6

Bit-Processing Instructions

Objectives

- Identify the bit-addressable address spaces in the 8051
- Discuss the advantages of bit addressability
- Illustrate the bit operations with I/O ports
- Appreciate the use of carry flag in bit processing instructions
- Develop the programs to illustrate the use of bit processing instructions

Key Terms

• Boolean Processor	• Bit-addressable RAM	• Complement
• Bit Addressability	• Carry Flag: Boolean Accumulator	• Bit/Byte Address
• Bit Addressable SFR	• Set/Clear	• Conditional Jumps

In an 8-bit microcontroller, the minimum size of data that can be accessed or processed at a time is one byte. But, in many real-world machine control applications, we may need to manipulate (read/write/modify) only one bit at a time. For example, sensing the state (ON/OFF) of switch, make decision based upon state of the switch, and turn ON or turn OFF the external device. Such operations need to manipulate only required bits of a byte. If we perform these operations by accessing whole bytes, we should manipulate only required bits of byte without disturbing other bits, which demands additional efforts and care from the programmer.

6.1 BIT ADDRESSABILITY

The 8051 simplifies above-mentioned problem by providing unique and powerful feature of single bit addressability and single-bit operations. It contains a complete Boolean (single bit) processor and its instruction set is optimized for the single-bit operations. It supports SET, CLEAR, COMPLEMENT, AND and OR single bit operations. This feature makes the 8051 one of the most obvious choice for industrial machine-control applications.

Bit-processing operations provide the following advantages.

1. Faster execution of a program
2. Less memory required by a program
3. Program listing becomes simple and more readable

Example 6.2 will demonstrate the above advantages of using bit-processing instructions.

THINK BOX 6.1

What wrong may happen if we use following instruction to clear bit 7 of PSW?
MOV PSW, #00H or ANL PSW, #00H?
Other bits (bit 0 to 6) are unnecessarily cleared (disturbed). This may change selected register bank and affect the status of other flags which may result into logical error in a program

6.2 BIT-ADDRESSABLE MEMORIES

The 8051 has bit-addressable memory locations in internal RAM and special function registers. The internal RAM contains 128 addressable bits and majority of SFRs are bit-addressable including all I/O port pins.

