ELECTRONICS-DSC -1005 D Semester: IV Electronics-Paper- IV Advance Communication and Microcontroller 8051

Section II: Microcontroller 8051

Unit 2: Instruction Set of 8051

Classification of instruction sets, Addressing modes .

Instruction set of 8051: data transfer, arithmetic, Logical, Jump, call,

Boolean instructions.

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

Immediate addressing mode:-

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

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.

Direct Addressing mode:-

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

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 the 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.

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)

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.

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.

Indexed addressing mode(External addressing mode):-

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

•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.

•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.

2.1 Addressing modes

When controller executes an instruction, it perform the specific function on data. The different ways to specify the address of data in an instruction are called as addressing modes.

2.1.1 Types of Addressing Modes

- Direct addressing mode
- Register Indirect addressing mode
- Register addressing mode
- Immediate addressing mode
- Indexed / External addressing mode

•**Direct addressing mode**

In this mode the operand is specified by an 8 bit address field in the instruction .Internal RAM and all SFRs can be accessed by direct addressing mode. The internal memory address from 00H to 7FH and the SFRs can access from 80H to FFH

e.g. **MOV A, 60H ;** copy the content of memory location 60H to register A

•**Register Indirect addressing mode**

In this addressing mode, the instruction specifies the register which contains the address of an operand. The registers from the register bank are used as a memory pointer . The @ sign indicated the register acts as a pointer to memory location. Only R0 and R1 registers are used as pointers for selected bank.

e.g. **MOV A,@R0** ; copy the content of the memory location whose address id pointed by register R0

- **Register addressing mode**

In this addressing mode the operand is copied from one register to another register. In this addressing mode both source and destination are registers.

e.g. **MOV A, R0** ; copy the content of R0 to register A

•**Immediate addressing mode**

This method is the simplest method to transfer a data. In this addressing mode the operand/data is the part of instruction. The # sign indicates that the data followed immediate by operand.

e.g. **MOV A,#20H** ; copy the immediate data to register A

•**Indexed/External addressing mode**

Using this addressing mode only external program memory can be accessed. This addressing mode is used to read the lookup table. Either the DPTR or PC is used as a memory pointer.

e.g**. MOVC A,@A+DPTR** ; copy the content of memory location to register A pointed by DPTR register

The 8051 instruction set is divided into different groups

1. Data Transfer Instructions
2. Arithmetic Instructions
3. Logical Instructions
4. Bit Processing Instructions
5. Program Branching Instructions

1. Data Transfer Instructions

2.2 Data Transfer Instructions

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

Mnemonic	Operation	Addressing mode	Number 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. Arithmetic Instructions

Mnemonic	Operation	Addressing mode	Number 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

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

2.3.3. 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

2.3.4. 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	Addressing mode	Number 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

2.3.5. 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.

2.3.6. 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.

2.3.7. 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

3. Logical Instructions :

Mnemonic	Operation	Addressing mode	Number 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

4. Bit Manipulation Instructions

2.5. Bit Manipulation Instructions

Mnemonic	Operation	Addressing mode	Number 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

BOOLEAN VARIABLE MANIPULATION

CLR	C	Clear carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to carry	2	24
ANL	C,/bit	AND complement of direct bit to carry	2	24
ORL	C,bit	OR direct bit to carry	2	24
ORL	C,/bit	OR complement of direct bit to carry	2	24
MOV	C,bit	Move direct bit to carry	2	12
MOV	bit,C	Move carry to direct bit	2	24
JC	rel	Jump if carry is set	2	24
JNC	rel	Jump if carry not set	2	24
JB	rel	Jump if direct bit is set	3	24
JNB	rel	Jump if direct bit is not set	3	24
JBC	bit,rel	Jump if direct bit is set and clear bit	3	24

5. Program Branching Instructions:

Branching instructions are classified into the two groups:

1. Jump instructions
2. Subroutine instructions

Jump instructions

a. Short relative jump

b. Absolute jump

c. Long jump

Short relative jump

Instruction	Description	Conditions for jump
SJMP	Unconditional short jump e.g. SJMP here	-
JC	Jump if carry e.g JC here	CY=1
JNC	Jump if not carry e.g JNC here	CY=0
JZ	Jump if zero e.g JZ here	A=0
JNZ	Jump if not zero JBC P1.5, here	A≠0
JB	Jump if bit e.g JB P1.5, here	Bit =1
JNB	Jump if not bit e.g JNB P1.5, here	Bit=0
JBC	Jump if bit and clear the bit e.g JBC P1.5, here	Bit=1
CJNE	compares the first two operands and jumps to the specified destination if their values are not equal. If the values are the same, execution continues next instruction. e.g. CJNE A,#25H, here	SFR or direct or indirect address becomes equal
DJNZ	decrements the byte indicated by the first operand and, if the resulting value is not zero, jumps to the address specified in the second operand. e.g. DJNZ R0, here	SFR or immediate data or direct or indirect address becomes zero

Absolute jump:

AJMP <11 bit address>

The Absolute jump is two byte unconditional jump. It jumps within page of 2K byte. In 8051, 64 Kbyte of program memory space is divided into 32 pages of 2Kbyte each.

