Microprocessors & Microcontroller-1

Prof. Nitin Ahire
Xavier Institute Of Engineering,Mahim

Instruction Groups

- The 8051 has 255 instructions
  - Every 8-bit number from 00 to FF is used as opcode except for number A5.

- The instructions are grouped into 5 groups
  - Arithmetic
  - Logic
  - Data Transfer
  - Boolean
  - Branching

Arithmetic Instructions

- ADD
  - 8-bit addition between the accumulator (A) and a second operand.
    - The result is always in the accumulator.
    - The CY flag is set/reset appropriately.
- ADDC
  - 8-bit addition between the accumulator, a second operand and the previous value of the CY flag.
    - Useful for 16-bit addition in two steps.
    - The CY flag is set/reset appropriately.

Example – 16-bit Addition

Add 1E44H to 56CAH. Save the result in R1: R2 registers

```
        ORG 0000H
        CLR     C           ; Clear the CY flag
        MOV     A, #44H     ; The lower 8-bits of the 1st number
        ADD     A, #CAH     ; The lower 8-bits of the 2nd number
        MOV     R1, A       ; The result 0EH will be in R1. CY = 1.
        MOV     A, #1EH     ; The upper 8-bits of the 1st number
        ADDC    A, #56H     ; The upper 8-bits of the 2nd number
        MOV     R2, A       ; The result of the addition is 75H

END
```

The overall result: 750EH will be in R2:R1. CY = 0.

Arithmetic Instructions

- DA A
  - Decimal adjust the accumulator.
    - Format the accumulator into a proper 2 digit packed BCD number.
    - Operates only on the accumulator.
    - Works only after the ADD instruction.

- SUBB A, source
  - Subtract with Borrow. (In the 8051we have only SUBB)
    - Subtract an operand and the previous value of the borrow (carry) flag from the accumulator.
      - A ← A - <operand> - CY.
      - The result is always saved in the accumulator.
      - The CY flag is set/reset appropriately.

Example – BCD addition

Add 34 to 49 BCD

```
CLR     C           ; Clear the CY flag
MOV     A, #34H     ; Place 1st number in A
ADD     A, #49H     ; Add the 2nd number.
                    ; A = 7DH
DA      A           ; A = 83H
```

Arithmetic Instructions

- INC
  - Increment the operand by one.
    - The operand can be a register, a direct address, an indirect address, the data pointer.
- DEC
  - Decrement the operand by one.
    - The operand can be a register, a direct address, an indirect address.
- MUL AB / DIV AB
  - Multiply A by B and place result in A:B.
  - Divide A by B and place result in A:B.

Logical Operations

- ANL / ORL
  - Work on byte sized operands or the CY flag.
    - ANL A, Rn
    - ANL A, direct
    - ANL A, @Ri; i = 0 or 1 only
    - ANL A, #data
    - ANL direct, A; ANL 0F0,A
    - ANL direct, #data; ANL P1,#00001111b
    - ANL C, bit; ANL C,P2.2

Logical Operations

- XRL
  - Works on bytes only.

    XRL A,#09H

- CPL / CLR
  - Complement / Clear.

    CPL A

    CPL C
  - Work on the accumulator or a bit.
    - CLR P1.2
    - CLR P2.4
    - CLR ACC.7
    - CLR A

Logical Operations

- RL / RLC / RR / RRC A
  - Rotate the accumulator.
    - RL and RR without the carry
    - RLC and RRC rotate through the carry.

- SWAP A
  - Swap the upper and lower nibbles of the accumulator.

- Write a program to convert unpacked numbers to packed number
- Data – convert the number 09h and 02h

Solution

- Org 0000h

```
MOV A,#02H;      A ←02
MOV B,A          B ←02
MOV A,#09H       A← 09
SWAP A           A←  90
ADD A,B           A ←92
MOV R0,A        R0 ←92
END
```

Result R0= 92h

- Write a program to convert packed number to packed number
- Data – convert the number 92h

