Module: 6

22/8/19

Addressing Modes of 8051

→ The way for fetch operand (find the address of operand)

1. Immediate
2. Register Direct
3. Direct
4. Register indirect
5. Indexed.

1. Immediate.

the data mov to A

Mov A, #65H

A → Destination

65H → Source

Mov dptr, #4100H

4100H → Address

→ dptr points to external memory

2. Register Direct

Mov A, R_1

{Mov R_1, R_2 is not possible
doesn't support a generalpurposes}

3. Direct

Mov A, 03H

03H → Address of Register 3 in Bank 0

Mov 03H, A

4. Register Indirect

Mov A, @R_0

@ → to indicate indirect

Mov A, @R_1

Mov A,@dptr → taking data from exte. mem

→ It doesnot support $R_2 - R_7$, Only R_0 & R_1

5. Indexed

offset

Movc A, @A+dptr (Content of ~~tw~~ data from offset & external)

Movc A,@A+PC

↓ Move the code

Instruction set of 8051

i) Data transfer

ii) Arithmetic

iii) Logical

iv) Bitwise manipulation

v) Program branching.

i) **Data transfer**: Non of the flags are affected

• **Mov**

It can be use in the following forms:

1) Register A as destination

```
Mov A,#25H
Mov A,R6
Mov A,30H
Mov A,@R0
```

2) Register A as source

```
Mov R1,A
Mov 31H,A
Mov @R1,A
```

3) Rn as destination ($R_n \rightarrow R_0 - R_7$)

Mov Rn,A   Mov Rn,#25H

Mov Rn,03H

[Mov R1, R2 ✗ Register with reg. is not possible]

4) Direct address as destination

Mov 05H, 02H
Mov 05H, #d5H

All are possible (Address, Register, immediate, accumulator, reg. indirect)

5) Indirect address as destination

Mov @R1,#25H $\rightarrow$ direct address

Mov @R0, A $\rightarrow$ Accumulator

Mov P3.4 , C $\rightarrow$ carry
$\rightarrow$ Port 3 . 4th pin

6) Mov DPTR,#16 bit value

Eg: Move dptr, # 4100H

• Movc – Move code

This opcode means move a code byte. ie, it moves the byte of data loaded in program code (ROM) into register A.

Movc A,@ A+DPTR
Movc A, @ A+PC

• Movx – Move to external

Movx A, @dptr
Movx @DPTR, A
} (starting at pgm) 16 bit

Movx A,@R1
Movx @R1,A
} 8 bit

• **PUSH direct** → PUSH direct address

General form: PUSH E0H

E0 → Address of Accumulator

SP=SP+1

[SP] ← A

PUSH 03H ;Reg 3 of Bank 0.

• **POP direct**

General form: POP E0H

~~SP=SP~~ A ← [SP]

SP = SP-1

POP 03H

• **XCH** (Exchange) XCHD: Exchange digits.

XCH A, R_4 XCH A, 30H   XCH A, @R_1

XCHD A, @R_1 → Exchange the lower nibble in A & lower nibble in R_1

A = 1101 $\underline{1000}$

@R_1 = 1000 $\underline{1100}$

ii) ARITHMETIC INSTRUCTIONS

• **ADD** flags affected are OV, AC, CY, & P flags will be set.

ADD A,#45H   ADD A,R_3 ADD A,30H

ADD A, @R_1

• **ADDC** - ADD with Carry - Except parity all the flags will be set. (CY=0)

• ADDC A, #0F2H

A+0F2+CY

• **SUB B** - Subtract with borrow (Carry = borrow) ↳ Except parity

SUBB A, #25H

A-25H-CY

• **INC** - Increment

INC A

INC DPTR

None of the flags are set

• **DEC** - Decrement

DEC A

~~DEC~~ DEC ~~DPTR~~

• **MUL AB** - Result BA → lower, B ↓ upper → OV is set if result > FF & CY will be clear

• **DIV AB** A → Qu. & B - Reminder → OV is set if B is 0.

