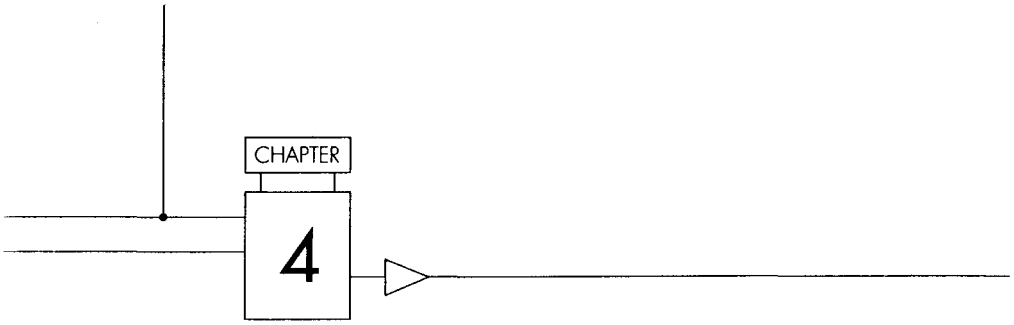


Logical Operations

Chapter Outline

Introduction

One application area the 8051 is designed to fill is that of machine control. A large part of machine control concerns sensing the on–off states of external switches, making decisions based on the switch states, and then turning external circuits on or off.

Single point sensing and control implies a need for *byte* and *bit* opcodes that operate on data using Boolean operators. All 8051 RAM areas, both 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 upon at the *individual* bit level. Bit operators are notably efficient when speed of response is needed. Bit operators yield compact program code that enhances program execution speed.

The two data levels, byte or bit, at which the Boolean instructions operate are shown in the following table:

BOOLEAN OPERATOR	8051 MNEMONIC
AND	ANL (AND logical)
OR	ORL (OR logical)
XOR	XRL (exclusive OR logical)
NOT	CPL (complement)

There are also rotate opcodes that operate only on a byte, or a byte and the carry flag, to permit limited 8- and 9-bit shift-register operations. The following table shows the rotate opcodes:

Mnemonic	Operation
RL	Rotate a byte to the left; the Most Significant Bit (MSB) becomes the Least Significant Bit (LSB)
RLC	Rotate a byte and the carry bit left; the carry becomes the LSB, the MSB becomes the carry
RR	Rotate a byte to the right; the LSB becomes the MSB
RRC	Rotate a byte and the carry to the right; the LSB becomes the carry, and the carry the MSB
SWAP	Exchange the low and high nibbles in a byte

Byte-Level Logical Operations

The byte-level logical operations use all four addressing modes for the source of a data byte. The A register or a direct address in internal RAM is the destination of the logical operation result.

Keep in mind that all such operations are done using each individual bit of the destination and source bytes. These operations, called *byte-level Boolean operations* because the entire byte is affected, are listed in the following table:

Mnemonic	Operation
ANL A,#n	AND each bit of A with the same bit of immediate number n; put the results in A
ANL A,add	AND each bit of A with the same bit of the direct RAM address; put the results in A
ANL A,Rr	AND each bit of A with the same bit of register Rr; put the results in A
ANL A,@Rp	AND each bit of A with the same bit of the contents of the RAM address contained in Rp; put the results in A
ANL add,A	AND each bit of A with the direct RAM address; put the results in the direct RAM address
ANL add,#n	AND each bit of the RAM address with the same bit in the number n; put the result in the RAM address
ORL A,#n	OR each bit of A with the same bit of n; put the results in A
ORL A,add	OR each bit of A with the same bit of the direct RAM address; put the results in A
ORL A,Rr	OR each bit of A with the same bit of register Rr; put the results in A
ORL A,@Rp	OR each bit of A with the same bit of the contents of the RAM address contained in Rp; put the results in A
ORL add,A	OR each bit of A with the direct RAM address; put the results in the direct RAM address
ORL add,#n	OR each bit of the RAM address with the same bit in the number n; put the result in the RAM address
XRL A,#n	XOR each bit of A with the same bit of n; put the results in A
XRL A,add	XOR each bit of A with the same bit of the direct RAM address; put the results in A
XRL A,Rr	XOR each bit of A with the same bit of register Rr; put the results in A
XRL A,@Rp	XOR each bit of A with the same bit of the contents of the RAM address contained in Rp; put the results in A
XRL add,A	XOR each bit of A with the direct RAM address; put the results in the direct RAM address

XRL add,#n	XOR each bit of the RAM address with the same bit in the number n; put the result in the RAM address
CLR A	Clear each bit of the A register to zero
CPL A	Complement each bit of A; every 1 becomes a 0, and each 0 becomes a 1

Note that no flags are affected 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 is 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 the *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 (now the destination) must use the *latch*. Logical operations that use the port as a source, but *not* as a destination, use the *pins* of the port as the source of the data.