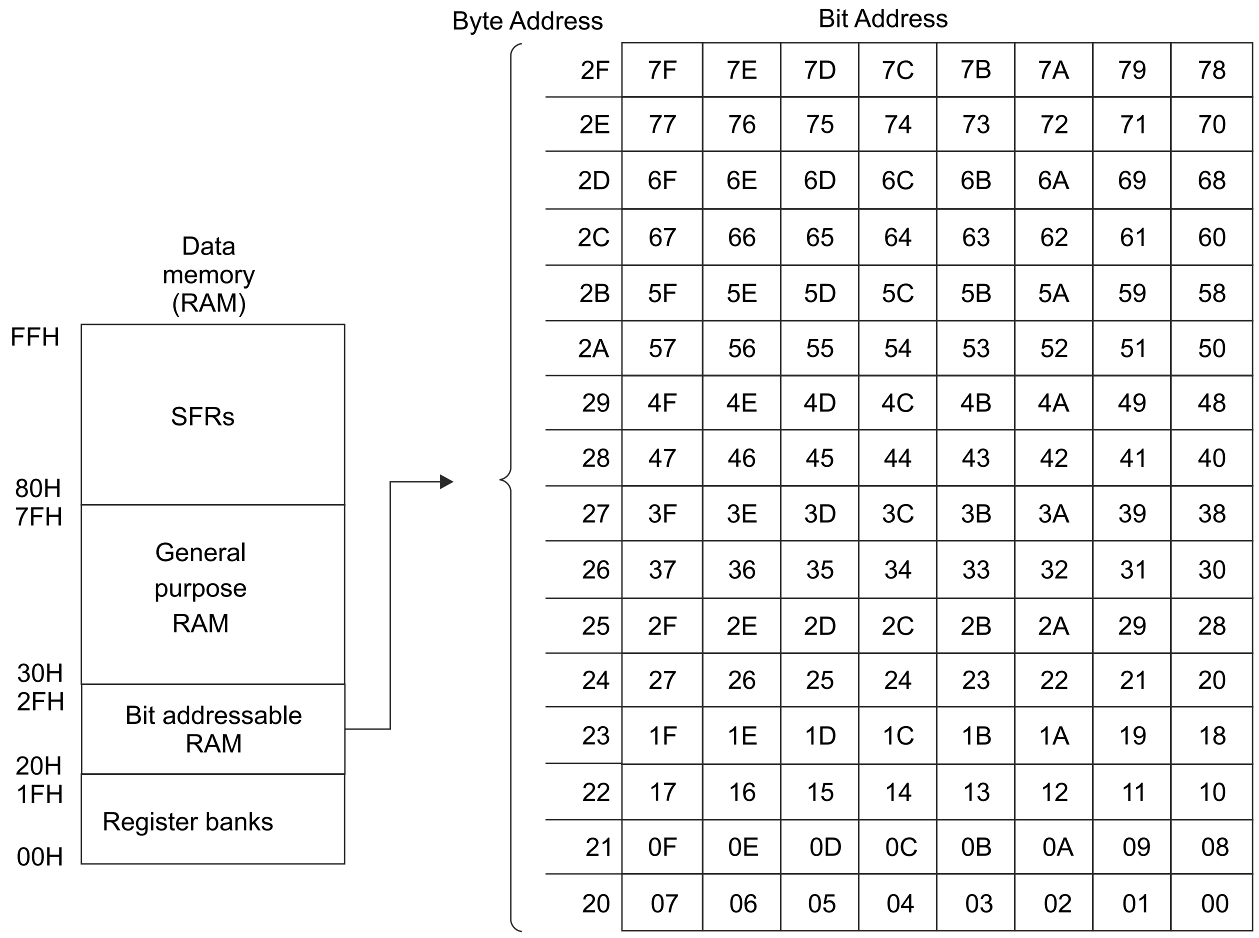
6.2.1 Bit-Addressable Internal RAM

The 8051 has a bit-addressable area of 16 bytes from byte addresses 20H to 2FH in internal RAM, forming a total of 128 (16x8) addressable bits. Addressable bits are assigned bit addresses from 00H to 7FH. The bit-addressable area with their individual bit addresses is shown in Figure 6.1.

As shown in Figure 6.1, internal RAM locations 20H to 2FH are both bit as well as byte addressable. Byte address 20H is assigned bit addresses 00H to 07H (00H address being assigned to least significant bit, 01H address to next higher bit and so on, and finally 07H address to most significant bit); 21H contains bits 08H to 0FH and so on. Hence bit addresses 00H to 7FH belongs to byte address 20H-2FH. The remaining locations in internal RAM must be accessed using their byte addresses only.

The relation between byte address and bit address can be established using the following equation.

$$(\text{Byte address})_H = 20_H + \text{Integer part of } \left[\frac{\text{Bit address (HEX)}}{8}\right] = \left[32\text{D} + \text{Integer part of}\left[\frac{\text{Bit address (HEX)}}{8}\right]\right]\text{H}$$



Byte Address	Bit Address							
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00

Fig. 6.1 *Internal RAM bit-addressable area*

Discussion Question How would a microcontroller know whether to access a bit or byte?

Answer The instruction (opcode of the instruction) that is used will determine whether a byte or a bit is being referenced without confusion. For example,

MOV A, 20H will access byte address 20H, while MOV C, 20H will access the bit address 20H. These two instructions have different op-codes.

Example 6.1

Which byte addresses do the bit addresses 07H, 24H and 4DH belong to?

Solution:

Bit addresses 07H, 24H and 4DH belongs to byte addresses 20H, 24H and 29H respectively. (See Figure 6.1)

6.2.2 Bit-Addressable Special Function Registers

All ports (P0, P1, P2 and P3), A, B, PSW, IP, IE, ACC, SCON, PCON and TCON are bit-addressable. Byte as well as bit addresses of all SFRs are shown in Figure 6.2.

As shown in Figure 6.2, P0 is assigned bit addresses 80H to 87H, P1 is assigned addresses 90H to 97H, (TCON is given addresses 88H-8FH). It should be noted that each port pin may be treated as a separate single bit port, i.e. each pin may be configured as an input or output independently as illustrated in Figure 6.3. Note that pins P1.0, P1.1, P1.2 and P1.4 are configured as an input and at the same time, all other pins of port 1 are used as an output.

