Arithmetic and Logic Instructions

Chapter-5, B.B Brey

Addition, Subtraction, Multiplication, Division, Comparison, Negative, Increment, and Decrement.

AND, OR, Exclusive-OR, NOT, Shifts, Rotates and logical Compare (TEST).

5-1 Addition, Subtraction, and Comparison

- **Addition:**
  - 8-, 16-, 32-bit binary addition (ADD)
    - 32-bit registers are available only for 80386 and above
    - 0000 00dw oorrrmmm disp — ADD reg/mem, reg/mem
    - 1000 00sw oo000mem disp imm — ADD reg/mem, imm
  - increment (INC)
  - add-with-carry (ADC)
  - XADD (80486 – Pentium 4): exchange and add
  - BCD and ASCII (5-3)
- Addressing modes for ADD
  - almost all mentioned before
    - over 32,000 variations
    - except memory-to-memory and segment register
- Part of the flag register contents changes
  - rightmost bits of the flag register
    - sign, zero, carry, auxiliary carry, parity, and overflow

Addition Instructions

TABLE 5–1 Addition instructions.

Assembly Language	*Operation*
ADD AL,BL	AL = AL + BL
ADD CX,DI	CX = CX + DI
ADD EBP,EAX	EBP = EBP + EAX
ADD CL,44H	CL = CL + 44H
ADD BX,245FH	BX = BX + 245FH
ADD EDX,12345H	EDX = EDX + 00012345H
ADD [BX],AL	AL adds to the contents of the data segment memory location addressed by BX with the sum stored in the same memory location
ADD CL,[BP]	The byte contents of the stack segment memory location addressed by BP add to CL with the sum stored in CL
ADD AL,[EBX]	The byte contents of the data segment memory location addressed by EBX add to AL with the sum stored in AL
ADD BX,[SI + 2]	The word contents of the data segment memory location addressed by the sum of SI plus 2 add to BX with the sum stored in BX
ADD CL,TEMP	The byte contents of the data segment memory location TEMP add to CL with the sum stored in CL
ADD BX,TEMP[DI]	The word contents of the data segment memory location addressed by TEMP plus DI add to BX with the sum stored in BX
ADD [BX + DI],DL	DL adds to the contents of the data segment memory location addressed by BX plus DI with the sum stored in the same memory location
ADD BYTE PTR [DI],3	A 3 adds to the byte contents of the data segment memory location addressed by DI
ADD BX,[EAX + 2*ECX]	The word contents of the data segment memory location addressed by the sum of 2 times ECX plus EAX add to BX with the sum stored in BX

Examples – Register, Immediate, Mem2Reg

- Register Addition

EXAMPLE 5–1

```
                ;A procedure that sums AX, BX, CD and DX;
                ;the result is returned in AX.
                ;
0000            ADDS    PROC    NEAR
0000  03 C3             ADD     AX,BX
0002  03 C1             ADD     AX,CX
0004  03 C2             ADD     AX,DX
0006  C3                RET
0007            ADDS    ENDP
```

- Immediate Addition

Z = 0 (result not zero) S = 0 (result positive)
C = 0 (no carry) P = 0 (odd parity)
A = 0 (no half-carry) O = 0 (no overflow)

EXAMPLE 5–2

```
0006  B2 12       MOV   DL,12H
0008  80 C2 33    ADD   DL,33H
```

- Memory-to-Register Addition

EXAMPLE 5–3

```
                ;A procedure that sums data in
                ;locations NUMB and NUMB+1;
                ;the result is returned in AX.
0000            SUMS    PROC    NEAR
0000  BF 0000 R         MOV     DI,OFFSET NUMB ;address NUMB
0003  B0 00             MOV     AL,0           ;clear sum
0005  02 05             ADD     AL,[DI]        ;add NUMB
0007  02 45 01          ADD     AL,[DI+1]      ;add NUMB+1
000A  C3                RET
000B            SUMS    ENDP
```

Examples – Array Addition

- Array Addition
  - 8-bit

EXAMPLE 5–4

```
                        ;A procedure that sums ARRAY elements 3, 5, and 7;
                        ;the result is returned in AL.
                        ;Note this procedure destroys the contents of SI.
0000                    SUM     PROC    NEAR
0000  B0 00                     MOV     AL,0                    ;clear sum
0002  BE 0003                   MOV     SI,3                    ;address element 3
0005  02 84 0002 R              ADD     AL,ARRAY[SI]            ;add element 3
0009  02 84 0004 R              ADD     AL,ARRAY[SI+2]          ;add element 5
000D  02 84 0006 R              ADD     AL,ARRAY[SI+4]          ;add element 7
0011  C3                        RET
0012                    SUM     ENDP
```