For example, the port 0 latch contains FFh, but the pins are all driving transistor bases and are close to ground level. The logical operation

ANL P0,#0Fh

which is designed to turn the upper nibble transistors off, reads FFh from the latch, ANDs it with 0Fh to produce 0Fh as a result, and then writes it back to the latch to turn these transistors off. Reading the pins produces the result 00h, turning all transistors off, in error. But, the operation

ANL A,P0

produces A = 00h by using the port 0 pin data, which is 00h.

The following table shows byte-level logical operation examples:

Mnemonic	Operation
MOV A,#0FFh	A = FFh
MOV R0,#77h	R0 = 77h
ANL A,R0	A = 77h
MOV 15h,A	15h = 77h
CPL A	A = 88h
ORL 15h,#88h	15h = FFh
XRL A,15h	A = 77h
XRL A,R0	A = 00h
ANL A,15h	A = 00h
ORL A,R0	A = 77h
CLR A	A = 00h
XRL 15h,A	15h = FFh
XRL A,R0	A = 77h

Note that instructions that can use the SFR port latches as destinations are ANL, ORL, and XRL.

CAUTION

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.

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, in a control register for instance, without having to wonder what you need to do to avoid altering some other crucial bit of the same byte. The assembler can also equate bit addresses to labels that make the program more readable. For example, bit 4 of TCON can become TR0, a label for the timer 0 run bit.

The ability to operate on individual bits creates the need for an area of RAM that contains data addresses that hold a single bit. Internal RAM byte addresses 20h to 2Fh serve this need and are both byte and bit addressable. The bit addresses are numbered from 00h to 7Fh to represent the 128d bit addresses (16d bytes × 8 bits) that exist from byte addresses 20h to 2Fh. Bit 0 of *byte* address 20h is *bit* address 00h, and bit 7 of *byte* address 2Fh is *bit* address 7Fh. *You must know your bits from your bytes to take advantage of this RAM area.*

Internal RAM Bit Addresses

The availability of individual bit addresses in internal RAM makes the use of the RAM very efficient when storing bit information. Whole bytes do not have to be used up to store one or two bits of data.

The correspondence between byte and bit addresses are shown in the following table:

BYTE ADDRESS (HEX)	BIT ADDRESSES (HEX)
20	00–07
21	08–0F
22	10–17
23	18–1F
24	20–27
25	28–2F
26	30–37
27	38–3F
28	40–47
29	48–4F
2A	50–57
2B	58–5F
2C	60–67
2D	68–6F
2E	70–77
2F	78–7F

Interpolation of this table shows, for example, the address of bit 3 of internal RAM byte address 2Ch is 63h, the bit address of bit 5 of RAM address 21h is 0Dh, and bit address 47h is bit 7 of RAM byte address 28h.

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. The SFRs that are also bit addressable form the bit address by using the five most significant bits of the direct address for that SFR, together with the three least significant bits that identify the bit position from position 0 (LSB) to 7 (MSB).

The bit-addressable SFR and the corresponding bit addresses are as follows:

SFR	DIRECT ADDRESS (HEX)	BIT ADDRESSES (HEX)
A	0E0	0E0–0E7
B	0F0	0F0–0F7
IE	0A8	0A8–0AF
IP	0B8	0B8–0BF
P0	80	80–87
P1	90	90–97
P2	0A0	0A0–0A7
P3	0B0	0B0–0B7
PSW	0D0	0D0–0D7
TCON	88	88–8F
SCON	98	98–9F

The patterns in this table show the direct addresses assigned to the SFR bytes all have bits 0–3 equal to zero so that the address of the byte is also the address of the LSB. For example, bit 0E3h is bit 3 of the A register. The carry flag, which is bit 7 of the PSW, is bit addressable as 0D7h. The assembler can also "understand" more descriptive mnemonics, such as P0.5 for bit 5 of port 0, which is more formally addressed as 85h.

Figure 4.1 shows all the bit-addressable SFRs and the function of each addressable bit. (Refer to Chapter 2 for more detailed descriptions of the SFR bit functions.)

Bit-Level Boolean Operations

The bit-level Boolean logical opcodes operate on any addressable RAM or SFR bit. The carry flag (C) in the PSW special-function register is the destination for most of the opcodes because the flag can be tested and the program flow changed using instructions covered in Chapter 6.

