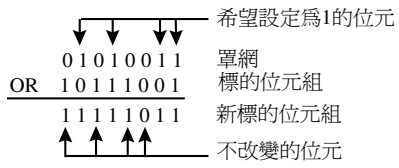
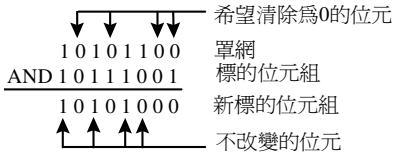
本章目標

- 了解80x86的邏輯運算指令與程式設計
- 了解80x86的位元運算指令與程式設計
- 了解80x86的移位與循環指令與程式設計
- 了解80x86的符號擴展指令與特殊指令的動作
- 了解80x86的字元串運算指令與程式設計
- 了解80x86的CPU控制與旗號位元指令的動作

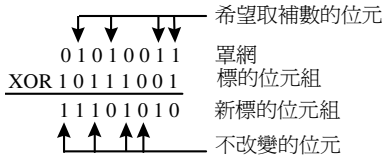
邏輯運算指令動作



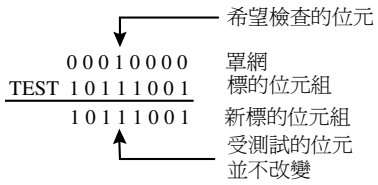
(a) OR運算



(b) AND運算



(c) XOR運算



(d) TEST運算

80x86邏輯運算指令

指令	動作	OF	SF	ZF	AF	PF	CF
AND reg1,reg2	reg1 ← reg2 ∧ reg1	0	*	*	U	*	0
AND reg,mem	reg ← reg ∧(mem)	0	*	*	U	*	0
AND mem,reg	(mem) ← reg ∧(mem)	0	*	*	U	*	0
AND reg,data	reg ← reg ∧ data	0	*	*	U	*	0
AND mem,data	(mem) ← (mem) ∧ data	0	*	*	U	*	0
AND ACC,data	ACC ← ACC ∧ data	0	*	*	U	*	0
OR reg1,reg2	reg1 ← reg2 ∨ reg1	0	*	*	U	*	0
OR reg,mem	reg ← reg ∨(mem)	0	*	*	U	*	0
OR mem,reg	(mem) ← reg ∨(mem)	0	*	*	U	*	0
OR reg,data	reg ← reg ∨ data	0	*	*	U	*	0
OR mem,data	(mem) ← (mem) ∨ data	0	*	*	U	*	0
OR ACC,data	ACC ← ACC ∨ data	0	*	*	U	*	0

80x86邏輯運算指令

XOR reg1,reg2	reg1 ← reg2 ⊕ reg1	0	*	*	U	*	0
XOR reg,mem	reg ← reg ⊕ (mem)	0	*	*	U	*	0
XOR mem,reg	(mem) ← reg ⊕ (mem)	0	*	*	U	*	0
XOR reg,data	reg ← reg ⊕ data	0	*	*	U	*	0
XOR mem,data	(mem) ← (mem) ⊕ data	0	*	*	U	*	0
XOR ACC,data	ACC ← ACC ⊕ data	0	*	*	U	*	0
TEST reg1,reg2	reg1 ∧ reg2	0	*	*	U	*	0
TEST reg,mem	reg ∧(mem)	0	*	*	U	*	0
TEST mem,reg	(mem) ∧ reg	0	*	*	U	*	0
TEST reg,data	reg ∧ data	0	*	*	U	*	0
TEST mem,data	(mem) ∧ data	0	*	*	U	*	0
TEST ACC,data	ACC ∧ data	0	*	*	U	*	0

$$F = \overline{AB} \oplus (C + D)$$

```
                    ;ex5.1-1.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 7B               INPUT_A   DB  01111011B ;input a
0001 9D               INPUT_B   DB  10011101B ;input b
0002 92               INPUT_C   DB  10010010B ;input c
0003 F1               INPUT_D   DB  11110001B ;input d
0004 00               RESULT_F  DB  ?         ;result f
                        ....
0005  A0 0000 R             MOV   AL,INPUT_A;get INPUT_A
0008  F6 D0                NOT   AL      ;complement it
000A  22 06 0001 R           AND   AL,INPUT_B;AND INPUT_B
000E  8A 0E 0002 R           MOV   CL,INPUT_C;get INPUT_C
0012  0A 0E 0003 R           OR    CL,INPUT_D;OR INPUT_D
0016  32 C1                XOR   AL,CL   ;XOR AL with CL
0018  A2 0004 R              MOV   RESULT_F,AL;save result
001B  C3                   RET
001C                LGSML   ENDP
001C                CODE    ENDS
                        END   LGSML
```