Solution

- ORG 0000H
- MOV A,#92H
- MOV B,A
- ANL A,#F0H
- SWAP A
- MOV R0,A
- MOV A,B
- ANL A,#0FH
- MOV R1,A
- END

Result: R0=09 and R1=02

Data Transfer Instructions

- MOV
  - 8-bit data transfer for internal RAM and the SFR.
    - MOV A, Rn — MOV A,R0
    - MOV A, direct ; — MOV A,90H
    - MOV A, @Ri ; — i = 0 or 1 only
    - MOV A, #data — MOV A,#34H
    - MOV Rn, A — MOV R0,A
    - MOV Rn, direct ; — MOV R0,0E0h
    - MOV Rn, #data — MOV R1,#35H

ACC	Accumulator	0E0H
B	B Register	0F0H
PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer (2 Bytes)	
DPL	Low Byte	82H
DPH	High Byte	83H
P0	Port 0	80H
P1	Port 1	90H
P2	Port 2	0A0H
P3	Port 3	0B0H
IP	Interrupt Priority Control	0B8H
IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
TCON	Timer/Counter Control	88H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH

Table 12 List of Registers

Data Transfer Instructions

- MOV direct, A
- MOV direct, Rn
- MOV direct, @Ri
- MOV direct, #data
- MOV @Ri, A
- MOV @Ri, direct
- MOV @Ri, #data

  MOV TMOD,#39H

Data Transfer Operations

- MOV (Single bit)
  - 1-bit data transfer involving the CY flag
    - MOV C, bit; MOV C, P0.0
    - MOV bit, C; MOV P0.0, C

- MOV
  - 16-bit data transfer involving the DPTR
    - MOV DPTR, #data

Data Transfer Instructions

- MOVC
  - Move Code Byte
    - Load the accumulator with a byte from program memory.
    - Must use indexed addressing

    - MOVC A, @A+DPTR
    - MOVC A, @A+PC

Example

- Example: look up table SQUR has the squares of values between 0 and 9 and register R3 has the values 0 to 9.

  write a program to fetch the squares from the look- up table

Solution

```
ORG 0000H
MOV DPTR,# SQUR ;       DPTR=0100h
MOV R3,#03h;            R3=03h
MOV A,R3;               A=03h
MOVC A,@A+DPTR;         A=09h

        ORG 0100H
SQUR: DB 00,01,04,09,16,25,36,49,64,81
 END
```

Example

- Assume that an ASC II character string is stored in on chip ROM starting address 200h. Write a program to bring each character into CPU and send it on P1.

solution

```
    ORG 0000h
    MOV DPTR,#200H          ;load DPTR
B1: CLR A                   ; A=0
    MOVC A,@A+DPTR          ; move data at A+ DPTR into A
    JZ EXIT                 ; exit if last ( null ) char
    MOV P1,A
    INC DPTR
    SJMP B1                 ;continue.
EXIT: END
```

```
    ORG 200h
    DB   “XAVIER INSTITUTE OF ENGG”,0
    END
```

On-chip and off-chip ROM

- EA^=VCC

meaning that upon reset 8051 executes the on chip program first, then when reaches the end of the on chip ROM it switches to external ROM.

on chip memory size 0000h to 0FFFh (4KB)

off chip memory size 1000h to FFFFh (60KB)

Total memory accessing capacity 64KB

Roll of RD^ and PSEN^

- For the ROM containing the program code, PSEN^ is used to fetch the code
- For ROM containing data, the RD^ signal is used to fetch the data
- To access the external data memory space we must use the instruction MOVX as described next.

Data Transfer Instructions

- MOVX
  - Data transfer between the accumulator and a byte from external data memory.
    - MOVX A, @Ri
    - MOVX A, @DPTR
    - MOVX @Ri, A
    - MOVX @DPTR, A

Data Transfer Instructions

- PUSH / POP
  - Push and Pop a data byte onto the stack.
  - The data byte is identified by a direct address from the internal RAM locations.

    - PUSH DPL
    - POP 40H
    - PUSH 7
    - POP 7

Data Transfer Instructions

- XCH
  - Exchange accumulator and a byte variable
    - XCH A, Rn
    - XCH A, direct
    - XCH A, @Ri

- XCHD
  - Exchange lower digit (nibble) of accumulator with the lower digit of the memory location specified.
    - XCHD A, @Ri

    - The lower 4-bits of the accumulator are exchanged with the lower 4-bits of the internal memory location identified indirectly by the index register.
    - The upper 4-bits of each are not modified.