The following table lists the Boolean bit-level operations.

Mnemonic	Operation
ANL C,b	AND C and the addressed bit; put the result in C
ANL C,/b	AND C and the complement of the addressed bit; put the result in C; the addressed bit is not altered
ORL C,b	OR C and the addressed bit; put the result in C
ORL C,/b	OR C and the complement of the addressed bit; put the result in C; the addressed bit is not altered
CPL C	Complement the C flag
CPL b	Complement the addressed bit
CLR C	Clear the C flag to zero
CLR b	Clear the addressed bit to zero
MOV C,b	Copy the addressed bit to the C flag
MOV b,C	Copy the C flag to the addressed bit
SETB C	Set the flag to one
SETB b	Set the addressed bit to one

Note that no flags, other than the C flag, are affected, unless the flag is an addressed bit.

As is the case for byte-logical operations when addressing ports as destinations, a port bit used as a destination for a logical operation is part of the SFR latch, not the pin. A port bit used as a source *only* is a pin, not the latch. The bit instructions that can use a SFR latch bit are: CLR, CPL, MOV, and SETB.

FIGURES 4.1 Bit-Addressable Control Registers

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	Reserved	P

PROGRAM STATUS WORD (PSW) SPECIAL FUNCTION REGISTER. BIT ADDRESSES D0h to D7h.

Bit	Function
7	Carry flag
6	Auxiliary carry flag
5	User flag 0
4	Register bank select bit 1
3	Register bank select bit 0
2	Overflow flag
1	Not used (reserved for future)
0	Parity flag

7	6	5	4	3	2	1	0
EA	Reserved	Reserved	ES	ET1	EX1	ET0	EX0

INTERRUPT ENABLE (IE) SPECIAL FUNCTION REGISTER. BIT ADDRESSES A8h TO AFh.

Bit	Function
7	Disables all interrupts
6	Not used (reserved for future)
5	Not used (reserved for future)
4	Serial port interrupt enable
3	Timer 1 overflow interrupt enable
2	External interrupt 1 enable
1	Timer 0 interrupt enable
0	External interrupt 0 enable

EA disables all interrupts when cleared to 0; if EA = 1 then each individual interrupt will be enabled if 1, and disabled if 0.

7	6	5	4	3	2	1	0
*	*	Reserved	PS	PT1	PX1	PT0	PX0

INTERRUPT PRIORITY (IP) SPECIAL FUNCTION REGISTER. BIT ADDRESSES B8h to BFh.

Bit	Function
7	Not implemented
6	Not implemented

Continued

Bit	Function
5	Not used (reserved for future)
4	Serial port interrupt priority
3	Timer 1 interrupt priority
2	External interrupt 1 priority
1	Timer 0 interrupt priority
0	External interrupt 0 priority

The priority bit may be set to 1 (highest) or 0 (lowest).

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TIMER/COUNTER CONTROL (TCON) SPECIAL FUNCTION REGISTER. BIT ADDRESSES 88h to 8Fh.

Bit	Function
7	Timer 1 overflow flag
6	Timer run control
5	Timer 0 overflow flag
4	Timer 0 run control
3	External interrupt 1 edge flag
2	External interrupt 1 mode control
1	External interrupt 0 edge flag
0	External interrupt 0 mode control

All flags can be set by the indicated hardware action; the flags are cleared when interrupt is serviced by the processor.

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SERIAL PORT CONTROL (SCON) SPECIAL FUNCTION REGISTER. BIT ADDRESSES 98h to 9Fh.

Bit	Function
7	Serial port mode bit 0
6	Serial port mode bit 1
5	*Multiprocessor communications enable*
4	Receive enable
3	Transmitted bit in modes 2 and 3
2	Received bit in modes 2 and 3
1	Transmit interrupt flag
0	Receive interrupt flag

Bit-level logical operation examples are shown in the following table:

Mnemonic	Operation
SETB 00h	Bit 0 of RAM byte 20h = 1
MOV C,00h	C = 1
MOV 7Fh,C	Bit 7 of RAM byte 2Fh = 1
ANL C,/00h	C = 0; bit 0 of RAM byte 20h = 1
ORL C,00h	C = 1
CPL 7Fh	Bit 7 of RAM byte 2Fh = 0
CLR C	C = 0
ORL C,/7Fh	C = 1; bit 7 of RAM byte 2Fh = 0

CAUTION

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 ORL C,/b do not alter the addressed bit b.