邏輯運算指令的使用例

```
                    ;ex5.1-2.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 05               BYTE1   DB   05H       ;bit number
0001 01               BYTE2   DB   01H       ;bit value
0002                DATA    ENDS
                        ....
0000  B8 ---- R             MOV   AX,DATA    ;load DS
0003  8E D8                 MOV   DS,AX
0005  F6 06 0001 R 20           TEST  BYTE2,20H  ;test the bit 5
000A  74 07                 JZ    CLRBIT     ;of byte2
000C  80 0E 0000 R 08     SETBIT:   OR    BYTE1,08H  ;set the bit 3
0011  EB 05                 JMP   SHORT RETURN;of byte1
0013  80 26 0000 R F7     CLRBIT:   AND   BYTE1,0F7H ;clear the bit 3
0018  C3                RETURN:   RET           ;of byte1
0019                SETBTV    ENDP
0019                CODE    ENDS
                        END   SETBTV
```

邏輯運算指令的使用例

```
                        ;ex5.1-3.asm
0000                    DATA    SEGMENT PUBLIC 'DATA'
0000 05                 BITNO   DB  05H     ;bit number
0001 01                 VALUE   DB  01H     ;bit value
0002 00                 MEMORY  DB  00H     ;memory location
0003 01 02 04 08        BMASK   DB  01H,02H,04H,08H
0007  10 20 40 80               DB  10H,20H,40H,80H
                        ....
0005  8A 1E 0000 R              MOV BL,BITNO  ;get bit number
0009  32 FF                 XOR BH,BH     ;zero BH
000B  8A 97 0003 R              MOV DL,BMASK[BX];get mask entry
000F  80 3E 0001 R 00           CMP VALUE,00H ;test value
0014  74 06                 JZ  CLRBIT
0016  08 16 0002 R      SETBIT: OR  MEMORY,DL ;set the MEMORY
001A  EB 06                 JMP SHORT RETURN ;bit
001C  F6 D2             CLRBIT: NOT DL      ;clear the MEMORY
001E  20 16 0002 R              AND MEMORY,DL ;bit
0022  C3                RETURN: RET
0023                    SETMBT  ENDP
0023                    CODE    ENDS
                            END SETMBT
```


計數一個位元組中"1"的個數

```
                        ;ex5.1-4.asm
0000                    DATA    SEGMENT PUBLIC 'DATA'
= 0008                  BCOUNT  EQU 08H     ;bit bumber
0000 47                 TDATA   DB  47H     ;test data
0001 00                 COUNT   DB  00H     ;result
0002 01 02 04 08        EMASK   DB  01H,02H,04H,08H ;mask
0006  10 20 40 80               DB  10H,20H,40H,80H
                        ....
0005  B9 0008                   MOV CX,BCOUNT ;put count in CX
0008  BE 0000                   MOV SI,00H    ;zero index
000B  32 E4                 XOR AH,AH     ;zero AH
000D  A0 0000 R         BEGIN:  MOV AL,TDATA  ;get test data
0010  22 84 0002 R              AND AL,EMASK[SI];test bit value
0014  74 02                 JZ  NEXT      ;if not zero
0016  FE C4                 INC AH        ;increase count
0018  46                NEXT:   INC SI        ;increase index
0019  49                    DEC CX        ;repeat until
001A  75 F1                 JNZ BEGIN     ;CX = 0
001C  88 26 0001 R              MOV COUNT,AH  ;store result
0020  C3                    RET
```

TEST指令的使用例

```
                    ;ex5.1-5.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
= 0008              BCOUNT  EQU   08H       ;bit number
0000 47             TDATA   DB    47H       ;test data
0001 00             COUNT   DB    00H       ;result
0002 01 02 04 08        EMASK   DB    01H,02H,04H,08H ;mask
0006  10 20 40 80               DB    10H,20H,40H,80H
                        ....
0005  B9 0008                   MOV   CX,BCOUNT  ;put count in CX
0008  BE 0000                   MOV   SI,00H     ;zero index
000B  32 E4                     XOR   AH,AH      ;zero AH
000D  A0 0000 R                 MOV   AL,TDATA   ;get test data
0010  84 84 0002 R          BEGIN:   TEST  AL,EMASK[SI];test bit value
0014  74 02                     JZ    NEXT       ;if not zero
0016  FE C4                     INC   AH         ;increase count
0018  46                 NEXT:  INC   SI         ;increase index
0019  E2 F5                     LOOP  BEGIN   ;repeat until CX=0
001B  88 26 0001 R              MOV   COUNT,AH   ;store result
001F  C3                        RET
```


80386 CPU位元運算指令

指令	動作	OF	SF	ZF	AF	PF	CF
BT r/m16,r16	CF← r16/(m16)中的第 r16 個位元值。	-	-	-	-	-	*
BT r/m32,r32	CF← r32/(m32)中的第 r32 個位元值。	-	-	-	-	-	*
BT r/m16,imm8	CF← r16/(m16)中的第 imm8 個位元值。	-	-	-	-	-	*
BT r/m32,imm8	CF← r32/(m32)中的第 imm8 個位元值。	-	-	-	-	-	*
BTC r/m16,r16	CF← r16/(m16)中的第 r16 個位元值; 並將該位元取補數。	-	-	-	-	-	*
BTC r/m32,r32	CF← r32/(m32)中的第 r32 個位元值; 並將該位元取補數。	-	-	-	-	-	*
BTC r/m16,imm8	CF← r16/(m16)中的第 imm8 個位元值; 並將該位元取補數。	-	-	-	-	-	*
BTC r/m32,imm8	CF← r32/(m32)中的第 imm8 個位元值; 並將該位元取補數。	-	-	-	-	-	*