Byte Address	Bit Address b7	b6	b5	b4	b3	b2	b1	b0	SFR name
FFH									
F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
E0H	E7	E6	E5	E4	E3	E2	E1	E0	A
D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
B8H	--	--	--	BC	BB	BA	B9	B8	IP
B0H	B7	B6	B5	B4	B3	B2	B1	B0	PORT 3 (P3)
A8H	AF	--	--	AC	AB	AA	A9	A8	IE
A0H	A7	A6	A5	A4	A3	A2	A1	A0	PORT 2 (P2)
99H	*								SBUF
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	PORT 1 (P1)
8DH	*								TH 1
8CH	*								TH 0
8BH	*								TL 1
8AH	*								TL 0
89H	*								TMOD
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
87H	*								PCON
83H	*								DPH
82H	*								DPL
81H	*								SP
80H	87	86	85	84	83	82	81	80	PORT 0 (P0)

SFRs

* Indicates the SFRs which are not bit addressable

Fig. 6.2 *Bit and byte address of SFRs*

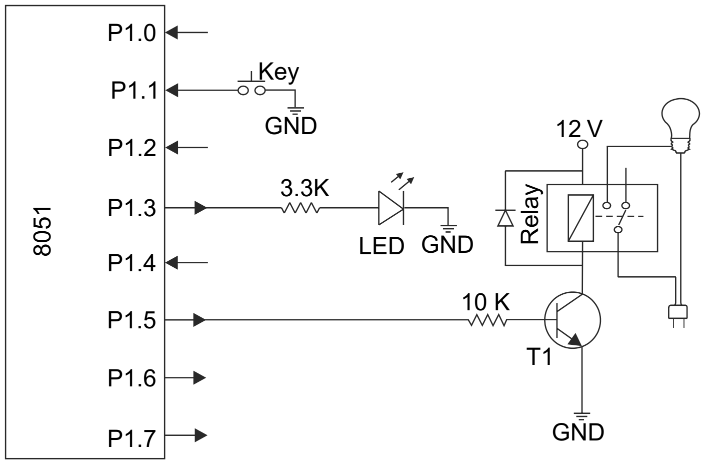


Fig. 6.3 *Independent operation of port pins*

The bits of ports and other SFRs can be accessed either using their addresses or names as given in Table 6.1.

Table 6.1 *SFR bits and their names*

SFR	BIT Name							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
P0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
ACC	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
B	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
PSW	PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
TCON	TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0
IE	IE.7	*IE.6*	*IE.5*	IE.4	IE.3	IE.2	IE.1	IE.0
IP	*IP.7*	*IP.6*	*IP.5*	IP.4	IP.3	IP.2	IP.1	IP.0

The assembler will replace name with appropriate address while assembling the process, for example, instruction CLR P0.0 will be considered as a CLR 80H.

THINK BOX 6.2

Do you observe any pattern in addresses of bit-addressable SFRs?
Yes. If the lower digit (nibble) of the address is 0 or 8 then only that address is bit-addressable.

THINK BOX 6.3

Do you see any relation between the byte address and bit addresses of SFR?
The address of bit 0 (LSB) of any bit-addressable SFR is same as its byte address.

Discussion Question Can we simultaneously configure different pins of the same port as an input and output? If yes, how?

Solution Yes, we can configure some pins of port as an input and others as an output at the same time because all ports are bit-addressable and each pin may be treated independently using bit processing instructions.

For example, let us configure P1.0 and P1.1 as input pins and P1.2 and 1.3 as output. To configure any port pin as an input, we have to write '1' to corresponding port latch bit, and to configure port pin as an output, there is no special requirement, i.e. we can directly provide output data on port pin by simply writing the data in corresponding port latch bit. (Refer Topic 13.1.1 for more details on configuring port as an input and output). Following instructions are needed to configure P1.0 and P1.1 as input pins,

```
        SETB P1.0       // Configure P1.0 as an input pin by writing 1 to P1.0 latch bit
        SETB P1.1       // Configure P1.1 as an input pin by writing 1 to P1.10 latch bit
```

THINK BOX 6.4

Why are none of the ROM locations bit-addressable?
ROM usually contains program codes which are byte oriented; therefore they need not be bit-addressable.

6.3 BIT-PROCESSING INSTRUCTIONS

Bit-processing instructions operate on a bit of bit-addressable RAM area and SFRs. The carry flag is a destination for majority of the bit-processing instructions because it can be easily tested and program flow can be altered using decision making instructions (conditional jump instructions). It is equivalent to a single-bit accumulator.

