8051 Microcontroller Logical Operations

Prepared by : A. B. Modi

Target Audience : 5th Semester Students

Learning Objectives

After learning this chapter, Students should be able to:

- ➢ Describe and Use byte-level AND, OR, XOR and NOT boolean instructions.
- ➢ Describe and Use bit-level AND, OR, XOR and Not boolean instructions.
- ➢ Describe and Use bit-level set, clear and data-moving instructions.
- ➢ Describe and Use Rotate instructions.
- ➢ Use the A register nibble-swapping instruction.

Introduction

- ➢ Data and SFRs may be manipulated using byte opcodes.
- ➢ Many of the SFRs, and a unique internal RAM area that is bit addressable, may be operated on the individual bit level.
- ➢ Bit operators are used when quick response is required as it makes the program compact.
- ➢ Boolean Opcodes : ANL, ORL, XRL and CPL
- ➢ Rotate opcodes : RL, RLC, RR, RRC, SWAP

Byte-Level Logical Operations

- ➢ Can be used with all types of addressing modes for the source of data bytes.
- ➢ Result can be stored on A or a direct address in internal RAM.
- ➢ These are called “byte-Level Logical operations” because the entire byte is affected.

ANL A, #n	*;AND each bit of A with the same bit of immediate number n, Result is placed in A.*
ANL A, add	*;AND each bit of A with the same bit of the direct RAM address, result is placed in A*
ANL A, Rr	*;AND each bit of A with the same bit of register Rr, result is placed In A.*
ANL A, @Rr	*;AND each bit of A with the same bit of the contents of RAM Address in Rp, result is placed in A.*
ANL add,A	*;AND each bit of A with the same bit of the direct RAM address, result is placed in direct RAM add.*
ANL add, #n	*;AND each bit of A with the same bit of the immediate number, result is placed in direct RAM add.*

ORL A, #n	*;OR each bit of A with the same bit of immediate number n, Result is placed in A.*
ORL A, add	*;OR each bit of A with the same bit of the direct RAM address, result is placed in A*
ORL A, Rr	*;OR each bit of A with the same bit of register Rr, result is placed in A.*
ORL A, @Rr	*;OR each bit of A with the same bit of the contents of RAM Address in Rp, result is placed in A.*
ORL add,A	*;OR each bit of A with the same bit of the direct RAM address, result is placed in direct RAM add.*
ORL add, #n	*;OR each bit of A with the same bit of the immediate number, result is placed in direct RAM add.*

XRL A, #n	*;XOR each bit of A with the same bit of immediate number n, Result is placed in A.*
XRL A, add	*;XOR each bit of A with the same bit of the direct RAM address, result is placed in A*
XRL A, Rr	*;XOR each bit of A with the same bit of register Rr, result is placed in A.*
XRL A, @Rr	*;XOR each bit of A with the same bit of the contents of RAM Address in Rp, result is placed in A.*
XRL add,A	*;XOR each bit of A with the same bit of the direct RAM address, result is placed in direct RAM add.*
XRL add, #n	*;XOR each bit of A with the same bit of the immediate number, result is placed in direct RAM add.*

CLR A ;clear each bit of A register to 0

CPL A ;complement each bit of register A

- ➢ ANL, ORL, XRL can use SFR port latches as destinations.
- ➢ If the direct address destination is one of the port SFRs, the data latched in the SFR, not the pin data, is used.
- ➢ No flags are affected unless the direct address is the PSW.
- ➢ Only internal RAM or SFRs may be logically manipulated.

- ➔ No flags are affected by the byte level logical operations unless the direct RAM address is the PSW.
- ➔ Many of these byte-level operations use a direct address, which can include the port SFR addresses, as a destination. The normal source of data from a port are the port pins; the normal destination for port data is the port latch.
- ➔ When the destination of a logical operation is the direct address of a port, the latch register, not the pins, is used both as source for the original data and then the destination for the altered byte of data.
- ➔ Any port operation that must first read the source data, logically operate on it, and then write it back to the source must use latch.
- ➔ Logical operations that use the port as a source, but not as a destination, use the pins of the oprt as the source of the data.

```
MOV A, #0FFh
MOV R0, #77h
ANL A,R0
MOV 15h, A
CPL A
ORL 15h, #88h
XRL A, 15h
XRL A, R0
ANL A, 15h
ORL A, R0
CLR A
XRL 15h, A
XRL A, R0
```

Bit - Level Logical Operations

- ➢ Certain internal RAM and SFRs can be addressed by their byte addresses or by the address of each bit within a byte.
- ➢ Bit addressing is very convenient when you wish to alter a single bit of a byte.
- ➢ Internal RAM byte addresses 20h to 2Fh are both byte and bit addressable.