  - scaled-index

EXAMPLE 5–5

```
                        ;A procedure that sums ARRAY elements 3, 5 and 7;
                        ;the result is returned in AX.
                        ;Note that the contents of registers EBX and ECX are
                        ;destroyed.
0000                    SUM     PROC    NEAR
0000  66| BB 00000000 R        MOV     EBX,OFFSET ARRAY        ;address ARRAY
0006  66| B9 00000003           MOV     ECX,3                   ;address element 3
000C  67& 8B 04 4B              MOV     AX,[EBX+2*ECX]          ;get element 3
0010  66| B9 00000005           MOV     ECX,5                   ;address element 5
0016  67& 03 04 4B              ADD     AX,[EBX+2*ECX]          ;add element 5
001A  66| B9 00000007           MOV     ECX,7                   ;address element 7
0020  67& 03 04 4B              ADD     AX,[EBX+2*ECX]          ;add element 7
0024  C3                        RET
0025                    SUM     ENDP
```

Increment Addition (INC)

- Increment Addition
  - opcode = 1111 111w
  - opcode = 0100 0rrr

```
40  INC AX
41  INC CX
42  INC DX
43  INC BX
44  INC SP
45  INC BP
46  INC SI
47  INC DI

48  DEC AX
49  DEC CX
4A  DEC DX
4B  DEC BX
4C  DEC SP
4D  DEC BP
4E  DEC SI
4F  DEC DI
```

EXAMPLE 5–6

```
                ;A procedure that sums NUMB and NUMB+1;
                ;the result is returned in AL.
                ;Note that the contents of DI are destroyed.
                ;
0000            SUMS    PROC    NEAR

0000  BF 0000 R         MOV     DI,OFFSET NUMB ;address NUMB
0003  B0 00             MOV     AL,0           ;clear sum
0005  02 05             ADD     AL,[DI]        ;add NUMB
0007  47                INC     DI             ;address NUMB+1
0008  02 05             ADD     AL,[DI]        ;add NUMB+1
000A  C3                RET
000B            SUMS    ENDP
```

TABLE 5–2 Increment instructions.

Assembly Language	*Operation*
INC BL	BL = BL + 1
INC SP	SP = SP + 1
INC EAX	EAX = EAX + 1
INC BYTE PTR [BX]	Adds 1 to the byte contents of the data segment memory location addressed by BX
INC WORD PTR [SI]	Adds 1 to the word contents of the data segment memory location addressed by SI
INC DWORD PTR [ECX]	Adds 1 to the doubleword contents of the data segment memory location addressed by ECX
INC DATA1	Increments the contents of data segment memory location DATA1

Addition-with-Carry (ADC)

- ADC = ADD + carry flag bit (C)
  - opcode: 0001 00dw
  - link two 16-bit additions into one 32-bit addition
  - add numbers that are wider than 16 bits in 8086-80286 or wider than 32 bits in 80386-Pentium 4

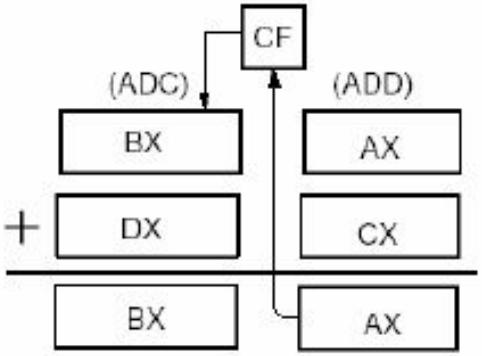
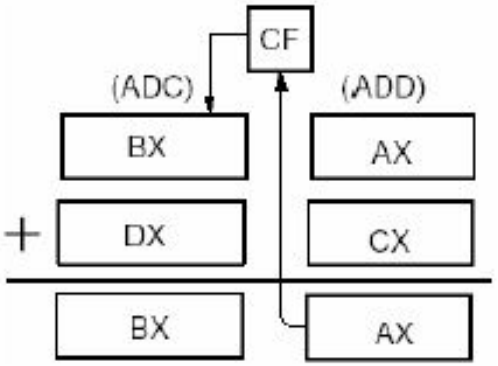


TABLE 5–3 Add-with-carry instructions.