80386 CPU位元運算指令

BTR r/m16,r16	CF← r16/(m16)中的第 r16 個位元值; 並清除該位元爲 0 。	-	-	-	-	-	*
BTR r/m32,r32	CF← r32/(m32)中的第 r32 個位元值; 並清除該位元爲 0 。	-	-	-	-	-	*
BTR r/m16,imm8	CF← r16/(m16)中的第 imm8 個位元值; 並清除該位元爲 0 。	-	-	-	-	-	*
BTR r/m32,imm8	CF← r32/(m32)中的第 imm8 個位元值; 並清除該位元爲 0 。	-	-	-	-	-	*
BTS r/m16,r16	CF← r16/(m16)中的第 r16 個位元值; 並設定該位元爲 1 。	-	-	-	-	-	*
BTS r/m32,r32	CF← r32/(m32)中的第 r32 個位元值; 並設定該位元爲 1 。	-	-	-	-	-	*
BTS r/m16,imm8	CF← r16/(m16)中的第 imm8 個位元值; 並設定該位元爲 1 。	-	-	-	-	-	*
BTS r/m32,imm8	CF← r32/(m32)中的第 imm8 個位元值; 並設定該位元爲 1 。	-	-	-	-	-	*

靜態位元運算指令使用例

```
                    ;ex5.2-1.asm
                    .386     ;386/486 and up processor only
0000                  DATA     SEGMENT PUBLIC 'DATA' USE16
0000 0005               BYTE1    DW   0005H   ;control word 1
0002 00FF               BYTE2    DW   00FFH   ;control word 2
0004                  DATA     ENDS
                        ....
0000                  SETBTV   PROC  NEAR
0000  B8 ---- R               MOV   AX,DATA ;load DS
0003  8E D8                   MOV   DS,AX
0005  0F BA 26 0002 R 05          BT    BYTE2,5 ;test bit 5
000B  73 08                   JNC   CLRBIT  ;of byte2
000D  0F BA 2E 0000 R 03          BTS   BYTE1,3 ;set bit 3 of
0013  EB 06                   JMP   RETURN  ;byte1
0015  0F BA 36 0000 R 03    CLRBIT:   BTR   BYTE1,3 ;clear bit 3
001B  C3                  RETURN:   RET         ;of byte1
001C                  SETBTV   ENDP
001C                  CODE     ENDS
                          END   SETBTV
```

靜態位元運算指令使用例

```
                        ;ex5.2-2.asm
                        .386      ;386/486 and up processor only
0000                      DATA      SEGMENT PUBLIC 'DATA' USE16
0000 0005                   BYTE1     DW    05H     ;control word 1
0002 0001                   BYTE2     DW    01H     ;control word 2
0004                      DATA      ENDS
                            ....
0000                      SETBTV    PROC  NEAR
0000  B8 ---- R                     MOV   AX,DATA ;load DS
0003  8E D8                         MOV   DS,AX
0005  0F BA 2E 0000 R 03                BTS   BYTE1,3;set byte1 bit 3
000B  0F BA 26 0002 R 05                BT    BYTE2,5;test bit 5
0011  72 06                         JC    RETURN  ;of byte2
0013  0F BA 36 0000 R 03                BTR   BYTE1,3 ;clear bit 3
0019  C3                  RETURN:   RET           ;of byte1
001A                      SETBTV    ENDP
001A                      CODE      ENDS
                              END   SETBTV
```


動態位元運算指令使用例

```
                        ;ex5.2-3.asm
                        .386      ;386/486 and up processor only
0000                      DATA      SEGMENT PUBLIC 'DATA' USE16
0000 0005                   BITNO     DW    0005H   ;bit number
0002 0001                   VALUE     DW    0001H   ;bit value
0004 0000                   MEMORY    DW    0000H   ;memory location
                            ....
0000                      SETMBT    PROC  NEAR
0000  B8 ---- R                     MOV   AX,DATA ;load DS
0003  8E D8                         MOV   DS,AX
0005  A1 0000 R                       MOV   AX,BITNO ;get bit number
0008  0F BA 26 0002 R 00                BT    VALUE,0  ;test bit 0 of value
000E  73 07                         JNC   CLRBIT
0010  0F AB 06 0004 R           SETBIT:   BTS   MEMORY,AX ;set the MEMORY
0015  EB 05                         JMP   SHORT RETURN ;bit
0017  0F B3 06 0004 R           CLRBIT:   BTR   MEMORY,AX ;clear the MEMORY
001C  C3                  RETURN:   RET           ;bit
001D                      SETMBT    ENDP
001D                      CODE      ENDS
                              END   SETMBT
```

