Shift and Rotate Instructions

There are several shift and rotate instructions; these are listed in Table 2.10 and diagrammed in Fig. 2.12. Be sure to notice that the rotated quantity can be an 8- or a 16-bit CPU register or memory location.

The main difference between a shift and a rotate is that the shifted bits "fall off" the end of the register, whereas the rotated bits "wrap around." Within the shift group of instructions there are both *arithmetic* (SAL and SAR) and *logical* (SHL and SHR) shift instructions.

The arithmetic shifts operate so that the sign bit (in bit 7 or 15) does not change

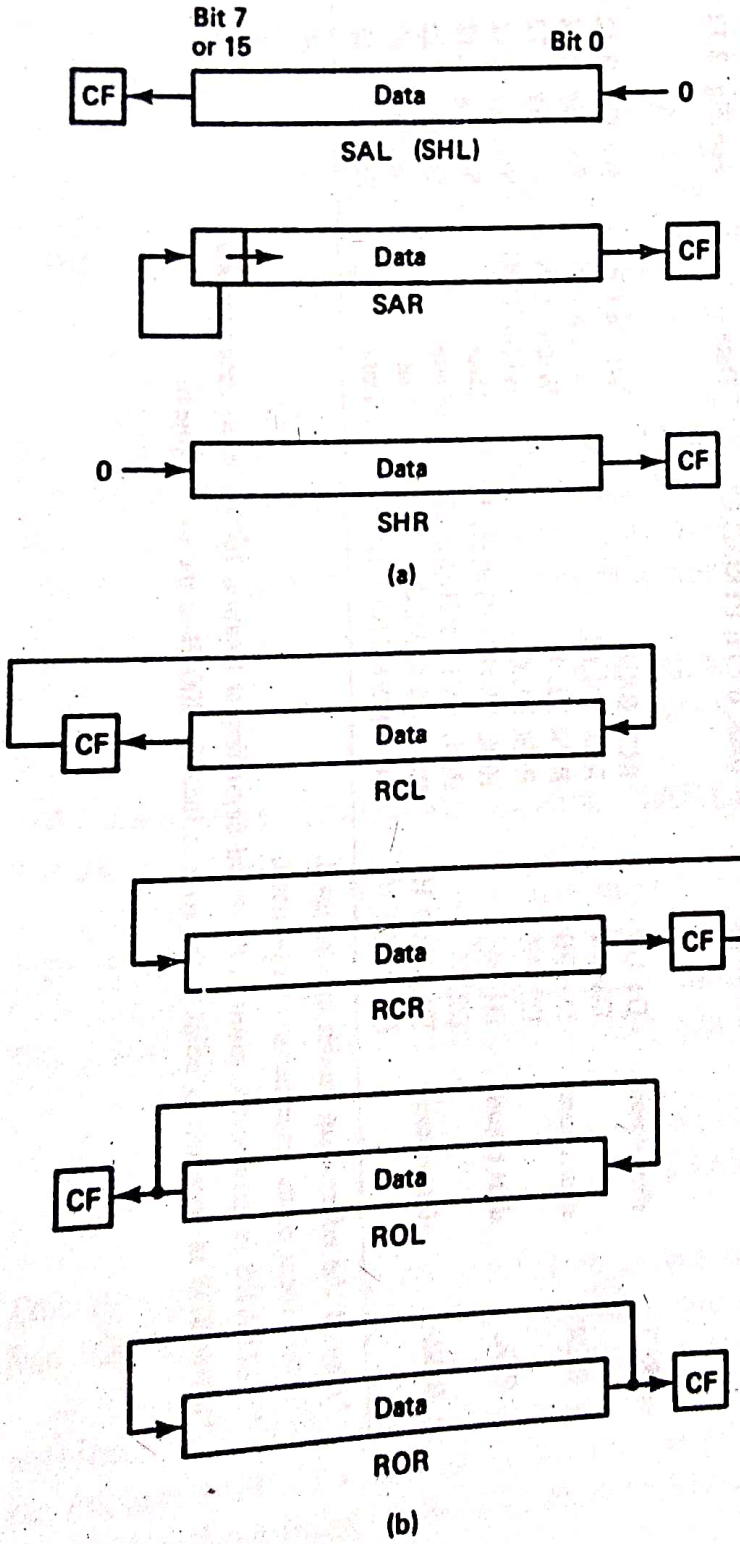

Figure 2.12 The 8086/88 has (a) three shift instructions and (b) four rotate instructions.