The carry flag in PSW is considered as Boolean Accumulator.

The format of bit processing instructions with examples are given below.

6.3.1 Instruction using Carry

SETB C	// set the carry flag to 1, C=1
CLR C	// clear the carry flag to 0, C=0
CPL C	// complement the carry flag, C = $\overline{C}$
MOV *bit*, C	// copy status of the carry flag to bit address *bit*, (*bit)* = C
MOV 01H, C	// If C=0, → bit address (01H) =0
MOV C, *bit*	// copy status of bit address *bit* to the carry flag, C= *(bit)*
MOV C, 10H	// if bit address (10H) = 0, → C=0
ANL C, *bit*	// AND operation between C and *bit*, C = C AND *bit*
ANL C, 05H	// If C=0 and (05H) =1, → C= 0 AND 1=0
ANL C, */bit*	// AND op. between C and $\overline{\text{bit}}$, C = C AND $\overline{\text{bit}}$, do not affect status of *bit*
ANL C, /05H	// If C=1 and (05H) =0, → C= 1 AND $\overline{0}$ =1
ORL C, *bit*	// OR operation between C and *bit*, C = C OR *bit*
ORL C, 05H	// If C=0 and (05H) =1, → C= 0 OR 1=1
ORL C, */bit*	// OR op. between C and $\overline{\text{bit}}$, C = C OR $\overline{\text{bit}}$, do not affect status of *bit*
ORL C, /05H	// If C=0 and (05H) =0, → C= 0 AND $\overline{0}$ =1

THINK BOX 6.5

How can we perform addition between two bits?
Assume *x* and *y* are two bits. The Sum= x EX-OR y = $x\bar{y} + \bar{x}y$ and Carry=xy

6.3.2 Other Instructions

SETB *bit*	// set bit address *bit* to1, *(bit)* =1
SETB 00H	// $(00H)_{Bit}=1$
CLR *bit*	// clear bit address *bit* to 0, *(bit)* =0
CLR 00H	// $(00H)_{Bit}=0$
CPL *bit*	// complement bit address *bit*, $(bit) = (\overline{bit})$
CPL 00H	// If $(00H)_{Bit}=1 \rightarrow (00H)_{Bit}=0$

Example 6.2

Write a part of a program to select register bank 1 using both bit as well as byte-processing instructions. Show that use of bit-processing instructions makes program more efficient in terms of memory requirements and speed of execution.

Solution:

Using byte-processing instructions:

```
          MOV PSW, #00001000B          // RS1=0, RS0=1 (3 Bytes, 2 machine cycles)
```

The problem in using the above instruction is that it modifies whole byte, i.e. it disturbs other bits of the PSW register which are not useful in bank selection operation. To preserve the other bits of the PSW, we may use the following two instructions:

```
          ORL PSW, # 08H               // set RS0 bit (3 bytes, 2 machine cycles)
          ANL PSW, #0EFH               // clear RS1 bit (3 bytes, 2 machine cycles)
                                       // Total 6 bytes and 4 machine cycles
```

Using bit-processing instructions,

```
          CLRB PSW.4                   // clear RS1 bit (2 bytes, 1 machine cycle)
          SETB PSW.3                   // set RS0 bit (2 bytes, 1 machine cycle)
                                       // Total 4 bytes and 2 machine cycles
```

It can be clearly seen that bit-processing instructions will make the program more efficient in terms of:

(i) Memory requirements, because it requires only 4 bytes compared to 6 bytes required by byte-processing instructions

(ii) Execution speed, because it requires only 2 machine cycles compared to 4 machine cycles required by byte-processing instructions

Example 6.3

Write instruction(s) to perform following operations:

(i) Read status of port pin P1.0 and store it in to bit D2 of byte address 22H as well as LSB of 2FH.

(ii) Complement the status of pins P1.0, P1.1 and P1.2 without affecting other bits.

(iii) Copy the status of port pin P1.0 to P0.5 as well as P0.6.

(iv) Set bit addresses 10H and 11H to 1.

(v) Set TR1 bit (timer1 run).