Assembly Language	*Operation*
ADC AL,AH	AL = AL + AH + carry
ADC CX,BX	CX = CX + BX + carry
ADC EBX,EDX	EBX = EBX + EDX + carry
ADC DH,[BX]	The byte contents of the data segment memory location addressed by BX add to DH with carry with the sum stored in DH
ADC BX,[BP + 2]	The word contents of the stack segment memory location addressed by BP plus 2 add to BX with carry with the sum stored in BX
ADC ECX,[EBX]	The doubleword contents of the data segment memory location addressed by EBX add to ECX with carry with the sum stored in ECX

ADC Example



EXAMPLE 5–8

```
                        ;A procedure that sums EBX-EAX and EDX-ECX;
                        ;the result is returned in EBX-EAX.
                        ;
0000                    SUM64   PROC    NEAR

0000 66| 03 C1                 ADD     EAX,ECX
0003 66| 13 DA                 ADC     EBX,EDX
0006 C3                         RET

0007                    SUM64   ENDP
```

EXAMPLE 5–7

```
                    ;A procedure that sums BX-AX and DX-CX;
                    ;the result is returned in BX-AX.
                    ;
0000                SUM32   PROC    NEAR

0000  03 C1                 ADD     AX,CX
0002  13 DA                 ADC     BX,DX
0004  C3                   RET

0005                SUM32   ENDP
```

Subtraction

- Subtraction: opposite function of Addition
  - 0001 01dw oorrrmmm disp — SUB reg/mem, reg/mem
  - 1000 00sw oo101mmm disp immediate — SUB reg/mem, imm
  - 0010 110w immediate — SUB AL/AX/EAX, imm

TABLE 5–4 Subtraction instructions.

Assembly Language	*Operation*
SUB CL,BL	CL = CL – BL
SUB AX,SP	AX = AX – SP
SUB ECX,EBP	ECX = ECX – EBP
SUB DH,6FH	DH = DH – 6FH
SUB AX,0CCCCH	AX = AX – CCCCH
SUB ESI,2000300H	ESI = ESI – 2000300H
SUB [DI],CH	Subtracts the contents of CH from the contents of the data segment memory location addressed by DI
SUB CH,[BP]	Subtracts the byte contents of the stack segment memory location addressed by BP from CH
SUB AH,TEMP	Subtracts the byte contents of the data segment memory location TEMP from AH
SUB DI,TEMP[ESI]	Subtracts the word contents of the data segment memory location addressed by TEMP plus ESI from DI
SUB ECX,DATA1	Subtracts the doubleword contents of the data segment memory location addressed by DATA1 from ECX

Examples – Register, Immediate

- Register Subtraction
- Immediate Subtraction

Z = 0 (result not zero) S = 1 (result negative)
C = 1 (borrow) P = 1 (even parity)
A = 1 (half-borrow) O = 0 (no overflow)

EXAMPLE 5–9

```
0000  2B D9          SUB   BX,CX
0002  2B DA          SUB   BX,DX
```

EXAMPLE 5–10

```
0000  B5 22          MOV   CH,22H
0002  80 ED 44       SUB   CH,44H
```

- Decrement Subtraction (DEC)

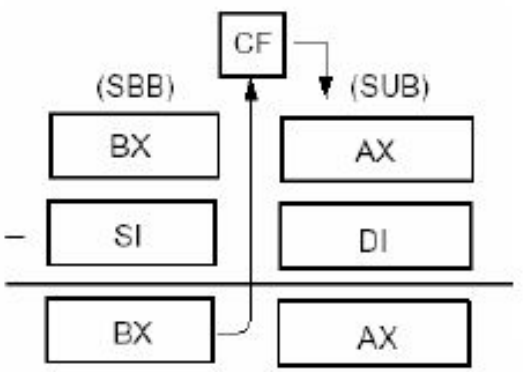
```
48  DEC AX
49  DEC CX
4A  DEC DX
4B  DEC BX
4C  DEC SP
4D  DEC BP
4E  DEC SI
4F  DEC DI
```

TABLE 5–5 Decrement instructions.