Rotate and Swap Operations

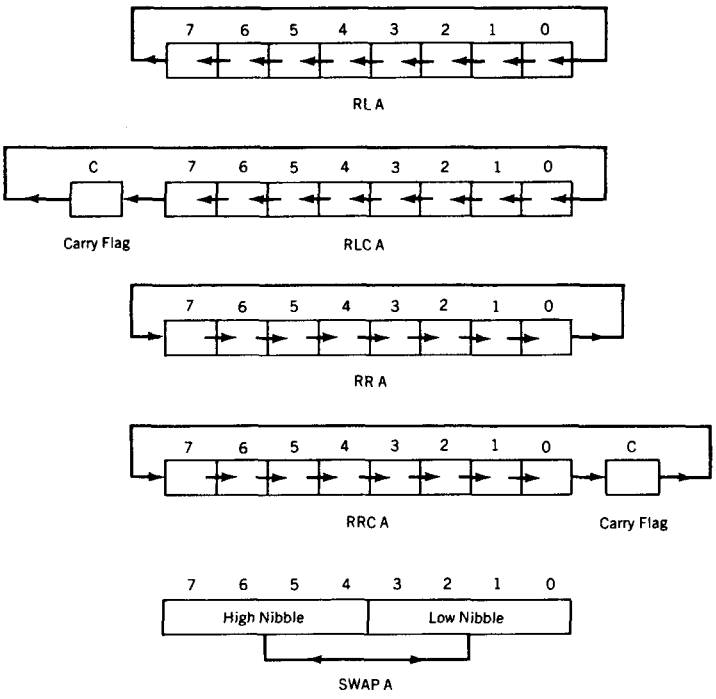
The ability to rotate data is useful for inspecting bits of a byte without using individual bit opcodes. The A register can be rotated one bit position to the left or right with or without including the C flag in the rotation. If the C flag is not included, then the rotation involves the eight bits of the A register. If the C flag is included, then nine bits are involved in the rotation. Including the C flag enables the programmer to construct rotate operations involving any number of bytes.

The SWAP instruction can be thought of as a rotation of nibbles in the A register. Figure 4.2 diagrams the rotate and swap operations, which are given in the following table:

Mnemonic	Operation
RL A	Rotate the A register one bit position to the left; bit A0 to bit A1, A1 to A2, A2 to A3, A3 to A4, A4 to A5, A5 to A6, A6 to A7, and A7 to A0
RLC A	Rotate the A register and the carry flag, as a ninth bit, one bit position to the left; bit A0 to bit A1, A1 to A2, A2 to A3, A3 to A4, A4 to A5, A5 to A6, A6 to A7, A7 to the carry flag, and the carry flag to A0
RR A	Rotate the A register one bit position to the right; bit A0 to bit A7, A6 to A5, A5 to A4, A4 to A3, A3 to A2, A2 to A1, and A1 to A0
RRC A	Rotate the A register and the carry flag, as a ninth bit, one bit position to the right; bit A0 to the carry flag, carry flag to A7, A7 to A6, A6 to A5, A5 to A4, A4 to A3, A3 to A2, A2 to A1, and A1 to A0
SWAP A	Interchange the nibbles of register A; put the high nibble in the low nibble position and the low nibble in the high nibble position

Note that no flags, other than the carry flag in RRC and RLC, are affected. If the carry is used as part of a rotate instruction, the state of the carry flag should be known before the rotate is done.

FIGURE 4.2 *Register A Rotate Operations*



The following table shows examples of rotate and swap operations:

Mnemonic	**Operation**
MOV A,#0A5h	A = 10100101b = A5h
RR A	A = 11010010b = D2h
RR A	A = 01101001b = 69h
RR A	A = 10110100b = B4h
RR A	A = 01011010b = 5Ah
SWAP A	A = 10100101b = A5h
CLR C	C = 0; A = 10100101b = A5h
RRC A	C = 1; A = 01010010b = 52h
RRC A	C = 0; A = 10101001b = A9h
RL A	A = 01010011b = 53h
RL A	A = 10100110b = A6h

```
SWAP A      C = 0; A = 01101010b = 6Ah
RLC A       C = 0; A = 11010100b = D4h
RLC A       C = 1; A = 10101000b = A8h
SWAP A      C = 1; A = 10001010b = 8Ah
```

CAUTION

Know the state of the carry flag when using RRC or RRL.

Rotation and swap operations are limited to the A register.

Example Programs

The programs in this section are written using only opcodes covered to this point in the text. The challenge is to minimize the number of lines of code.