Solution:

```
(i)      MOV C, P1.0     //  read  P1.0  in to carry, C= P1.0
         MOV 12H, C      // (12H) = C, bit D2 of byte address 22H has bit address 12H
         MOV 78H, C      // (78H) = C, LSB of byte address 2FH has bit address 78H
```

Note that when the port bit is used as a source operand, the status of the port pin is read.

```
(ii)     CPL P1.0        // complement  P1.0
         CPL P1.1        // complement  P1.1
         CPL P1.2        // complement  P1.2
```

Note that when the port bit is used as a destination (or source as well as destination) operand, the status of port latch is modified (or read–modified and written back).

```
(iii)    MOV C, P1.0     // read  P1.0  into carry, C= P1.0
         MOV P0.5, C     // send status of carry into P0.5
         MOV P0.6, C     // send status of carry into P0.6
```

```
(iv)    SETB 10H
        SETB 11H
(v)     SETB TR1
```

Example 6.4

Write a program to read status of the pin P2.1 and send its complement to the pin P1.0.

Solution:

```
SETB P2.1        // configure P2.1 as an input pin
MOV C, P2.1      // read status of pin P2.1 into carry flag
CPL C            // complement the status
MOV P1.0, C      // send complement of the status to the pin P1.0
```

Example 6.5

Show the status of the carry flag and bit address 10H after execution of each of the following instructions.

```
SETB C
CLR 01H
ORL C, /01H
MOV 10H, C
```

Solution:

```
SETB C        // C=1
CLR 01H       // (01H) =0, clear second bit (D1) of byte address 20H
ORL C, /01H   // C  OR  0̅ = 1    OR   1 =1
MOV 10H, C    // (10H) = C=1, bit D0 of byte address 22H
```

Both, bit address 10H and C will be set to 1 after execution of given instructions.

Example 6.6

Find the status of carry flag and other memory locations (involved in instructions) after execution of each of the following instructions.

```
CLR C
MOV 20H, #0FH
MOV C, 00H
ANL C, /07H
ORL C, 01H
CPL 06H
```

Solution:

```
CLR C            // C = 0
MOV 20H, #0FH    // (20H) = 0FH
MOV C, 00H       // C = 1
ANL C, /03H      // C = 0, (03H) =1, bit address 03H is 4th bit of 20H
ORL C, 01H       // C = 1, if (01H) =1, bit address 01H is 2nd bit of 20H
CPL 06H          // (06H) =1, bit address 06H is 7th bit of 20H
```

Example 6.7

Assume that a switch is connected to pin P1.1 and the LED is connected to pin P1.2. Write a program to read the status of switch and send this status to LED.

Solution:

```
SETB P1.1        // configure P1.1 as an input pin
MOV C, P1.1      // read status of pin P1.1 into carry flag
MOV P1.2, C      // send contents of carry flag to P1.2 (LED)
```

The above program fragment can be made more readable by using BIT directive. The BIT directive is used to assign a name to bit-addressable RAM locations. For example, P1.1 pin may be given a name SWITCH and pin P1.2 may be given a name LED using BIT directive to make program more readable and easy to understand. This is shown below.

```
SWITCH   BIT   P1.1     // assign name SWITCH to P1.1
LED    BIT   P1.2       // assign name LED to P1.2

SETB SWITCH             // configure P1.1 as an input pin
MOV C, SWITCH           // read status of pin switch (P1.1) into carry flag
MOV LED, C              // send contents of carry flag to LED (P1.2)
```

Note the program is now easy to understand. Moreover, use of BIT directive also allows easier modification of the program, i.e. we have to change RAM address only once while assigning the name to the address using BIT directive and corresponding change will be reflected in all places where the name is used. For example, if we change the first line of the above program as 'SWITCH   BIT   P2.1', we will get P1.1 replaced with P2.1 at all the places where the name SWITCH is used. (Refer Section 12.1.2 for more details of BIT directive or BIT data type)

Example 6.8

Realize single bit EX-OR instruction using other bit processing logical instructions.

Solution:

Assume that we want to perform the EX-OR operation between the two bits stored at bit addresses 00H (x) and 01H (y) and store result at address 02H.