Assembly Language	*Operation*
DEC BH	BH = BH – 1
DEC CX	CX = CX – 1
DEC EDX	EDX = EDX – 1
DEC BYTE PTR [DI]	Subtracts 1 from the byte contents of the data segment memory location addressed by DI
DEC WORD PTR[BP]	Subtracts 1 from the word contents of the stack segment memory location addressed by BP
DEC DWORD PTR[EBX]	Subtracts 1 from the doubleword contents of the data segment memory location addressed by EBX
DEC NUMB	Subtracts 1 from the contents of the data segment memory location NUMB

Subtraction-with-Borrow (SBB)

- SBB = SUB – carry flag bit (C)



EXAMPLE 5–11

```
0004   2B C7            SUB   AX,DI
0006   1B DE            SBB   BX,SI
```

TABLE 5–6 Subtraction-with-borrow instructions.

Assembly Language	*Operation*
SBB AH,AL	AH = AH – AL – carry
SBB AX,BX	AX = AX – BX – carry
SBB EAX,ECX	EAX = EAX – ECX – carry
SBB CL,2	CL = CL – 2 – carry
SBB BYTE PTR[DI],3	Both a 3 and carry subtract from the contents of the data segment memory location addressed by DI
SBB [DI],AL	Both AL and carry subtract from the data segment memory location addressed by DI
SBB DI,[BP + 2]	Both carry and the word contents of the stack segment memory location addressed by the sum of BP and 2 subtract from DI
SBB AL,[EBX + ECX]	Both carry and the byte contents of the data segment memory location addressed by the sum of EBX and ECX subtract from AL

Comparison

- Comparison (CMP)
  - subtraction that
    - changes only the flag bits
    - destination operand never changes
  - normally followed by a conditional jump
- Compare and Exchange (CMPXCHG)
  - CMPXCHG CX, DX
    - AX = (CX == AX) ? DX : CX;
    - if CX = AX then AX = DX otherwise, AX = CX
    - 80486 – Pentium 4

EXAMPLE 5–12

```
0000 3C 10 CMP  AL,10H ;compare with 10H
0002 73 1C JAE  SUBER  ;if 10H or above
```

TABLE 5–7 Comparison instructions.

Assembly Language	*Operation*
CMP CL,BL	CL – BL
CMP AX,SP	AX – SP
CMP EBP,ESI	EBP – ESI
CMP AX,2000H	AX – 2000H
CMP [DI],CH	CH subtracts from the contents of the data segment memory location addressed by DI
CMP CL,[BP]	The byte contents of the stack segment memory location addressed by BP subtract from CL
CMP AH,TEMP	The byte contents of the data segment memory location TEMP subtract from AH
CMP DI,TEMP[BX]	The word contents of the data segment memory location addressed by the sum of TEMP plus BX subtract from DI
CMP AL,[EDI + ESI]	The byte contents of the data segment memory location addressed by the sum of EDI plus ESI subtract from AL

5-2 Multiplication and Division

- Multiplication

multiplication instructions

	Assembly Language	*Operation*
TABLE 5–8 8-bit	MUL CL	AL is multiplied by CL; the unsigned product is in AX
	IMUL DH	AL is multiplied by DH; the signed product is in AX
	IMUL BYTE PTR[BX]	AL is multiplied by the byte contents of the data segment memory location addressed by BX; the signed product is in AX
	MUL TEMP	AL is multiplied by the byte contents of the data segment memory location addressed by TEMP; the unsigned product is in AX
TABLE 5–9 16-bit	MUL CX	AX is multiplied by CX; the unsigned product is in DX–AX
	IMUL DI	AX is multiplied by DI; the signed product is in DX–AX
	MUL WORD PTR[SI]	AX is multiplied by the word contents of the data segment memory location addressed by SI; the unsigned product is in DX–AX
TABLE 5–10 32-bit	MUL ECX	EAX is multiplied by ECX; the unsigned product is in EDX–EAX
	IMUL EDI	EAX is multiplied by EDI; the signed product is in EDX–EAX
	MUL DWORD PTR[ECX]	EAX is multiplied by the doubleword contents of the data segment memory location addressed by ECX; the unsigned product is in EDX–EAX

• Division

Division

TABLE 5–11 8-bit

Assembly Language	*Operation*
DIV CL	AX is divided by CL; the unsigned quotient is in AL and the remainder is in AH
IDIV BL	AX is divided by BL; the signed quotient is in AL and the remainder is in AH
DIV BYTE PTR[BP]	AX is divided by the byte contents of the stack segment memory location addressed by BP; the unsigned quotient is in AL and the remainder is in AH