• **DA** · Decimal-adjust accumulator after addition. Carry will be set

iii) LOGIICAL INSTRUCTIONS

27/08/19

- ANL (Logical AND) - Non of The flags are affected

- ANL dest, source (Result will be on destination)

Eg: ANL A,#09H

ANL C,/P2·2

(Logical AND operation with C and compliment of P2·2)

ANL A,R1

ANL A,30H

Eg:
```
 0011 1001
 0000 1001
 ---------
 0000 1001
    0 9
```

- ORL (Logical OR) - Non of The flags

- ORL A,# 39H

- XRL (Logical,XOR) - Non of the flags.

- XRL A,# 39H

- CLR (clear) - Non of the flags will be affected

- CLR A , Clear A

Application : Before do SUBB

- CPL (complement) - Non of The flags

CPL A

Eg: A 1100 0001 (C 1)

CPL A

0011 1110 (3 E)

- RL (Rotate Left)

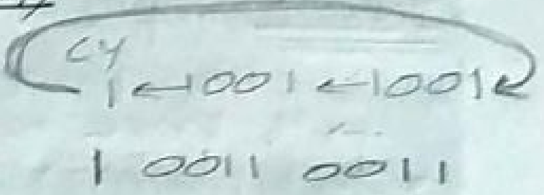
Mov A,#69H

Eg: A= 0101 1001

1011 0010

- RLC (Rotate left with carry)

- RLC A

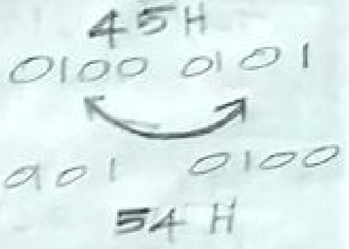


- RR (Rotate Right)

  RR A

- RRC (Rotate Right with Carry)

  RRC A



- SWAP

  SWAP A

  swaps the upper ~~upp~~ nibble and lower nibble.

iv) BITWISE MANIPULATION (Boolean Variable Mani.)

- CLR (Clear)

  CLR C

- SET B (set a particular bit)

  SETB P3.0 (RXD)
  SET B PSW.4 RS_1
  SET B PSW.3 RS_0

- CPL
- ANL
- ORL
- MOV
- JC (Jump on Carry)

- JC target address

  If there is carry it jump to target address

• JNC (Jump on No carry)

↳ If there is no carry, jump to target address

JNC target address.

• JB (Jump on bit)

Jump, if a bit is set

JB bit, target address.

↓ if it is set then jump to this.

JB P3.2, loop

• JBC ~~Jump~~

Jump if a bit is set & then clear it.

JBC P3.2, loop

• JNB

Jump if a bit is not set.

JNB P3.2, loop

v. Program Branching

• CJNE (Compare and jump if not equal)

CJNE destination, source, target

CJNE A, #99, BACK

↳ It jumps to BACK, if A & #99 is not same. When it is same then it will stop.

• **DJNZ** (Decrement & jump if not equal to zero)

• DJNZ byte target

If the byte is not zero, then decrement it by '1' and jump to target. when byte will become zero, it will stop.

Eg:

```
DJNZ R3, here
DJNZ R2, Back
```

• **NOP** (No operation)

- For making delay in program.
- Increase the delay.

• **ACALL** (Absolute call)

ACALL 11 bit address

- It can call upto 2K bytes memory address (PC)

• **LCALL** (Long call)

LCALL 16 bit address

- It can move further
- It can call upto 64K bytes of ROM.