Internal RAM Bit Addresses

Byte Address	Bit Addresses	Byte Address	Bit Addresses
20h	00h - 07h	28h	40h - 47h
21h	08h - 0Fh	29h	48h - 4Fh
22h	10h - 17h	2Ah	50h - 57h
23h	18h - 1Fh	2Bh	58h - 5Fh
24h	20h - 27h	2Ch	60h - 67h
25h	28h - 2Fh	2Dh	68h - 6Fh
26h	30h - 37h	2Eh	70h - 77h
27h	38h - 3Fh	2Fh	78h - 7Fh

Whole bytes do not have to be used up to store one or two bits of data.

SFR Bit Addresses

- All SFRs may be addressed at the byte level by using the direct address assigned to it, but not all of the SFRs are addressable at the bit level.

SFR	Direct Address	Bit addresses (D0 - D7)
A	0E0h	0E0h - 0E7h
B	0F0h	0F0h - 0F7h
IE	0A8h	0A8h - 0AFh
IP	0B8h	0B8h - 0BFh
P0	80h	80h - 87h
P1	90h	90h - 97h
P2	0A0h	0A0h - 0A7h
P3	0B0h	0B0h - 0B7h
PSW	0D0h	0D0h - 0D7h
TCON	88h	88h - 8Fh
SCON	98h	98h - 9Fh

Bit Level Boolean Operations

- ➢ The bit level boolean logical opcodes operate on any addressable RAM or SFR bit.
- ➢ Only the SFRs that have been identified as bit addressable may be used in bit operations.
- ➢ If the destination bit is a port bit, the SFR latch bit is affected, not the pin.
- ➢ ANL C,/b and ORLc,/b do not alter the addressed bit b.

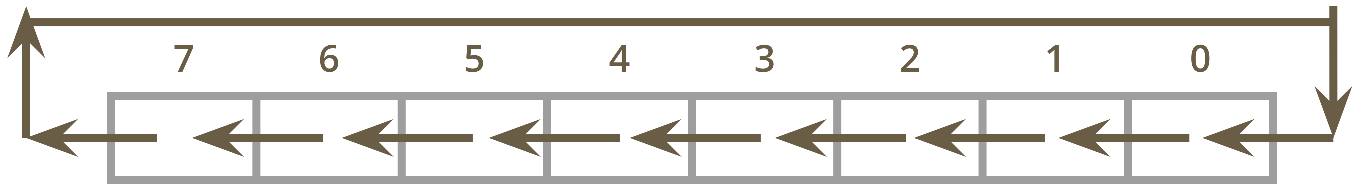
```
ANL C, b ;AND C and the addressed bit; put the result in Carry Flag
ANL C,/b;AND C and the complement of the addressed bit; put the result in Carry
         Flag;addressed bit is not altered
ORL C, b ; OR C and the addressed bit; put the result in Carry Flag
ORL C,/b ;OR C and the complement of the addressed bit; put the result in Carry
         Flag;addressed bit is not altered
CPL C    ;Complement the Carry Flag
CPL b    ;Complement the addressed bit
CLR C    ;Clear the Carry Flag to 0
CLR b    ;Clear the addressed bit to 0
MOV C,b  ;Copy the addressed bit to the Carry Flag
MOV b,C  ;Copy the Carry Flag to the addressed bit
SETB C   ;Set the Flag to 1
SETB b   ;Set the addressed bit to 1
```

```
STB 00h
MOV C, 00h
MOV 7Fh, C
ANL C, /00h
ORL C, 00h
CPL 7Fh
CLR C
ORL C, /7Fh
```

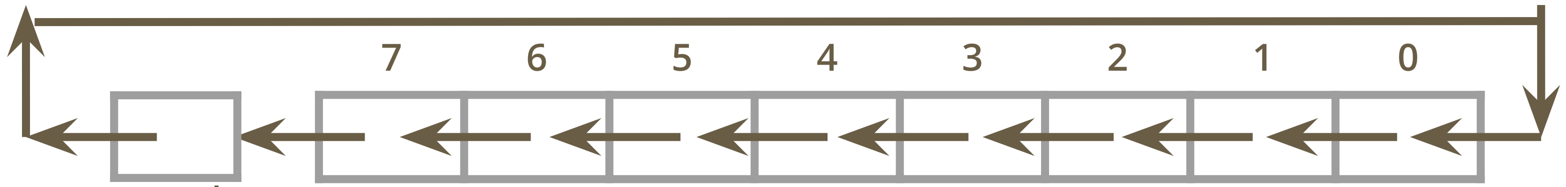
Rotate and Swap Operations

- ➢ The ability to rotate data is useful for inspecting bits of a byte without using individual bit opcodes.
- ➢ The A register can be rotated one bit position to the left or right without Carry Flag, then the rotation involves the 8 bits of the A register.
- ➢ The A register can be rotated one bit position to the left or right with Carry Flag, then the rotation involves the 9 bits of the A register.
- ➢ The SWAP instruction can be thought of as a rotation of nibbles in A register.

