠ direct then PC=PC + rel Else PC=PC+3 (next instruction)	3	+127 to -128 byte	CJNE A,30H,HERE	Compare the content of register A and the data at the address 30H. if they are not equl then jump to relative address HERE else program counter(PC) starts executing next instruction
CJNE A, #data, rel	If A≠ data then PC=PC + rel Else PC=PC+3 (next instruction)	3	+127 to -128 byte	CJNE A,#50H,H1	Compare the content of register A and immediate data 50H. if they are not equal then jump to relative address H1 else program counter(PC) starts executing next instruction
CJNE Rn, #data, rel	If Rn≠ data then PC=PC + rel Else PC=PC+3 (next instruction)	3	+127 to -128 byte	CJNE R0,#60H,H1	Compare the content of register R0 and immediate data 60H. if they are not equal then jump to relative address H1 else program counter(PC) starts executing next instruction
CJNE @Rn, #data, rel	If @Rn≠ data then PC=PC + rel Else PC=PC+3 (next instruction)	3	+127 to -128 byte	CJNE @R1,#30H,H3	Compare the content memory location whose address is pointed by register R1 and immediate data 30H. if they are not equal then jump to relative address H3 else program counter(PC) starts executing next instruction
DJNZ Rn,rel	Rn=Rn-1if Rn≠0 PC=PC+rel	2 or 3	+127 to -128 byte	DJNZ R5,ADD	Decrement the value of register R5 , if not equal to zero jump to relative address ADD. Else program counter(PC) starts executing next instruction

Call and Return instructions in 8051

In some cases, a microcontroller needs to perform the same tasks multiple numbers of times across the program, such as generating a delay. A subroutine is responsible for performing these repetitive tasks. Using subroutines, saves memory and makes the program more efficient.

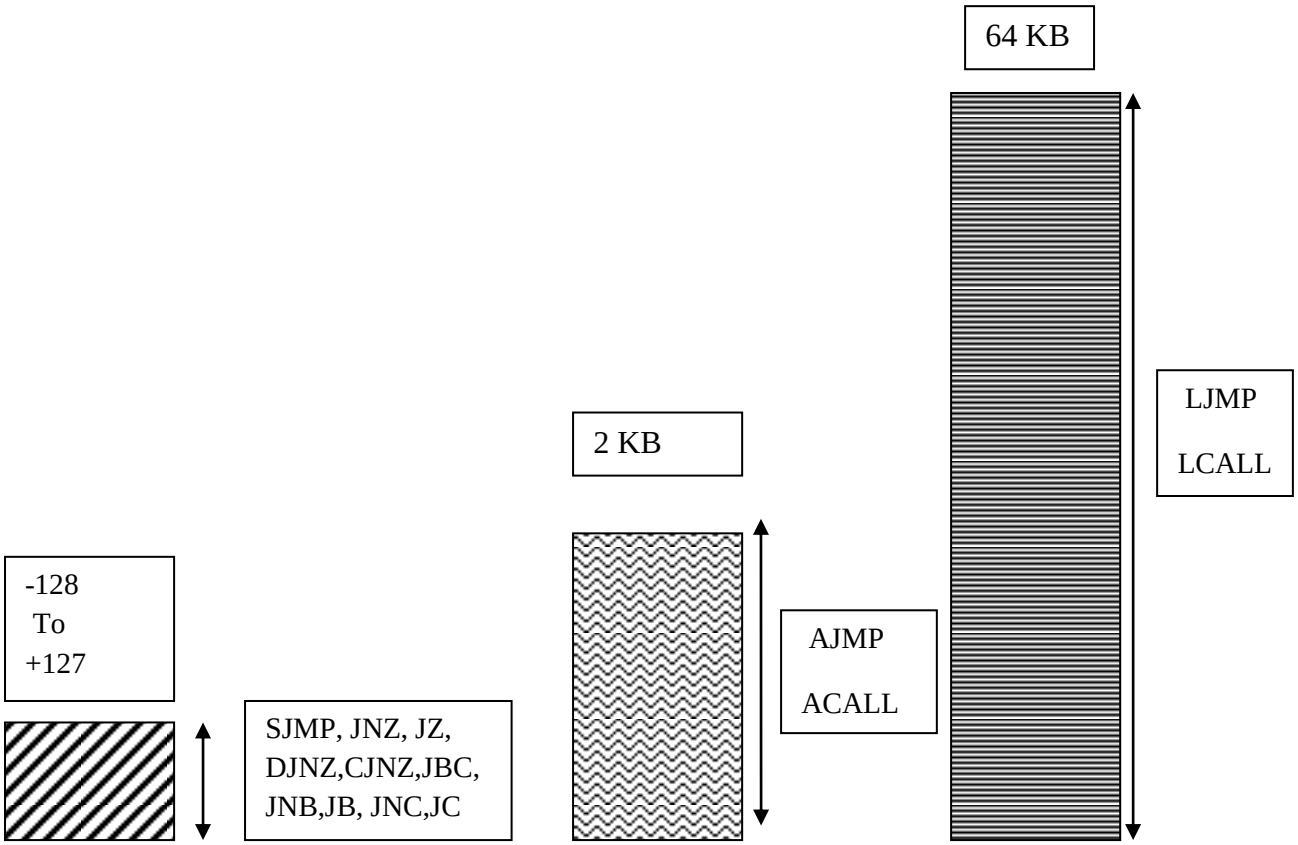
Instead of repeating the same few lines of code for some task we need to execute multiple times, we can just write it once and give it a label. Every time we need those lines to execute, just use the label to jump to the area where we had stored the labeled code. In essence, a subroutine is like a function, and it is placed at a different memory location than the program memory. The CALL(call)