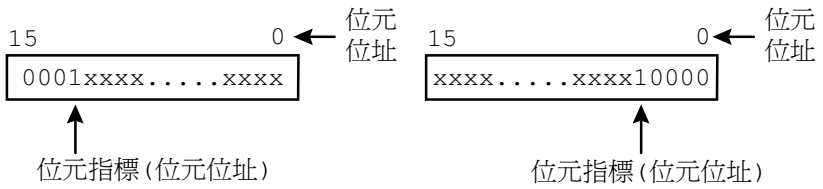
動態位元運算指令使用例

```
                        ;ex5.2-4.asm
                        .386     ;386/486 and up processor only
0000                      DATA     SEGMENT PUBLIC 'DATA' USE16
0000 0005                   BITNO    DW   0005H  ;bit number
0002 0001                   VALUE    DW   0001H  ;bit value
0004 0000                   MEMORY   DW   0000H  ;memory location
                              ....
0000                      SETMBT   PROC NEAR
0000  B8 ---- R                   MOV  AX,DATA ;load DS
0003  8E D8                       MOV  DS,AX
0005  A1 0000 R                   MOV  AX,BITNO  ;get bit number
0008  0F AB 06 0004 R               BTS  MEMORY,AX ;set the MEMORY
000D  0F BA 26 0002 R 00            BT   VALUE,0  ;test the value
0013  72 05                     JC   SHORT RETURN
0015  0F B3 06 0004 R               BTR  MEMORY,AX ;clear the MEMORY
001A  C3                  RETURN:  RET           ;bit
001B                      SETMBT   ENDP
001B                      CODE     ENDS
                              END  SETMBT
```


計數一個位元組中"1"位元的個數

```
                        ;ex5.2-5.asm
                        .386     ;386/486 and up processor only
0000                      DATA     SEGMENT PUBLIC 'DATA' USE16
0000 0047                   TDATA    DW   0047H  ;test data
0002 0000                   COUNT    DW   0000H  ;result
= 0008                    BCOUNT   EQU  08H     ;bit bumber
                              ....
0005  33 DB                       XOR  BX,BX     ;zero result
0007  8B CB                       MOV  CX,BX     ; and counter
0009  A1 0000 R                   MOV  AX,TDATA  ;get test data
000C  0F A3 C8            AGAIN:   BT   AX,CX     ;test bit value
000F  73 03                     JNC  NEXT      ;if not zero
0011  83 C3 01                    ADD  BX,1    ;increase result
0014  41                  NEXT:    INC  CX     ;repeat until
0015  83 F9 08                    CMP  CX,BCOUNT ;CX=BCOUNT
0018  75 F2                     JNE  AGAIN
001A  89 1E 0002 R                 MOV  COUNT,BX  ;store result
001E  C3                  RETURN:  RET
```

指令BSF與BSR的動作

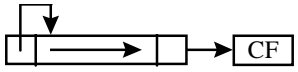


(a) 指令BSR　　(b) 指令BSF

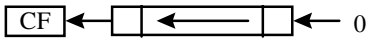
80386 CPU位元掃描運算指令

指令	動作	OF	SF	ZF	AF	PF	CF
BSF r16,r/m16	若 r16/(m16)不爲 0，則 ZF = 0 並儲存其最右邊 1 位元之位元位址於 r16 內；否則設定 ZF = 1 。	-	-	*	-	-	-
BSF r32,r/m32	若 r32/(m32)不爲 0，則 ZF = 0 並儲存其最右邊 1 位元之位元位址於 r32 內；否則設定 ZF = 1 。	-	-	*	-	-	-
BSR r16,r/m16	若 r16/(m16)不爲 0，則 ZF = 0 並儲存其最左邊 1 位元之位元位址於 r16 內；否則設定 ZF = 1 。	-	-	*	-	-	-
BSR r32,r/m32	若 r32/(m32)不爲 0，則 ZF = 0 並儲存其最左邊 1 位元之位元位址於 r32 內；否則設定 ZF = 1 。	-	-	*	-	-	-

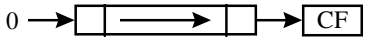
算術與邏輯移位



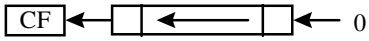
(a) 算術右移位



(b) 算術左移位

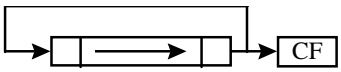


(c) 邏輯右移位

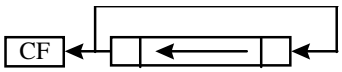


(d) 邏輯左移位

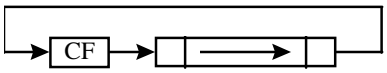
循環移位與連結進位循環移位



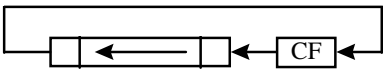
(a) 右循環移位



(b) 左循環移位

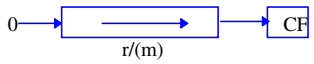
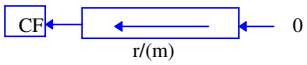
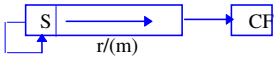
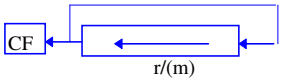