Boolean Operations

- This group of instructions is associated with the single-bit operations of the 8051.
- This group allows manipulating the individual bits of bit addressable registers and memory locations as well as the CY flag.
  - The P, OV, and AC flags cannot be directly altered.
- This group includes:
  - Set, clear, and, or ,complement, move.
  - Conditional jumps.

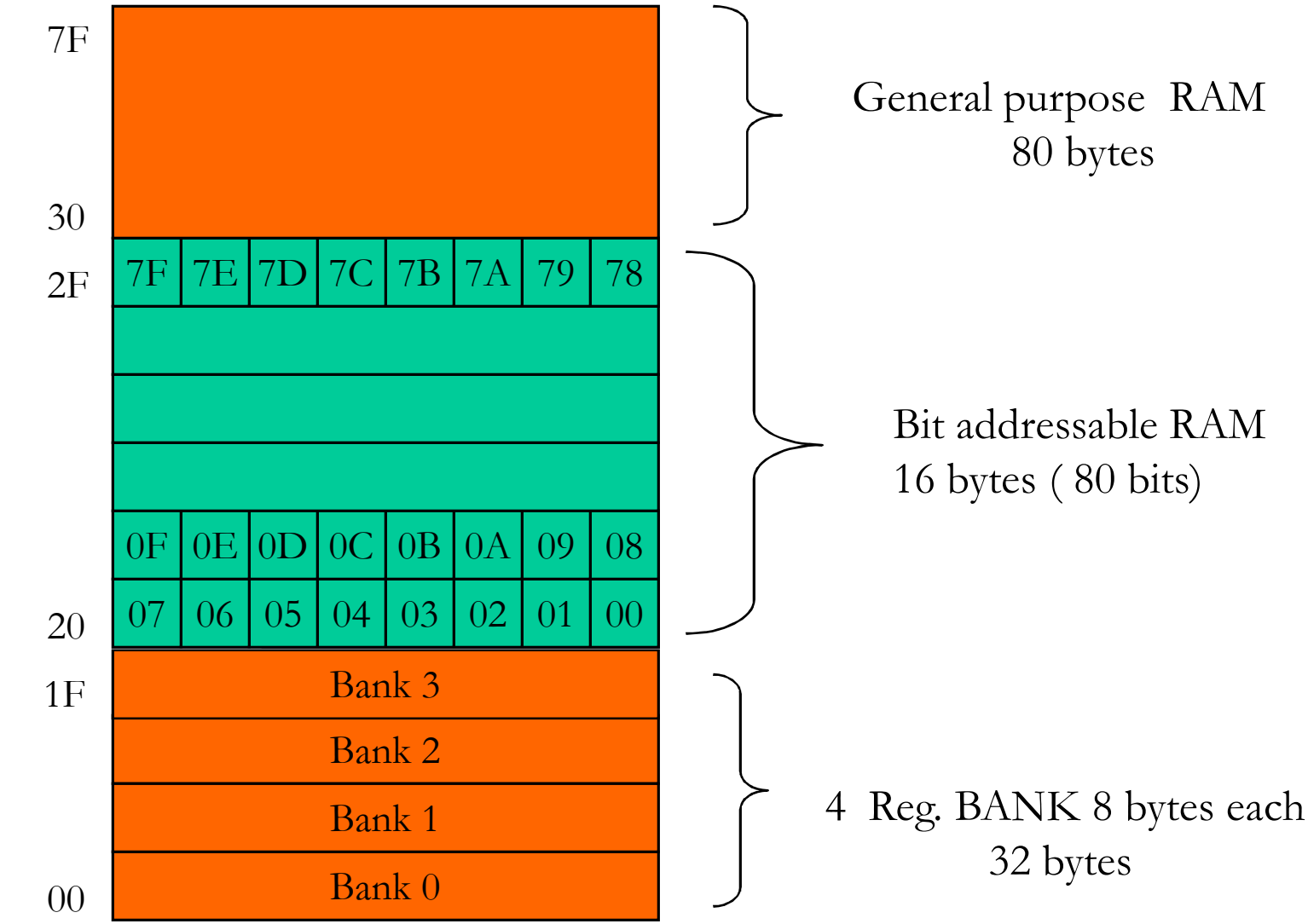
Boolean Operations

- CLR
  - Clear a bit or the CY flag.
    - **CLR P1.1**
    - **CLR C**
    - **CLR ACC.2**