We know that the EX-OR operation for two bits x and y is given as x EX-OR y= $x\bar{y} + \bar{x}y$

```
MOV C, 00H
ANL C, /01H           // xȳ
MOV 02H, C            // save partial result xȳ into 02H
MOV C, 01H
ANL C, /00H           // x̄y
ORL C, 02H            // xȳ + x̄y
MOV 02H, C            // save result into 02H
```

6.3.3 Conditional Jump Instructions

These instructions are discussed in more detail in Chapter 7.

```
JNC rel          // jump to address rel if C=0 (jump if no carry)
JC rel           // jump to address rel if C=1 (jump if carry)
JB bit, rel      // jump to address rel if bit =1 (jump if bit)
JNB bit, rel     // jump to address rel if bit =0 (jump if no bit)
JBC bit, rel     // jump to address rel if bit = 1, and then clear bit (jump if bit, then clear)
```

THINK BOX 6.6

An instruction 'MOV C, *bit*' requires 1 machine cycle. What do you think about machine cycles required by an instruction 'MOV *bit*, C'?
2 Machine cycles!

THINK BOX 6.7

Why does there exist bit-processing instructions in a microcontroller, which usually works on the bytes?
In many real-world machine-control applications, we may need to manipulate (read/write/modify) only one bit (or few bits) at a time. Bit processing instructions facilitate these operations easily.

Summary of Bit-Processing Instructions

Bit-processing instructions are summarized in Table 6.2.

Table 6.2 *Bit-processing instructions with examples*

Mnemonics	Operation	Addressing Modes			
		Direct	Indirect	Register	Immediate
ANL C, **BIT**	C= C AND **BIT**	ANL C, **bit** ANL C, **10H**			
ANL C, **/BIT**	C= C AND / **BIT**	ANL C, **/bit** ANL A, **/12H**			
ORL C, **BIT**	C= C OR **BIT**	ORL C, **bit** ORL C, **12H**			
ORL C, **/BIT**	C= C OR / **BIT**	ORL C, **/bit** ORL C, **/7FH**			
MOV C, **BIT**	C= **BIT**	MOV C, **bit** MOV C, **20H**			
MOV BIT, **C**	BIT= **C**	MOV bit, **C** MOV 10H, **C**			
CLR **C**	C= **0**	CLR C			
CLR **BIT**	BIT= **0**	CLR **bit** CLR **12H**			
SETB **C**	C= **1**	SETB C			
SETB **BIT**	BIT= **1**	SETB **bit** SETB **10H**			
JC **rel**	Jump if C = 1	JC **rel** JC **HERE**			
JNC **rel**	Jump if C = 0	JNC **rel** JNC **HERE**			
JB BIT, **rel**	Jump if BIT = 1	JB BIT, **rel** JB 12H, **HERE**			
JNB BIT, **rel**	Jump if BIT = 0	JNB BIT, **rel** JNB 10H, **NEXT**			
JBC BIT, **rel**	Jump if BIT = 1; CLR BIT	JBC BIT, **rel** JBC 10H, **NEXT**			
CPL **C**	C = NOT C	CPL C			
CPL **BIT**	BIT = NOT BIT	CPL **bit** CPL **12H**			

POINTS TO REMEMBER

- The 8051 contains a complete Boolean (single bit) processor and its instruction set is optimized for the single-bit operations. It supports SET, CLEAR, COMPLEMENT, AND and OR single bit operations.
- Faster execution, less memory requirements and program readability are the advantages offered by the bit processing instructions.
- The 8051 has a bit-addressable area of 16 bytes from byte addresses 20H to 2FH in internal RAM, forming a total of 128 (16x8) addressable bits. Addressable bits are assigned bit addresses from 00H to 7FH.
- All ports (P0, P1, P2 and P3), A, B, PSW, IP, IE, ACC, SCON, PCON and TCON are bit-addressable.
- The carry flag in PSW is considered as Boolean Accumulator. The carry flag is destination for a majority of bit-processing instructions because it can be easily tested and program flow can be altered using decision making instructions.