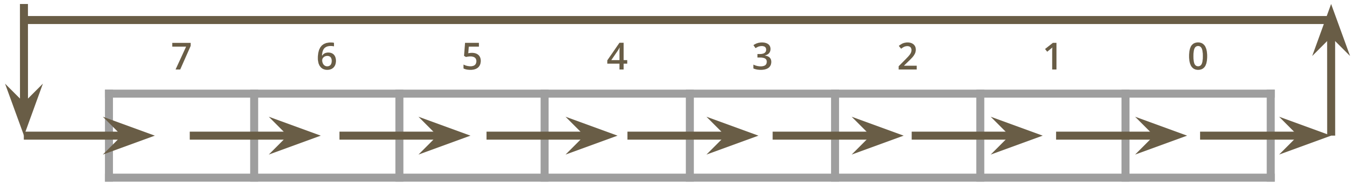
```
RL A     ;Rotate the A register one bit position to the left

RLC A    ;Rotate the A register and the carry flag one bit position to the left

RR A     ;Rotate the A register one bit position to the right

RRC A    ;Rotate the A register and the carry flag one bit position to the right

SWAP A   ;interchange the nibbles of register A
```



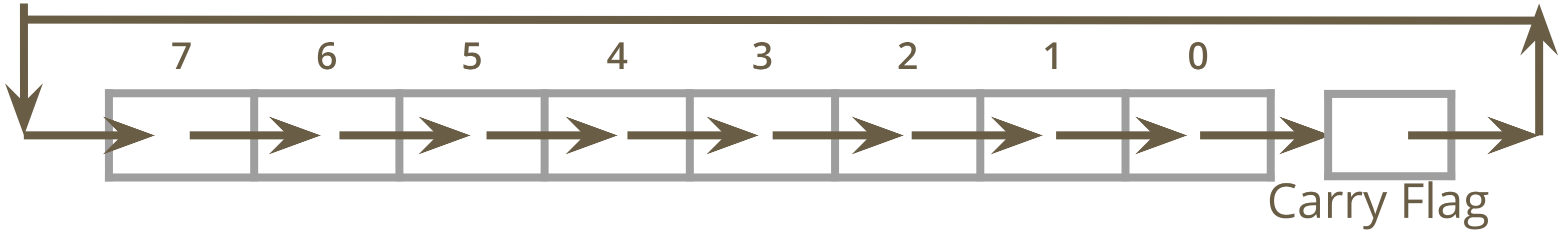
Instruction : RL A



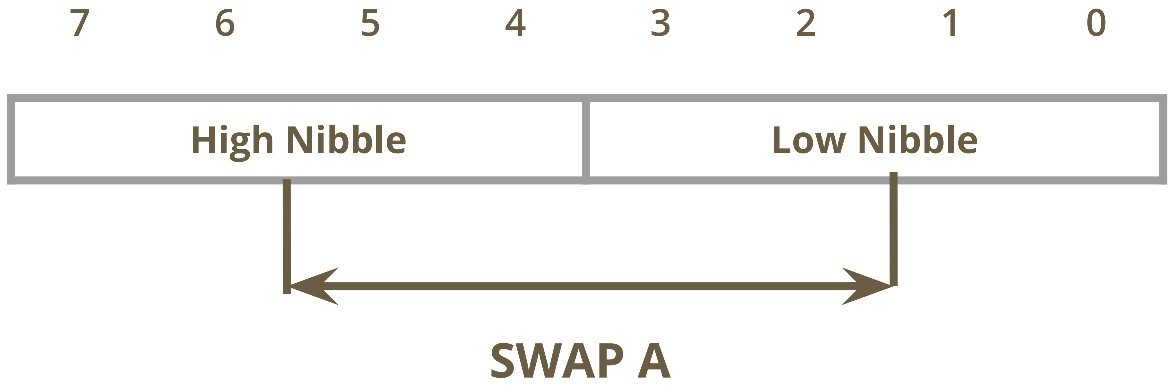
Instruction : RLC A



Instruction : RR A



Instruction : RRC A


7
6
5
4
3
2
1
0
High Nibble
Low Nibble
SWAP A

Now try these examples

1. OR the contents of ports 1 & 2; put the result in external RAM location 0100h.
2. Set port 0, bits 1,3,5,7 to 1; and bits 0,2,4,6 to 0.
3. Invert the data on the port 0 pins and write the data to port 1.
4. Swap the nibbles of R0 and R1 so that the low nibble of R0 swaps with the high nibble of R1 and the high nibble of R0 swaps with the low nibbles of R1.
5. Make the low nibble of R5 the complement of the high nibble of R6.
6. XOR a number with whatever is in A so that the result is FFh.
7. Store the most significant nibble of A in both the nibbles of register R5.
8. Treat registers R0 and R1 as 16 bit registers, and rotate them one place to the left; bit 7 of R0 becomes bit 0 of R1, bit 7 of R1 becomes bit 0 of R0, and so on.