TABLE 5–12 16-bit

DIV CX	DX–AX is divided by CX; the unsigned quotient is in AX and the remainder is in DX
IDIV SI	DX–AX is divided by SI; the signed quotient is in AX and the remainder is in DX
DIV NUMB	AX is divided by the contents of the data segment memory location NUMB; the unsigned quotient is in AX and the remainder is in DX

TABLE 5–13 32-bit

DIV ECX	EDX–EAX is divided by ECX; the unsigned quotient is in EAX and the remainder is in EDX
DIV DATA2	EDX–EAX is divided by the doubleword contents of data segment memory location DATA2; the unsigned quotient is in EAX and the remainder is in EDX
IDIV DWORD PTR[EDI]	EDX–EAX is divided by the doubleword contents of the data segment memory location addressed by EDI; the signed quotient is in EAX and the remainder is in EAX

5-4 Basic Logic Instructions

•AND

–0010 00dw oorrrmmm disp	AND reg/mem, reg/mem
–1000 00sw oo100mmm disp imm	AND reg/mem/A?, imm

```
  x x x x  x x x x   Unknown number
· 0 0 0 0  1 1 1 1   Mask
-------------------
  0 0 0 0  x x x x   Result
```

•OR

–0000 10dw oorrrmmm disp	OR reg/mem, reg/mem
–1000 00sw oo001mmm disp imm	OR reg/mem/A?, imm
–0000 110w imm	OR A?, imm

```
  x x x x  x x x x   Unknown number
+ 0 0 0 0  1 1 1 1   Mask
-------------------
  x x x x  1 1 1 1   Result
```

•XOR

–0001 10dw oorrrmmm disp	XOR reg/mem, reg/mem
–1000 00sw oo110mmm disp imm	XOR reg/mem/A?, imm
–0010 101w imm	XOR A?, imm

```
  x x x x  x x x x   Unknown number
⊕ 0 0 0 0  1 1 1 1   Mask
-------------------
  x x x x  x̄ x̄ x̄ x̄   Result
```

EXAMPLE 5–26

```
0003  81 E3 0F0F    AND   BX,0F0FH   ;mask BX
```

EXAMPLE 5–27

```
0008  0D 3030       OR    AX,3030H   ;to ASCII
```

EXAMPLE 5–28

```
0000  81 C9 0600    OR    CX,0600H   ;set bits 9 and 10
0004  83 E1 FC      AND   CX,0FFFCH  ;clear bits 0 and 1
0007  81 F1 1000    XOR   CX,1000H   ;invert bit 12
```

NOT and NEG

- NOT: 1's complement
  - 1111 011w oo010mmm disp NOT reg/mem
- NEG: 2's complement
  - 1111 011w oo011mmm disp NEG reg/mem

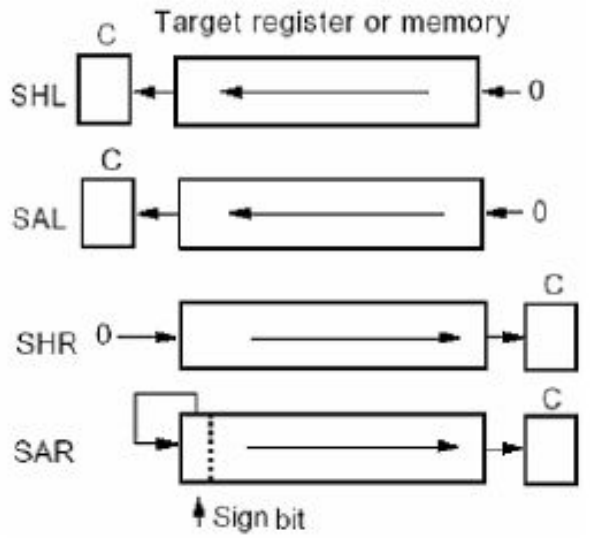
TABLE 5–19 NOT and NEG instructions.