OBJECTIVE QUESTIONS

1. Which of the following SFRs is bit-addressable?
   (a) SP (b) P2 (c) TMOD (d) SBUF
2. Bit-addressable locations of the 8051 include
   (a) 16 internal RAM bytes (b) majority of SFRs
   (c) I/O ports (d) all of the above
3. The 8051 has bit-level instructions for
   (a) AND (b) OR (c) complement (d) all
4. Bit-addressable internal RAM has address range of
   (a) 10H to 1FH (b) 20H to 2FH (c) 30H to 3FH (d) 40H to 4FH
5. Which of the following instructions is not a bit-processing instruction?
   (a) CLR C (b) CLR A (c) CPL 10H (d) CLR 10H
6. Which of the following SFRs are bit-addressable?
   (a) P0 (b) DPL (c) B (d) TMOD
7. The 8051 does not have bit-level instructions for
   (a) AND (b) OR (c) Complement (d) EX-OR
8. Which of the following instructions are invalid?
   (a) SETB C (b) SETB B (c) ANL C, #00H (d) MOV C, 10H
9. If (20H)=00H and CY=1 the contents of internal RAM address 20H after execution of following instructions will be,
   ```
   ORL C, 00H
   MOV 00H, C
   ```
   (a) 00H (b) 01H (c) 20H (d) FFH
10. If (20H) = FFH and CY=1 the contents of internal RAM address 20H after execution of following instructions will be,
   ```
   ANL C, 00H
   CPL C
   MOV 00H, C
   ```
   (a) 00H (b) 01H (c) FFH (d) FEH

Answers to Objective Questions

1. (b)	3. (d)	5. (b)	7. (d)	9. (b)
2. (d)	4. (b)	6. (a), (c)	8. (b), (c)	10. (d)

REVIEW QUESTIONS WITH ANSWERS

1. **List the advantages offered by the bit-processing instructions.**

A. Bit-processing instructions makes program more efficient in terms of speed of execution and code density, i.e. faster execution of program, and less memory required by the program. Furthermore, program listing and development becomes simple.

2. **What is meant by a Boolean processor?**

A. It is a circuit capable of handling single-bit operations. It is equivalent to single bit ALU.

3. **All ports of the 8051 are bit-addressable. True/False.**

A. True.

4. **Which of the following registers are bit-addressable?**
   A, B, R0, DPL, TL0, TCON

A. A, B and TCON are bit-addressable.

5. **In which address space do bit addresses 00H-7FH and 80H-FFH belong?**

A. 00H-7FH belongs to internal RAM at byte addresses 20H- 2FH.
80H-FFH belongs to SFRs.

6. Carry for the bit-processing instructions is equivalent to single-bit accumulator. True/False.

A. True.

7. Write instructions to save the status of pin P1.0 at the bit address 10H.

A.
```
MOV C, P1.0        or              CLR 10H
MOV 10H, C                         JB P1.0, HIGH
                                   SJMP NEXT
                             HIGH: SETB 10H
                             NEXT: …
```

8. Write two instructions to clear the carry flag.

A. CLR C or CLR 0D7H

9. State the validity of the following instructions.

(a) SETB C (b) SETB B (c) ANL C, 00H
(d) ANL C, #00H (e) ANL 00H, C (f) MOV C, 10H
(g) MOV C, @R0

A. (a) Valid.
(b) Invalid, SETB is used to set bits, while B is a byte.
(c) Valid.
(d) Invalid, immediate addressing is not supported by bit processing instructions.
(e) Invalid, C is always destination in bit processing AND operations.
(f) Valid.
(g) Invalid, Indirect addressing is not supported by bit processing instructions.

10. Write the instruction to configure port pin P1.0 as an input.

A. SETB P1.0 (by writing 1 to port bit latch).

11. Each pin of a port can be programmed independently as an input or output. True/False.

A. True.

12. All SFRs of the 8051 are bit-addressable. True/False.

A. False.

13. Which byte addresses does the bit addresses 06H and 7FH belong to?

A. 20H and 2FH respectively.

14. What is the address of MSB of port1?

A. 97H.

EXERCISE

1. List all bit-addressable SFRs and write bit addresses assigned to them.
2. Write a program to EX-OR first and second (D0 and D1) bits of Accumulator and save the result in third bit of accumulator.
3. How can the status of C and OV flag be monitored?
4. What is the range of bit addresses assigned to SFRs?
5. Write instructions to toggle pin P1.0 continuously if P2.0 is high, otherwise clear P1.0.
6. How can the status of port pins be monitored?
7. Write a program to generate a rectangular wave of 75% duty cycle on pin P1.0.
8. Show with suitable example that use of bit-processing instructions makes program more efficient in terms of speed of execution and memory requirements.
9. When port bit is used as a destination operand, the status of port latch is modified. True/false.
10. What is common use of ANL C, /*bit* and ORL C, /*bit* instructions?