instruction is used to transfer the control from the presently executing program code to the subroutine, and the RET (return) instruction is used to return the control to the program code.

Mnemonic	Operation	No. of Bytes	Range of address	Example	Description
ACALL 11 bit addr	PC=PC+2 SP=SP+1 SP=PC_{0-7} SP=SP+1 SP=PC_{8-15} PC= 11 bit addr	2	2KB	ACALL DELAY	1. program counter is incremented by two 2. stack pointer is incremented by one 3. lower byte of the stack pointer is pushed on the SP 4. stack pointer incremented by one and the higher byte is pushed on the stack 5. program counter is loaded with 11 bit address
LCALL 16 bit addr	PC=PC+3 SP=SP+1 SP=PC_{0-7} SP=SP+1 SP=PC_{8-15} PC= 16 bit addr	3	64KB	LCALL LCD	1. program counter is incremented by three 2. stack pointer is incremented by one 3. lower byte of the stack pointer is pushed on the SP 4. stack pointer incremented by one and the higher byte is pushed on the stack 5. program counter is loaded with 16 bit address
RET	PC_{8-15} =SP SP=SP-1 PC_{0-7}= SP SP=SP-1	1	-----	RET	1. the higher byte is pop out of the stack 2. stack pointer is decremented by one 3. lower byte is pop out of the stack pointer 4. stack pointer is decremented by one
RETI	PC_{8-15} =SP SP=SP-1 PC_{0-7}= SP SP=SP-1	1	--------	RETI	1. the higher byte is pop out of the stack 2. stack pointer is decremented by one 3. lower byte is pop out of the stack pointer 4. stack pointer is decremented by one

2.11.5.1 Jump and Call Instruction Range:

Jump and call instructions come with a one of three ranges for the address. The instruction with the relative range can address from +127 to -128 bytes from the present address. The instruction with absolute has range of 2K bytes from the present address and instruction with long can cover total i.e. 64K bytes of memory space.



2.11.6 Addressing modes:-

The ways to specify the address of data in instruction is called addressing modes. The data could be in a register, or in memory, or be provided as an immediate value. There are four addressing modes.

1) Immediate addressing mode
2) Direct addressing mode
3) Register addressing modes
4) Indirect addressing modes (indirect register addressing mode)
5) Indexed addressing mode

2.11.6.1 Immediate addressing mode:-

In this instruction data is directly written in the instruction. i.e. data is a part of instruction.

e.g. 1)MOV A, #data (MOV A, #55)→ copy specified data into accumulator.

2)MOV Rn, #data(MOV R0,#22H)→copy specified data into Register(R0-R7).

3)ADD A, #data→ Adds the specified data with content of accumulator.

4)SUBB A, #data→ Subtract the specified data with the content of accumulator.

5)ORL A, #data→ The specified data is ORed with the content of accumulator.

6)MOV DPTR,#16bit data→ Copy the 16-bit data into DPTR. e.g. MOV DPTR, #9006.

2.11.6.2 Direct Addressing mode:-

All 128 bytes of internal RAM & SFR`S can be addressed using assigned values. In this instruction address of operands are directly specified.

e.g. 1)MOV A, add→ Copy data from location specified by given address into Accumulator.

a)MOV A, 80H ;Read the content of port P0(add 80H) & copy them into Acc.

b)MOV 80H, A ;write into port P0.

c)MOV PSW, A (move D0,E0); Move the content of Acc. Into flags.