Long Jump : (long jump)

LJMP <16 bit address>

It is used to access the entire program memory from 0000H to FFFFH. It is a 3-byte instruction (except for JMP @ A+DPTR). First byte is the op-code and second and third bytes represent the 16-bit target address which is any memory location from 0000 to FFFFH. **eg: LJMP 7000H**

Subroutine Instructions

In the 8051 a subroutines are handled by CALL and RET instructions. The main difference between CALL and JUMP instructions is that CALL has return path and jump has no return path. When CALL instructions are executed then the content of the PC pointed to the next of the CALL is stored into the stack pointer. During RET instructions it is loaded into the PC.

There are two types of CALL instructions

a. Absolute call : ACALL
b. Long call : LCALL

Absolute call

ACALL <11 bit address>

The Absolute CALL is two byte call instruction. It call within page of 2K byte. In 8051, 64 Kbyte of program memory space is divided into 32 pages of 2Kbyte each.

Long CALL

LCALL <16 bit address>

This is long call instruction which unconditionally calls the subroutine located at the indicated 16 bit address. This is a 3 byte instruction. No flags are affected. e.g. LCALL 7000H

RET instruction

A RET instruction pops top two content from the stack memory and loaded into PC. After execution of RET instruction a main program is executed. e.g. RET

Jump on bit conditions

Compare bytes and jump if *not* equal

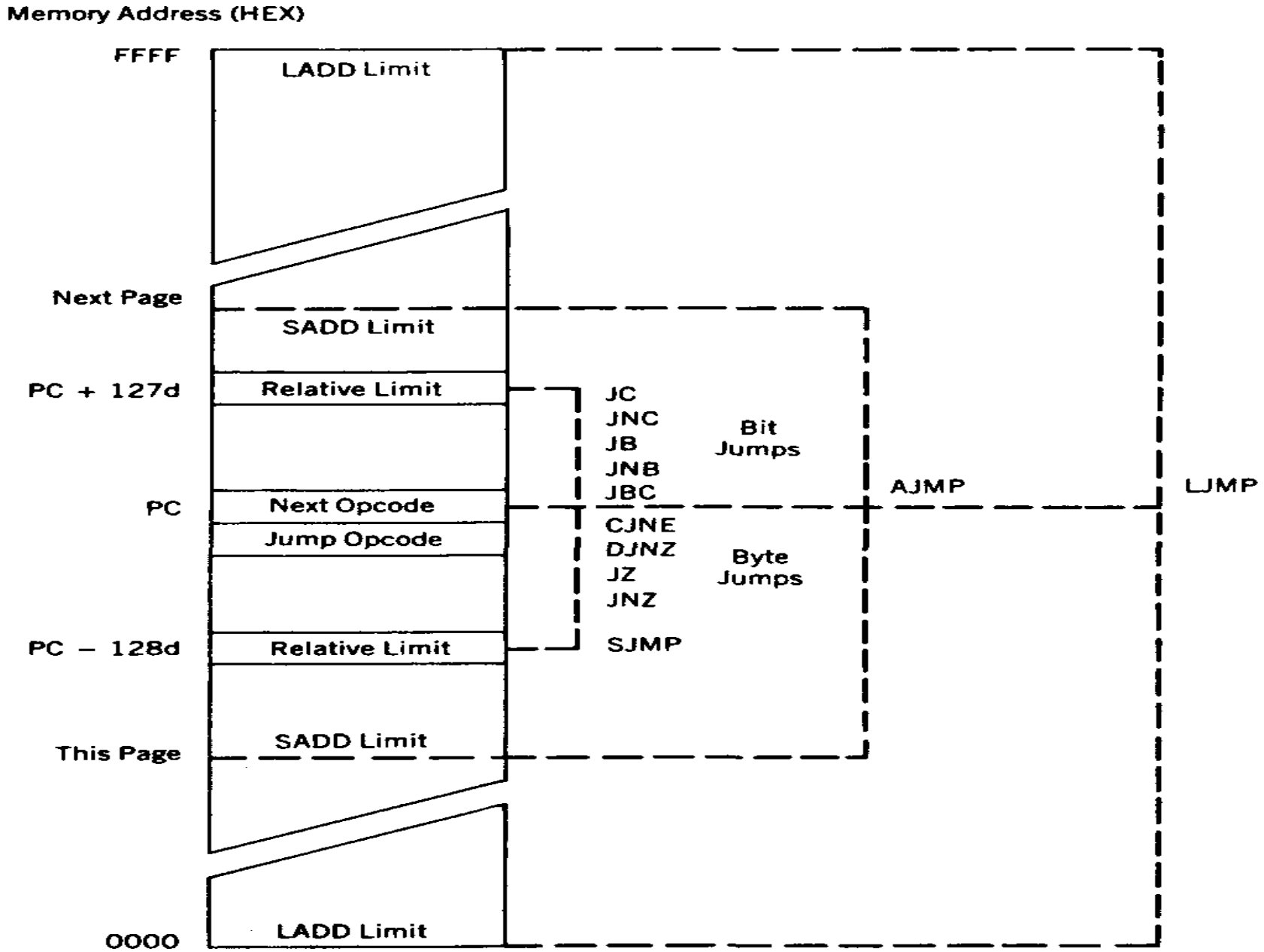
Decrement byte and jump if zero

Jump unconditionally

Call a subroutine

Return from a subroutine

Jump Instruction Ranges



Bit Jumps

Mnemonic	Operation
JC radd	Jump relative if the carry flag is set to 1
JNC radd	Jump relative if the carry flag is reset to 0
JB b,radd	Jump relative if addressable bit is set to 1
JNB b,radd	Jump relative if addressable bit is reset to 0
JBC b,radd	Jump relative if addressable bit is set, and clear the addressable bit to 0

```
ADDRESS   MNEMONIC      COMMENT
LOOP:     MOV A,#10h    ;A = 10h
          MOV R0,A      ;R0 = 10h
ADDA:     ADD A,R0      ;add R0 to A
          JNC ADDA      ;if the carry flag is 0, then no carry is
                        ;true; jump to address ADDA; jump until A
                        ;is F0h; the C flag is set to
                        ;1 on the next ADD and no carry is
                        ;false; do the next instruction
```

Byte Jumps

Mnemonic	Operation
CJNE A,add,radd	Compare the contents of the A register with the contents of the direct address; if they are *not* equal, then jump to the relative address; set the carry flag to 1 if A is less than the contents of the direct address; otherwise, set the carry flag to 0
CJNE A,#n,radd	Compare the contents of the A register with the immediate number n; if they are *not* equal, then jump to the relative address; set the carry flag to 1 if A is less than the number; otherwise, set the carry flag to 0