(c) 連結進位右循環移位

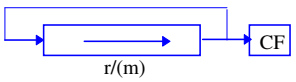
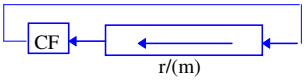
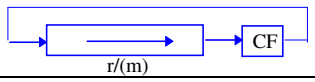


(d) 連結進位左循環移位

80x86移位與循環移位指令

指令	動作	OF	SF	ZF	AF	PF	CF
SHR r/m,1		*	*	*	U	*	*
SHR r/m,imm8		U	*	*	U	*	*
SHR r/m,CL		U	*	*	U	*	*
SHL/SAL r/m,1		*	*	*	U	*	*
SHL/SAL r/m,imm8		U	*	*	U	*	*
SHL/SAL r/m,CL		U	*	*	U	*	*
SAR r/m,1		*	*	*	U	*	*
SAR r/m,imm8		U	*	*	U	*	*
SAR r/m,CL		U	*	*	U	*	*
ROL r/m,1		*	-	-	-	-	*
ROL r/m,imm8		U	-	-	-	-	*
ROL r/m,CL		U	-	-	-	-	*

80x86移位與循環移位指令

ROR r/m,1		*	-	-	-	-	*
ROR r/m,imm8		U	-	-	-	-	*
ROR r/m,CL		U	-	-	-	-	*
RCL r/m,1		*	-	-	-	-	*
RCL r/m,imm8		U	-	-	-	-	*
RCL r/m,CL		U	-	-	-	-	*
RCR r/m,1		*	-	-	-	-	*
RCR r/m,imm8		U	-	-	-	-	*
RCR r/m,CL		U	-	-	-	-	*

ROR指令的使用例

```
                        ;ex5.3-1a.asm
0000                      DATA     SEGMENT PUBLIC 'DATA'
0000                      DATA     ENDS
                        ;swap two nibbles in register AL
                        ;using static rotation instruction
0000                      CODE     SEGMENT PUBLIC 'CODE'
                        ASSUME   CS:CODE,DS:DATA
0000                      SWAP4B   PROC  NEAR
0000  D0 C8                       ROR   AL,1    ;rotate register
0002  D0 C8                       ROR   AL,1    ;AL right 4 bits
0004  D0 C8                       ROR   AL,1
0006  D0 C8                       ROR   AL,1
0008  C3                        RET
0009                      SWAP4B   ENDP
0009                      CODE     ENDS
                              END   SWAP4B
```


ROR指令的使用例

```
                        ;ex5.3-1b.asm
0000                      DATA     SEGMENT PUBLIC 'DATA'
0000                      DATA     ENDS
                        ;swap two nibbles in register AL
                        ;using dynamic rotation instruction
0000                      CODE     SEGMENT PUBLIC 'CODE'
                        ASSUME   CS:CODE,DS:DATA
0000                      SWAP4B   PROC  NEAR
0000  B1 04                     MOV   CL,4    ;rotate register
0002  D2 C8                       ROR   AL,CL   ;AL right 4 bits
0004  C3                        RET
0005                      SWAP4B   ENDP
0005                      CODE     ENDS
                              END   SWAP4B
```

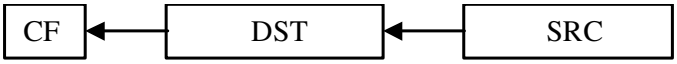
循環移位指令使用例

```
                        ;ex5.3-2.asm
0000                    DATA      SEGMENT PUBLIC 'DATA'
= 0008                  BCOUNT    EQU   08H       ;bit number
0000 47                 TDATA     DB    47H       ;test data
0001 00                 COUNT     DB    00H       ;result
0002                    DATA      ENDS
                            ....
0000                    B1CNTS    PROC  NEAR
0000  B8 ---- R                   MOV   AX,DATA    ;load DS
0003  8E D8                       MOV   DS,AX
0005  B9 0008                     MOV   CX,BCOUNT  ;put count in CX
0008  32 E4                       XOR   AH,AH      ;zero AH
000A  A0 0000 R                   MOV   AL,TDATA   ;get test data
000D  D0 C8             BEGIN:    ROR   AL,1       ;test bit value
000F  73 02                       JNC   NEXT       ;if not zero
0011  FE C4                       INC   AH         ;increase count
0013  E2 F8             NEXT:     LOOP  BEGIN    ;loop BCOUNT times
0015  88 26 0001 R                MOV   COUNT,AH   ;store result
0019  C3                        RET
001A                    B1CNTS    ENDP
```