2)MOV Rn, add→ Copy data from locations specified by given address into register Rn.

e.g.MOV R2,12H ; copy data from location specified by 12h into R2.

3)Add Address→ Adds the data from the location specified by given address with content of accumulator.

4)XCH A, Address→ Exchange the data from the location specified by given address with the Content of accumulator**.**

5)XRL A, address→ The data from the location specified by given address is XORed with the content of accumulator.

This address made is generally fast. It is quickly accessible since the value is stored in internal RAM. It is more flexible than immediate addressing.

2.11.6.3 Register Addressing mode:-

In register addressing mode, the register A(accumulator), R0-R7 are specified in the instruction itself. (R0-R7) are used in currently selected bank.

e.g. 1)MOV A, R0; Move data from register R0 to A A←R If R0=55H then A=55H

2)MOV A, Rn ; A←Rn.

3)ADD A, Rn ; Adds the content of register Rn with Accumulator.

4)SUBB A, Rn; Subtracts the contents of register Rn from Acc with Borrow.

5)XRL A, Rn ; The contents of register Rn are XOred with accumulator content.

6)XCH A, Rn ; The content of register Rn & Acc are exchanged.

7)ANL A, Rn ; The content of register Rn are ANDed with accumulator.

8)ADDC A, Rn; The content of Rn are added into Accumulator with carry movement of data between register. Rn is not allowed i.e. MOV R5, R6 is not valid (only R0,R1 are used as pointer register)

2.11.6.4 Indirect (Register)addressing mode:-

In this addressing mode, register is used to hold the actual address of operand or data. The register itself is not the address. These instructions uses R0, R1 as data pointer and register, but R2-R7 are not allowed to hold address.

e.g. 1)MOV A, @Ro ; Move to accumulator content of memory location pointed by reg. R0

2)MOV @R1, A; move the content of the accumulator into memory location pointed by register R1.

3)ADD A,@R0; Adds the content of memory location pointed b register R0 with content of accumulator .

4)SUB A,@R1; Subtracts the content of memory location pointed out by register R1 from contents of accumulator.

5)INC @R0; increment the contents of memory location pointed out by register R0.

6)DEC @R1;Decrement the contents of memory location pointed by regsiter.

7)ANL A,@R; The content of memory location pointed out by reg R1 is ANDed with content of accumulator. But, MOV A,@R3;This is not valid.

2.11.6.5 Indexed addressing mode(External addressing mode):-

It is widely used in accessing data elements of look-up table entries located in the program ROM.

1) MOVC A,@A+DPTR : Here MOVC, "C" means code. The contents of A are added to the 16-bit register DPTR to form the 16-bit address from which data is copied into A.
2) MOVC A,@A+PC : The contents of A are added to the 16-bit register PC to form the 16-bit address from which data is copied into A.

Exercise: Solve the following objectives (1 mark each)

1. _____instruction has Immediate addressing mode.

   i) MOV A, R7 ii) MOV A,@R1 iii) ADD A, #45H iv) ADD A,48H
2. _____instruction has Direct addressing mode.

   i) MOV A, R7 ii) MOV A,@R1 iii) ADD A, #45H iv) ADD A,48H
3. _____instruction has Register addressing mode.

   i) MOV A, R7 ii) MOV A,@R1 iii) ADD A, #45H iv) ADD A,48H
4. _____instruction has Register Indirect addressing mode.

   i) MOV A, R7 ii) MOV A,@R1 iii) ADD A, #45H iv) ADD A,48H
5. _____is an instruction from data transfer group.

   i) MOV ii) SETB iii) ADD iv) DA A
6. _____ is an instruction from Arithmetic group.

   i) MOV ii) SETB iii) ADD iv) DJNZ
7. _____ is an instruction from Boolean or Bit Manipulation Instruction group.

   i) MOV ii) SETB iii) ADD iv) DA A

8. _____ is/are an instruction/s from Boolean or Bit Manipulation Instruction group.

   i) CLR bit ii) MOV bit,bit iii) JB bit iv) all of the above

9. _____ is/are Single bit Jump Instruction/s.

   i) JC ii) JNB iii) JB iv) all of the above

10. _____ is/are instruction/s rotate the contents of Accumulator right through carry flag.

    i) RR A ii) RL A iii) RRC A iv) RLC A

11. _____ is an instruction from Branching group.

    i) MOV ii) SETB iii) DJNZ iv) DA A

12. _____ is instructions used to BCD adjustment of accumulator.