CJNE Rn,#n,radd	Compare the contents of register Rn with the immediate number n; if they are *not* equal, then jump to the relative address; set the carry flag to 1 if Rn is less than the number; otherwise, set the carry flag to 0
CJNE @Rp,#n,radd	Compare the contents of the address contained in register Rp to the number n; if they are *not* equal, then jump to the relative address; set the carry flag to 1 if the contents of the address in Rp are less than the number; otherwise, set the carry flag to 0

DJNZ Rn,radd	Decrement register Rn by 1 and jump to the relative address if the result is *not* zero; no flags are affected
DJNZ add,radd	Decrement the direct address by 1 and jump to the relative address if the result is *not* 0; no flags are affected unless the direct address is the PSW
JZ radd	Jump to the relative address if A is 0; the flags and the A register are not changed
JNZ radd	Jump to the relative address if A is *not* 0; the flags and the A register are not changed

Unconditional Jumps

Mnemonic	Operation
JMP @A+DPTR	Jump to the address formed by adding A to the DPTR; this is an unconditional jump and will always be done; the address can be anywhere in program memory; A, the DPTR, and the flags are unchanged
AJMP sadd	Jump to absolute short range address sadd; this is an unconditional jump and is always taken; no flags are affected
LJMP ladd	Jump to absolute long range address ladd; this is an unconditional jump and is always taken; no flags are affected
SJMP radd	Jump to relative address radd; this is an unconditional jump and is always taken; no flags are affected
NOP	Do nothing and go to the next instruction; NOP (no operation) is used to waste time in a software timing loop; or to leave room in a program for later additions; no flags are affected

Calls and Returns

Mnemonic	Operation
ACALL sadd	Call the subroutine located on the same page as the address of the opcode immediately following the ACALL instruction; push the address of the instruction immediately after the call on the stack
LCALL ladd	Call the subroutine located anywhere in program memory space; push the address of the instruction immediately following the call on the stack
RET	Pop two bytes from the stack into the program counter
RETI	Pop two bytes from the stack into the program counter and reset the interrupt enable flip-flops

8051 conditional jump instructions

Instructions	Actions
JZ	Jump if A = 0
JNZ	Jump if A ≠ 0
DJNZ	Decrement and Jump if A ≠ 0
CJNE A,byte	Jump if A ≠ byte
CJNE reg,#data	Jump if byte ≠ #data
JC	Jump if CY = 1
JNC	Jump if CY = 0
JB	Jump if bit = 1
JNB	Jump if bit = 0
JBC	Jump if bit = 1 and clear bit

- All conditional jumps are short jumps
  - The address of the target must within -128 to +127 bytes of the contents of PC

Mnemonic	Operation	Number 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 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 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 5.