Assembly Language	*Operation*
NOT CH	CH is one's complemented
NEG CH	CH is two's complemented
NEG AX	AX is two's complemented
NOT EBX	EBX is one's complemented
NEG ECX	ECX is two's complemented
NOT TEMP	The contents of the data segment memory location TEMP is one's complemented
NOT BYTE PTR[BX]	The byte contents of the data segment memory location addressed by BX is one's complemented

5-5 Shift and Rotate

- Shift and rotate: binary manipulation at the bit level
  - applications in low-level software used to control I/O devices
- Shift
  - Two directions: like multiplication and division
    - left shift: multiplication by powers of 2^{+n}
    - right shift: division by powers of 2^{-n}
  - logical vs. arithmetic shifts: Logical shift operations function with unsigned numbers, and arithmetic shifts function with signed numbers
    - The arithmetic right shift copies the sign-bit through the number
    - The logical right shift copies a 0 through the number
- Rotate
  - rotate information from one end to another or through the carry flag

Shift Instructions



1101 000w ooTTTmmm disp SAL reg/mem, 1
1101 001w ooTTTmmm disp SAL reg/mem, CL
1100 000w ooTTTmmm disp imm SAL reg/mem, imm

TTT:

100 SHL=SAL
101 SHR
111 SAR

EXAMPLE 5–31

```
0000 C1 E2 0E    SHL DX,14
                   or
0003 B1 0E       MOV CL,14
0005 D3 E2       SHL DX,CL
```

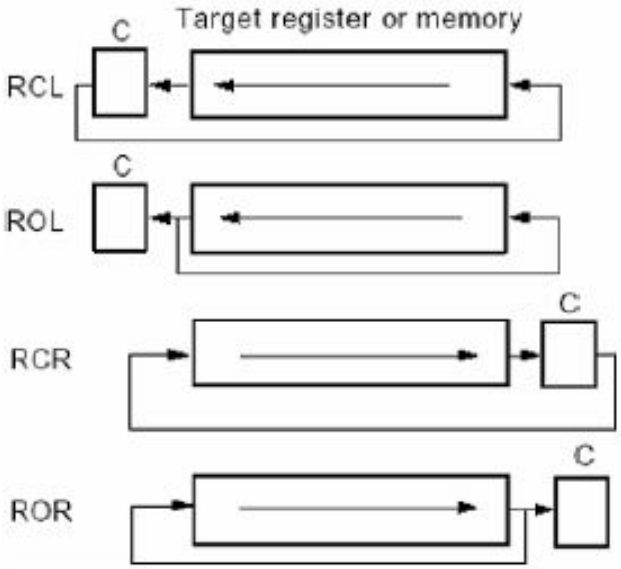
TABLE 5–20 Shift instructions.

Assembly Language	*Operation*
SHL AX,1	AX is logically shifted left 1 place
SHR BX,12	BX is logically shifted right 12 places
SHR ECX,10	ECX is logically shifted right 10 places
SAL DATA1,CL	The contents of the data segment memory location DATA1 is arithmetically shifted left the number of places specified by CL
SAR SI,2	SI is arithmetically shifted right 2 places
SAR EDX,14	EDX is arithmetically shifted right 14 places

EXAMPLE 5–32

```
                ;Multiply AX by 10 (1010)
                ;
0000  D1 E0       SHL   AX,1        ;AX times 2
0002  8B D8       MOV   BX,AX
0004  C1 E0 02    SHL   AX,2        ;AX times 8
0007  03 C3       ADD   AX,BX       ;10 times AX
                ;
                ;Multiply AX by 18 (10010)
                ;
0009  D1 E0       SHL   AX,1        ;AX times 2
000B  8B D8       MOV   BX,AX
000D  C1 E0 03    SHL   AX,3        ;AX times 16
0010  03 C3       ADD   AX,BX       ;18 times AX
                ;
                ;Multiply AX by 5 (101)
                ;
0012  8B D8       MOV   BX,AX
0014  D1 E0       SHL   AX,1        ;AX times 2
0016  D1 E0       SHL   AX,1        ;AX times 4
0018  03 C3       ADD   AX,BX       ;5 times AX
```

Rotate Instructions



1101 000w ooTTTmmm disp ROL reg/mem, 1
1101 001w ooTTTmmm disp ROL reg/mem, CL
1100 000w ooTTTmmm disp imm ROL reg/mem, imm

TTT:
- 000 ROL
- 001 ROR
- 010 RCL
- 011 RCR

EXAMPLE 5–33

```
0000  D1 E0          SHL   AX,1
0002  D1 D3          RCL   BX,1
0004  D1 D2          RCL   DX,1
```

TABLE 5–21 Rotate instructions.

Assembly Language	*Operation*
ROL SI,14	SI rotates left 14 places
RCL BL,6	BL rotates left through carry 6 places
ROL ECX,18	ECX rotates left 18 places
RCR AH,CL	AH rotates right through carry the number of places specified by CL
ROR WORD PTR[BP],2	The word contents of the stack segment memory location addressed by BP rotate right 2 places

Thanks