EXAMPLE PROBLEM 4.1

Double the number in register R2, and put the result in registers R3 (high byte) and R4 (low byte).

▪ **Thoughts on the Problem** The largest number in R2 is FFh; the largest result is 1FEh. There are at least three ways to solve this problem: Use the MUL instruction (multiply, covered in Chapter 5), add R2 to itself, or shift R2 left one time. The solution that shifts R2 left is as follows:

Mnemonic	Operation
MOV R3,#00h	Clear R3 to receive high byte
CLR C	Clear the carry to receive high bit of 2 × R2
MOV A,R2	Get R2 to A
RLC A	Rotate left, which doubles the number in A
MOV R4,A	Put low byte of result in R4
CLR A	Clear A to receive carry
RLC A	The carry bit is now bit 0 of A
MOV R3,A	Transfer any carry bit to R3

COMMENT

Note how the carry flag has to be cleared to a known state before being used in a rotate operation.

EXAMPLE PROBLEM 4.2

OR the contents of ports 1 and 2; put the result in external RAM location 0100h.

▪ ***Thoughts on the Problem*** *The ports should be input ports for this problem to make any* physical sense; otherwise, we would not know whether to use the pin data or the port SFR latch data.

The solution is as follows:

Mnemonic	*Operation*
MOV A,90h	Copy the pin data from port 1 to A
ORL A,0A0h	OR the contents of A with port 2; results in A
MOV DPTR,#0100h	Set the DPTR to point to external RAM address
MOVX @DPTR,A	Store the result

COMMENT

Any time the port is the source of data, the pin levels are read; when the port is the destination, the latch is written. If the port is both source and destination (read–modify–write instructions), then the latch is used.

EXAMPLE PROBLEM 4.3

Find a number that, when XORed to the A register, results in the number 3Fh in A.

- **Thoughts on the Problem** Any number can be in A, so we will work backwards:

3Fh = A XOR N A XOR 3Fh = A XOR A XOR N = N

The solution is as follows:

Mnemonic	Operation
MOV R0,A	Save A in R0
XOR A,#3Fh	XOR A and 3Fh; forming N
XOR A,R0	XOR A and N yielding 3Fh

COMMENT

Does this program work? Let's try several A's and see.

A = FFh	A XOR 3Fh = C0h	C0h XOR FFh = 3Fh
A = 00h	A XOR 3Fh = 3Fh	3Fh XOR 00h = 3Fh
A = 5Ah	A XOR 3Fh = 65h	65h XOR 5Ah = 3Fh

Summary

Boolean logic, rotate, and swap instructions are covered in this chapter. Byte-level operations involve each individual bit of a source byte operating on the same bit position in the destination byte; the results are put in the destination, while the source is not changed:

ANL destination,source
ORL destination,source
XRL destination,source
CLR A
CPL A
RR A
RL A
RRC A
RLC A
SWAP A

Bit-level operations involve individual bits found in one area of internal RAM and certain SFRs that may be addressed both by the assigned direct-byte address and eight individual bit addresses. The following Boolean logical operations may be done on each of these addressable bits:

ANL bit
ORL bit

CLR bit

CPL bit

SETB bit

MOV destination bit, source bit

Problems

Write programs that perform the tasks listed using only opcodes that have been discussed in this and previous chapters. Write comments for each line of code and try to use as few lines as possible.

1. Set Port 0, bits 1,3,5, and 7, to one; set the rest to zero.
2. Clear bit 3 of RAM location 22h without affecting any other bit.
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 nibble of R1.
5. Complement the lower nibble of RAM location 2Ah.
6. Make the low nibble of R5 the complement of the high nibble of R6.
7. Make the high nibble of R5 the complement of the low nibble of R6.
8. Move bit 6 of R0 to bit 3 of port 3.
9. Move bit 4 of RAM location 30h to bit 2 of A.
10. XOR a number with whatever is in A so that the result is FFh.
11. Store the most significant nibble of A in both nibbles of register R5; for example, if A = B6h, then R5 = BBh.
12. Store the least significant nibble of A in both nibbles of RAM address 3Ch; for example, *if* A = 36h, *then* 3Ch = 66h.
13. Set the carry flag to one if the number in A is even; set the carry flag to zero if the number in A is odd.
14. 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.
15. Repeat Problem 14 but rotate the registers one place to the right.
16. Rotate the DPTR one place to the left; bit 15 becomes bit 0.
17. Repeat problem 16 but rotate the DPTR one place to the right.
18. Shift register B one place to the left; bit 0 becomes a zero, bit 6 becomes bit 7, and so on. Bit 7 is lost.