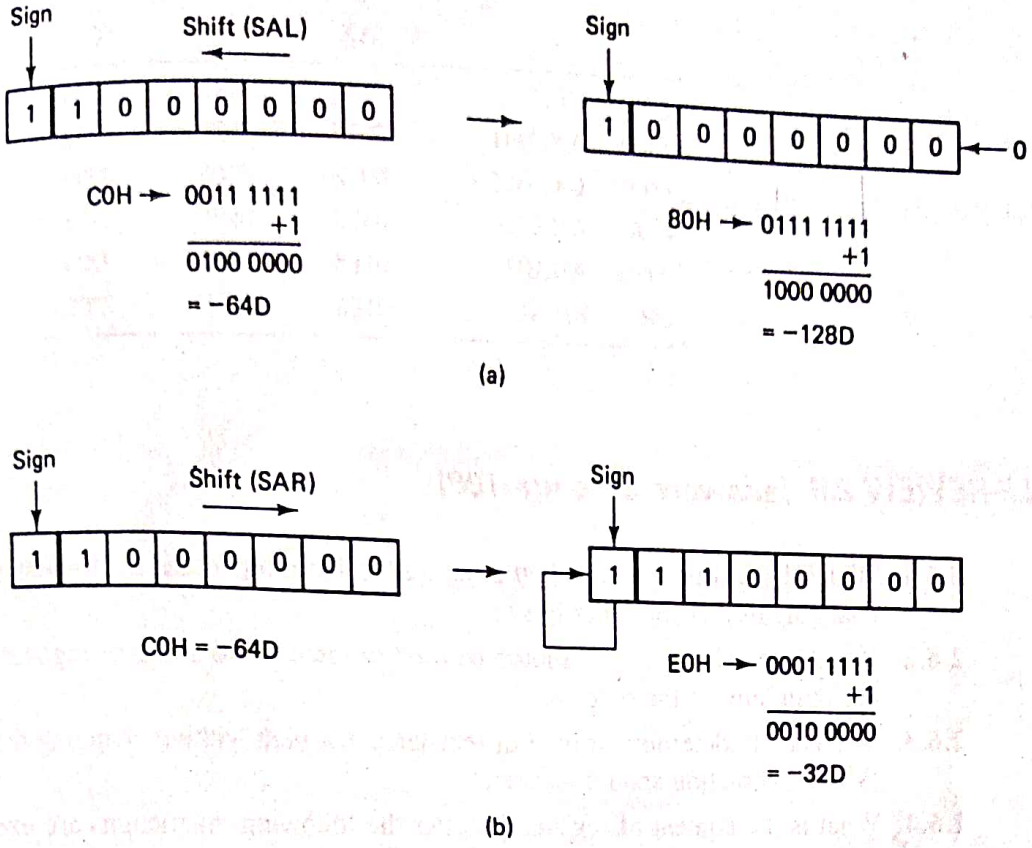

Figure 2.13 (a) The SAL instruction *multiplies* the signed binary number by a power of 2. (b) The SAR instruction *divides* the signed binary number by a power of 2.

when the shift occurs. Figure 2.13 shows the effect of the SAL and SAR instructions on the data byte COH. As you can see, the SAL instruction in effect multiplies the data by 2 (maintaining the correct sign), and SAR divides the data by 2. The overflow flag (OF) will be set if the shifted quantity exceeds the limits for an 8- or 16-bit register (+127 to −128 for bytes and +32767 to −32768 for words).

The shift and rotate instructions can be repeated up to 255 times by loading register CL with the desired count. For example, to rotate the contents of register DX left five times through the carry, the following instructions should be given:

```
MOV  CL,5
RCL  DX,CL
```

Example 2.12

Determine the contents of registers AX, BX, and CX after the following program is run.

```
MOV  CL,3
MOV  AX,7FH
MOV  BX,0505H
ROL  AX,CL
AND  AH,BH
OR   BL,AL
```

Solution The results are as follows.

		AX	BX	CX
MOV	CL,3	????	????	??03
MOV	AX,7FH	007F	????	??03
MOV	BX,0505H	007F	0505	??03
ROL	AX,CL	03F8	0505	??03
AND	AH,BH	01F8	0505	??03
OR	BL,AL	01F8	05FD	??03

SELF-REVIEW 2.6 (answers on page 108)

2.6.1. Which logical instruction should be used to force bits 0 and 1 of register D changing any of the other bits?

2.6.2. Which logical instruction should be used to force bits 0 and 1 of register DI changing any of the other bits?

2.6.3. You need to determine if bit 3 of register AH is high without changing the c Which instruction should be used?

2.6.4. What is the content of register BL after the following instructions are execu

```
MOV  BL,0B2H
MOV  CL,2
SAR  BL,CL
```

2.6.5. Which two instructions will have the effect of dividing register BX by 8?

.7 ARITHMETIC INSTRUCTIONS

The arithmetic instructions give the microprocessor its computational ca unlike earlier 8-bit processors, these instructions are not limited to addition a of 8-bit numbers in the accumulator. The 8086/88 can add and subtract general CPU registers, and using certain dedicated reg numbers.

using the

```
ADD  CX,AX      ;CX ← CX + AX
ADC  DX,BX      ;DX ← DX + BX + CF
```

The first instruction does not include the carry, as there is no carry in at this point. If the addition of AX and CX sets CF, the second addition will add this to the sum of DX and BX.

The SUB (subtract) and SBB (subtract with borrow) instructions work similarly, with CF representing the borrow condition. When adding or subtracting one from a memory pointer or counter variable, the INC (increment) and DEC (decrement) instructions should be used. Note that the contents of a memory location can also be specified as the destination operand for these instructions.

The NEG (negate) instruction forms the 2's complement of the destination operand, effectively reversing its sign. This is done by subtracting the destination operand from 0.

Example 2.13

Determine the value of AL and the value of the flags following the instruction sequence

```
MOV  AL,5
NEG  AL
```

Solution AL → 00000000 − 00000101 = 11111011 = FBH = −5. The flags are affected as follows:

```
CF = 1    ;NEG sets CF except when the operand is 0
PF = 0    ;FBH has an odd number of logic 1's
AF = 1    ;There is a borrow out of bit 4
ZF = 0    ;The result is not 0
SF = 1    ;Bit 7 is set
OF = 0    ;There is no overflow condition
```

The CMP (compare) instruction is useful for determining the relative size of tw operands. Normally, it is followed by a conditional jump instruction such as "jump equal" or "jump if greater than or equal."

Multiplication and Division Instructions

Multiplication and division can be performed on signed or unsigned numbers as indicat in Table 2.12. The source operand can be a memory location or a CPU register, but t destination ... AX (and DX for 32-bit results). The register usa