動態循環移位指令使用例

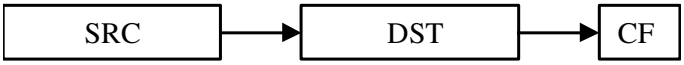
```
                        ;ex5.3-3.asm
0000                    DATA      SEGMENT PUBLIC 'DATA'
0000 05                 BITNO     DB    05H     ;bit number
0001 01                 VALUE     DB    01H     ;bit value
0002 00                 MEMORY    DB    00H     ;memory location
0003                    DATA      ENDS
                            ....
0000                    SETMBT    PROC  NEAR
0000  B8 ---- R                   MOV   AX,DATA    ;load DS
0003  8E D8                       MOV   DS,AX
0005  8A 0E 0000 R                MOV   CL,BITNO   ;get bit number
0009  80 26 0001 R 01             AND   VALUE,01H  ;extract bit 0
000E  A0 0002 R                   MOV   AL,MEMORY  ;get memory
0011  D2 C8             SETIT:    ROR   AL,CL   ;rotate the given
0013  24 FE                       AND   AL,0FEH ;bit to bit 0,
0015  0A 06 0001 R                OR    AL,VALUE;set its value,then
0019  D2 C0                       ROL   AL,CL   ;rotate it back
001B  A2 0002 R                   MOV   MEMORY,AL  ;save result
001E  C3                        RET
001F                    SETMBT    ENDP
```

SHLD與SHRD指令的動作



reg/mem

(a) SHLD 指令



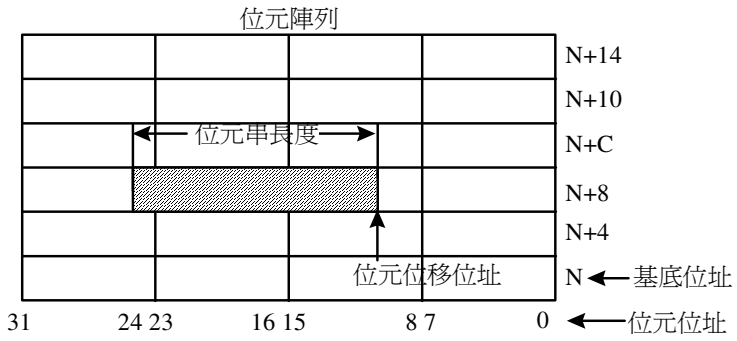
reg/mem

(b) SHRD 指令

80386 CPU雙精確制移位指令

指令	動作	OF	SF	ZF	AF	PF	CF
SHLD r/m16,r16,imm8	r16/(m16)儲存 SHL(r/m16:r16)的結果。	U	*	*	U	*	*
SHLD r/m32,r32,imm8	r32/(m32)儲存 SHL(r/m32:r32)的結果。	U	*	*	U	*	*
SHLD r/m16,r16,CL	r16/(m16)儲存 SHL(r/m16:r16)的結果。	U	*	*	U	*	*
SHLD r/m32,r32,CL	r32/(m32)儲存 SHL(r/m32:r32)的結果。	U	*	*	U	*	*
SHRD r/m16,r16,imm8	r16/(m16)儲存 SHR(r/m16:r16)的結果。	U	*	*	U	*	*
SHRD r/m32,r32,imm8	r32/(m32)儲存 SHR(r/m32:r32)的結果。	U	*	*	U	*	*
SHRD r/m16,r16,CL	r16/(m16)儲存 SHR(r/m16:r16)的結果。	U	*	*	U	*	*
SHRD r/m32,r32,CL	r32/(m32)儲存 SHR(r/m32:r32)的結果。	U	*	*	U	*	*

位元串運算的基本資料結構



符號擴展

例題 5.4-1 (符號擴展)

將下列兩數相加：-28(8 位元)與+96(16 位元)。

解： 8 位元的-28 先做符號擴展爲 16 位元後，與 16 位元的+96 相加，得到正確的結果 +68 ：

```
                        C = 1
  -28        1 1 1 1 1 1 1 1   1 1 1 0 0 1 0 0   (-28) --- 8位元
+ +96      + 0 0 0 0 0 0 0 0   0 1 1 0 0 0 0 0   (+96) --- 16位元
  +68        0 0 0 0 0 0 0 0   0 1 0 0 0 1 0 0   (+68) --- 16位元
```

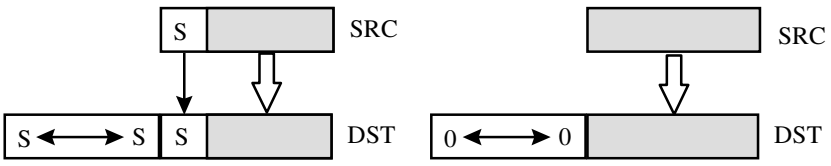
若-28 未做符號擴展爲 16 位元的值，即其高序位元組視爲 0 ，而直接與 16 位元的 +96 相加，則得到不正確的結果+324 ：

```
                        C = 1
  -28        0 0 0 0 0 0 0 0   1 1 1 0 0 1 0 0   (+228) --- 8位元
+ +96      + 0 0 0 0 0 0 0 0   0 1 1 0 0 0 0 0   (+96) --- 16位元
  +68        0 0 0 0 0 0 0 1   0 1 0 0 0 1 0 0   (+324) --- 16位元
```