    i) MOV ii) SETB iii) ADD iv) DA A

13. _____ instruction/s is/are related to stack operation.

    i) PUSH ii) POP iii) RET iv) all of the above

14. If A=25H, then after execution of SWAP A instruction, the contents of A becomes _____

    i) 05H ii) 20H iii) 52H iv) 25H

15. The range of short jump instruction is _____

    i) 0 to 255 ii) -128 to 127 iii) 0 to 2K iv) 0 to 65535

16. The range of absolute jump instruction is _____

    i) 0 to 255 ii) -128 to 127 iii) 0 to 2K iv) 0 to 65535

17. The range of Long jump instruction is _____

    i) 0 to 255 ii) -128 to 127 iii) 0 to 2K iv) 0 to 65535

18. _____ instruction is used to exchange the 8-bit contents between source location and destination location _____

    i) SWAP ii) MOV iii) XCH iv) XCHD

19. _____is Three byte instruction.

    i) AJMP ii) JNZ iii) SJMP iv) LJMP

20. _____is/are Two byte instruction/s.

    i) AJMP ii) JNZ iii) SJMP iv) all of these

21. _____ is/are Short relative jump instruction/s.

    i) SJMP ii) DJNZ iii) CJNE iv) all of these

22. _____ is/are Unconditional Short relative jump instruction/s.

i) SJMP ii) DJNZ iii) CJNE iv) all of these

23. _____ is/are Conditional Short relative jump instruction/s.

i) JC ii) JB iii) CJNE iv) all of these

24. _____ is/are illegal CALL instruction/s.

i) SCALL ii) ACALL iii) LCALL iv) all of these

25. ----------instruction is used at end of the subroutine.

i) PUSH ii) POP iii) RET iv) ACALL

26. ----------instruction is used to move the contents of External Program memory into ACC.

i) MOVX A,@DPTR ii) MOVX A,@R1 iii) MOV A,@R1 iv) MOVC A,@DPTR

27. _____ instruction is used to exchange the lower 4-bits(nibble) contents between source location and destination location _____

i) SWAP ii) MOV iii) XCH iv) XCHD

28. After execution of MUL AB instruction the lower byte of the result is stored in _____ register and higher byte of result is stored in _____ register.

i) B and A ii) A and B iii) A and PSW iv) B and PSW

29. _____ instruction is used to BCD adjustment of Accumulator .

a) MOV b) DJNZ c) DAA d) SETB

30. _____is three byte instruction.

a) SJMP b) JNZ c) AJMP d) LJMP

31. _____instruction has Immediate addressing mode.

i) MOV A, R1 ii) MOV A,@R0 iii) ADD A, #55H iv) ADD A,23H

32. If A=37H, then after execution of SWAP A instruction, the contents of A becomes __ i) 30 H

ii) 07 H iii) 70H iv) 73 H

33. --------- is the single bit instruction.

i) ADD ii) NOP iii) JB iv) ACALL

34. ADD instruction is ----------bytes instruction

a) 1 b) 2 c) 3 d) 4

35. ----- instruction is a two byte instruction.

a) ANL A, direct b) ANL A,R0 c) MUL AB d) DIV AB

36. After multiplication the higher byte is stored in ----------register

a) Register A b) Register B c) Register R0 d) Register R7

37. After division the remainder is stored at ----------register

a) Register A b) Register B c) Register R0 d) Register R7

38. following is the single bit instruction

a) JB b) ADD c) SUBB d) NOP

39. upon execution of ----------instruction only program counter is modified

a) JB b) ADD c) SUBB d) NOP

Solve the following for 10 marks

Q1. Describe the different addressing modes in 8051 and give the example of each?

Q2. Classify the instruction set according to functions and explain each with two suitable examples.

Q3. Describe the logical and arithmetic instructions in 8051?

Q4.What is addressing mode? What are the different addressing modes? Give the example of each addressing mode

Q5. Describe CALL and JUMP Instructions? What is the difference between the CALL and JUMP?

Q6.What is the difference between the absolute, long and short Jump?

Solve the following for 5 marks

Q1. Describe any five logical instructions

Q2. Describe any five arithmetic instructions

Q3. What is the difference between the CALL and JUMP?

Q4. Explain the following instructions

1. MOVX A,@DPTR
2. CJNE @Rn, #data, rel
3. DJNZ

Q5. Explain the following instructions

(a) ANL
(b) ORL
(c) CPL
(d) CLR
(e) ADDC

Q6. Describe any five Bit manipulation instructions.

Q7. Describe any five Bit manipulation jump instructions.