• **RET** (Return)

- Return from call

• **RETI**

- Return from interrupt

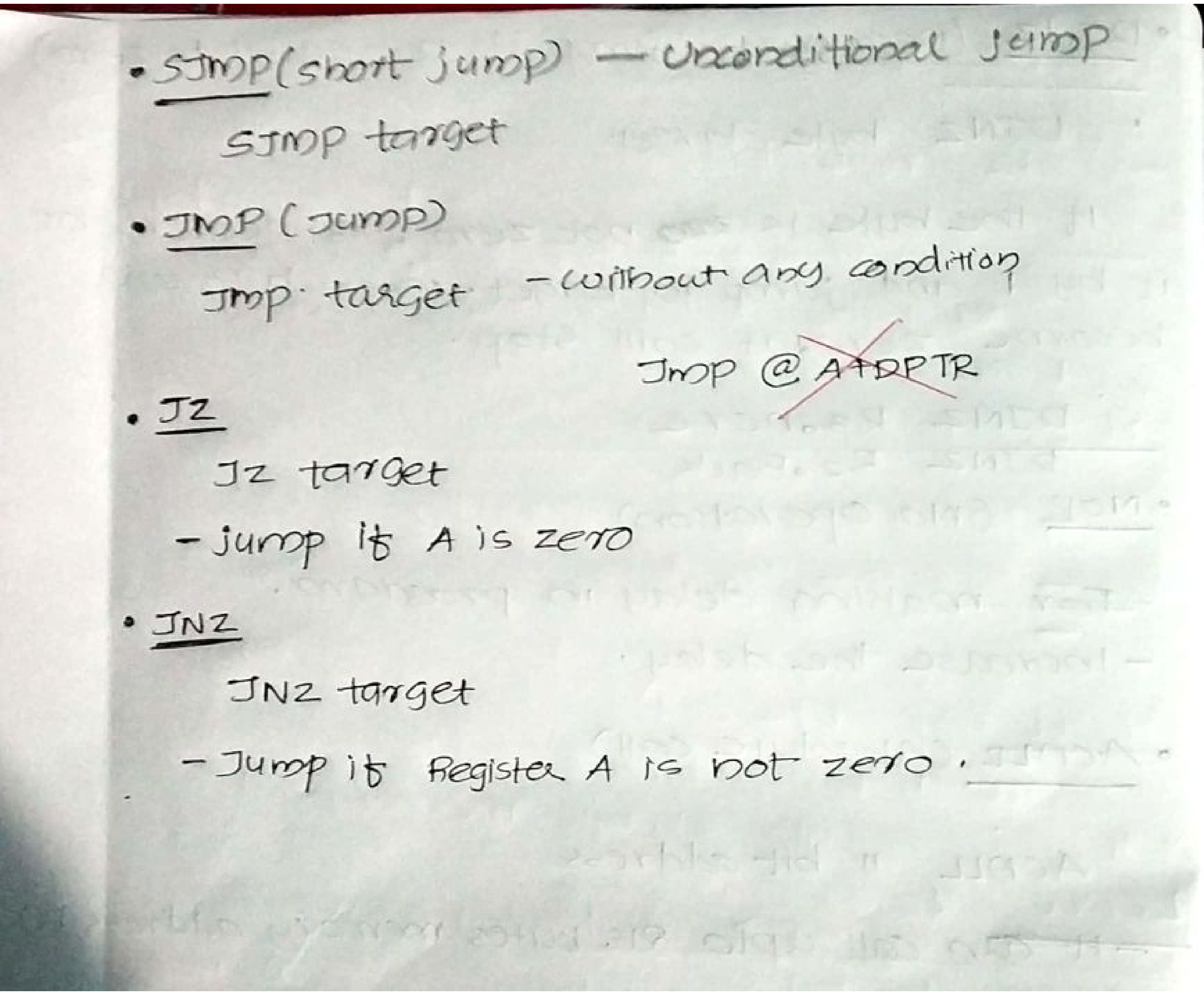
• **AJMP** (Absolute jump)

AJMP target

• **LJMP** (Long jump)

LJMP target

Unconditional jump



- **SJMP** (short jump) — Unconditional jump

  SJMP target

- **JMP** (jump)

  Jmp target - without any condition

  Jmp @ ~~A+DPTR~~

- **JZ**

  JZ target

  - jump if A is zero

- **JNZ**

  JNZ target

  - Jump if Register A is not zero.