80x86符號擴展指令

指令	動作	OF	SF	ZF	AF	PF	CF
CBW	AH ← AL(7)。	-	-	-	-	-	-
CWD	DX ← AX(15)。	-	-	-	-	-	-
CWDE	EAX(31:16) ← AX(15)。	-	-	-	-	-	-
CDQ	EDX ← EAX(31)。	-	-	-	-	-	-

MOVSX與MOVZX指令的動作



(a) MOVSX 指令　　(b) MOVZX 指令

符號擴展相關運算指令

指令	動作	OF	SF	ZF	AF	PF	CF
MOVSX r16,r8/m8	r16 ← r8/(m8)之符號擴展後。	-	-	-	-	-	-
MOVSX r32,r8/m8	r32 ← r8/(m8)之符號擴展後。	-	-	-	-	-	-
MOVSX r32,r16/m16	r32 ← r16/(m16)之符號擴展後。	-	-	-	-	-	-
MOVZX r16,r8/m8	r16 ← r8/(m8)之零值擴展後。	-	-	-	-	-	-
MOVZX r32,r8/m8	r32 ← r8/(m8)之零值擴展後。	-	-	-	-	-	-
MOVZX r32,r16/m16	r32 ← r16/(m16)之零值擴展後。	-	-	-	-	-	-
ADD r16/m16,imm8	r16/(m16)←r16/(m16) + SE(imm8)	*	*	*	*	*	*
ADD r32/m32,imm8	r32/(m32)←r32/(m32) + SE(imm8)	*	*	*	*	*	*
ADC r16/m16,imm8	r16/(m16)←r16/(m16) + SE(imm8) + C	*	*	*	*	*	*
ADC r32/m32,imm8	r32/(m32)←r32/(m32) + SE(imm8) + C	*	*	*	*	*	*
SUB r16/m16,imm8	r16/(m16)←r16/(m16) - SE(imm8)	*	*	*	*	*	*
SUB r32/m32,imm8	r32/(m32)←r32/(m32) - SE(imm8)	*	*	*	*	*	*

符號擴展相關運算指令

SBB r16/m16,imm8	r16/(m16)←r16/(m16) - SE(imm8) - C	*	*	*	*	*	*
SBB r32/m32,imm8	r32/(m32)←r32/(m32) - SE(imm8) - C	*	*	*	*	*	*
CMP r16/m16,imm8	r16/(m16) - SE(imm8)	*	*	*	*	*	*
CMP r32/m32,imm8	r32/(m32) - SE(imm8)	*	*	*	*	*	*
AND r16/m16,imm8	r16/(m16)←r16/(m16) ∧ SE(imm8)	0	*	*	U	*	0
AND r32/m32,imm8	r32/(m32)←r32/(m32) ∧ SE(imm8)	0	*	*	U	*	0
OR r16/m16,imm8	r16/(m16)←r16/(m16) ∨ SE(imm8)	0	*	*	U	*	0
OR r32/m32,imm8	r32/(m32)←r32/(m32) ∨ SE(imm8)	0	*	*	U	*	0
XOR r16/m16,imm8	r16/(m16)←r16/(m16) ⊕ SE(imm8)	0	*	*	U	*	0
XOR r32/m32,imm8	r32/(m32)←r32/(m32) ⊕ SE(imm8)	0	*	*	U	*	0
IMUL r16,imm8	r16←r16 × SE(imm8)	*	U	U	U	U	*
IMUL r32,imm8	r32←r32 × SE(imm8)	*	U	U	U	U	*
IMUL r16,r16/m16,imm8	r16←r16/(m16) × SE(imm8)	*	U	U	U	U	*
IMUL r32,r32/m32,imm8	r32←r32/(m32) × SE(imm8)	*	U	U	U	U	*
註： SE 表符號擴展(sign extension)。							

80x86字元串運算指令

指令	動作	OF	SF	ZF	AF	PF	CF
MOVS(MOVSB/ MOVSW/MOVSD)	ES:((E)DI) ← DS:((E)SI); SI←(E)SI ± 運算元長度; (E)DI←(E)DI ± 運算元長度	-	-	-	-	-	-
LODS(LODSB/ LODSW/LODSD)	ACC ← DS:((E)SI); SI←(E)SI ± 運算元長度	-	-	-	-	-	-
STOS(STOSB/ STOSW/STOSD)	ES:((E)DI) ← ACC; DI←(E)DI ± 運算元長度	-	-	-	-	-	-
CMPS(CMPSB/ CMPSW/CMPSD)	DS:((E)SI) - ES:((E)DI); SI←(E)SI ± 運算元長度; (E)DI←(E)DI ± 運算元長度	*	*	*	*	*	*