- SETB
  - Set a bit or the CY flag.
    - **SETB psw.2**
    - **SETB C**

- CPL
  - Complement a bit.
    - **CPL 06H** ; Complement bit 06H of the bit addressable memory

Internal RAM memory organization



Boolean Operations

- ORL / ANL
  - OR / AND a bit with the CY flag.
    - ORL C, 20H ; OR bit 20 of bit addressable memory with the CY flag
    - ANL C, 34H ; AND complement of bit 34 of bit addressable memory with the CY flag.

- MOV
  - Data transfer between a bit and the CY flag.
    - MOV C, 3FH ; Copy the CY flag to bit 3F of the bit addressable memory.
    - MOV P1.2, C ; Copy the CY flag to bit 2 of P1.

Unconditional Jump

- LJMP 16 bit address
- SJMP 8 bit address
- AJMP (Absolute jump)

function: Transfer control unconditionally to a new address.

Unconditional Jump

- LJMP (long jump) :This is a 3 - byte instruction the first byte is the opcode and the next two bytes are the target object.
- LJMP is used to jump to any address location within 64KB code memory space of the 8051.

Unconditional Jump

- SJMP (Short jump) : This is a 2 –byte instruction the first byte is opcode and the second byte is a singed number displacement, which is added with the PC of the instruction following the SJMP to get the target address
- There fore in this jump the target address must be within -128 to +127 byte of the PC of the instruction after the SJMP. Generally it called relative Address.

Branching Instructions

- AJMP (Absolute jump)

  It transfer the program execution to the target address unconditionally

  the target address for this instruction must be with in 2K byte memory.

Branching Instructions

- Indirect Jump – JMP
  - JMP @A + DPTR

Branching Instructions

- The 8051 provides 2 forms for the CALL instruction:
  - Absolute Call – ACALL
    - Uses an 11-bit address similar to AJMP
    - The subroutine must be within the same 2K page.
  - Long Call – LCALL
    - Uses a 16-bit address similar to LJMP
    - The subroutine can be anywhere.
  - Both forms push the 16-bit address of the next instruction on the stack and update the stack pointer.

Branching Instructions

- The 8051 provides 2 forms for the return instruction:
  - Return from subroutine – RET
    - Pop the return address from the stack and continue execution there.
  - Return from ISR – RETI
    - Pop the return address from the stack.
    - Restore the interrupt logic to accept additional interrupts at the same priority level as the one just processed.
    - Continue execution at the address retrieved from the stack.
    - The PSW is not automatically restored.

Branching Instructions

- The 8051 supports 5 different conditional jump instructions.
  - ALL conditional jump instructions use an 8-bit signed offset.
  - Jump on Zero – JZ / JNZ
    - Jump if the A == 0 / A != 0
      - The check is done at the time of the instruction execution.
  - Jump on Carry – JC / JNC
    - Jump if the C flag is set / cleared.

Branching Instructions

- Jump on Bit – JB / JNB
  - Jump if the specified bit is set / cleared.
  - Any addressable bit can be specified.

- Jump if the Bit is set then Clear the bit – JBC
  - Jump if the specified bit is set.
  - Then clear the bit.

Branching Instructions

- Compare and Jump if Not Equal – CJNE
  - Compare the magnitude of the two operands and jump if they are not equal.
    - The values are considered to be unsigned.
    - The Carry flag is set / cleared appropriately.

    - CJNE A, direct, rel
    - CJNE A, #data, rel
    - CJNE Rn, #data, rel
    - CJNE @Ri, #data, rel

Example

- Keep monitoring P1 indefinitely for the value of 99h. Get out only when P1 has the value 99h

solution

```
        MOV P1,#0FFh
Back:   MOV A,P1
        CJNE A, #99, Back
```

Branching Instructions

- Decrement and Jump if Not Zero – DJNZ
  - Decrement the first operand by 1 and jump to the location identified by the second operand if the resulting value is not zero.

    - DJNZ Rn, rel
    - DJNZ direct, rel

- No Operation
  - NOP

  Do noting