80x86字元串運算指令

SCAS(SCASB/ SCASW/SCASD)	ACC - ES:((E)DI); DI←(E)DI ± 運算元長度	*	*	*	*	*	*
REP(MOVS,LODS, STOS)	重複執行(MOVS,LODS,STOS)指令，直到(E)CX = 0 。	-	-	-	-	-	-
REPE/REPZ(CMPS SCAS)	重複執行(CMPS,SCAS)指令，直到(E)CX = 0 或 ZF = 0 。	-	-	-	-	-	-
REPNE/REPNZ (CMPS,SCAS)	重複執行(CMPS,SCAS)指令，直到(E)CX = 0 或 ZF = 1 。	-	-	-	-	-	-
註： ACC 爲 AL 、 AX 、或 EAX ，而運算元長度則爲 1 、 2 、或 4 。							

使用MOVSB指令的資料陣列搬移程式

```
                        ;ex5.5-1.asm
0000                    DATA      SEGMENT PUBLIC 'DATA'
= 0008                  LENTH     EQU   08H       ;bytes of array
0000 12 23              SRCA      DB    12H,23H    ;source array
0002  24 67                       DB    24H,67H
0004  76 98                       DB    76H,98H
0006  23 45                       DB    23H,45H
0008  0008 [ 00 ]         DSTA      DB    8 DUP(00);destination array
                          ....
0000                    BLKMOV    PROC  NEAR
0000  B8 ---- R                   MOV   AX,DATA  ;load DS and ES
0003  8E D8                       MOV   DS,AX    ;ES and DS use
0005  8E C0                       MOV   ES,AX    ;the same segment
0007  B9 0008                     MOV   CX,LENTH ;get length
000A  8D 36 0000 R                LEA   SI,SRCA ;set source pointer
000E  8D 3E 0008 R                LEA   DI,DSTA ;set dest. pointer
0012  FC                        CLD     ;set auto-increment mode
0013  A4                  MLOOP:  MOVSB         ;move data
0014  E2 FD                       LOOP  MLOOP   ;repeat until CX = 0
0016  C3                       RET
```


線性搜尋程式

```
                        ;ex5.5-2.asm
0000                    DATA      SEGMENT PUBLIC 'DATA'
= 0008                  LENTH     EQU   08H      ;array size
0000 0023 0002            TDATA     DW    23H,02H  ;data array
                          ....
0010 0040                 KEYD      DW    40H  ;data to be searched
                          ....
0000  B8 ---- R                   MOV   AX,DATA ;load DS and ES
0003  8E D8                       MOV   DS,AX
0005  8E C0                       MOV   ES,AX
0007  B9 0008                     MOV   CX,LENTH;get length
000A  8D 3E 0000 R                LEA   DI,TDATA;set pointer
000E  A1 0010 R                   MOV   AX,KEYD ;get key data
0011  FC                        CLD     ;set auto-increment mode
0012  AF                  MLOOP:  SCASW         ;search key data
0013  E0 FD                       LOOPNZ MLOOP
0015  74 05                CHECK:  JE    FOUND   ;found ?
0017  B8 FFFF                     MOV   AX,-1   ;no, move -1 to
001A  EB 05                       JMP   SHORT FOUND1 ;keyd
001C  B8 0008              FOUND:  MOV   AX,LENTH;yes,adjust index
001F  2B C1                       SUB   AX,CX   ;value
0021  A3 0010 R            FOUND1: MOV   KEYD,AX
0024  C3                        RET
```

REP前標

例題 5.5-3　(REP 前標)

試計算下列程式片段(a)與(b)需要的執行時間。

(a)　NEXT:　MOVSB
　　LOOP　NEXT

(b)　REP　MOVSB

解：以 8086 微處理器爲例：

(a)　每次均需 18(MOVSB) + 17(LOOP) = 35 個時脈

(b)　第一次需要 9 + 17 = 26 個時脈，而第二次以後則每次僅需 17 個時脈。

所以使用 REP 前標不但簡化機器碼也同時縮短執行時間。

REP前標與MOVSB指令

```
                        ;ex5.5-4.asm
0000                    DATA     SEGMENT PUBLIC 'DATA'
= 0008                  LENTH    EQU   08H       ;bytes of array
0000 12 23              SRCA     DB    12H,23H   ;source array
0002  24 67                      DB    24H,67H
0004  76 98                      DB    76H,98H
0006  23 45                      DB    23H,45H
0008  0008 [ 00 ]         DSTA     DB    8 DUP(00)  ;dest. array
0010                    DATA     ENDS
                           ....
0000                    BLKMOV   PROC  NEAR
0000  B8 ---- R                  MOV   AX,DATA ;load DS and ES
0003  8E D8                      MOV   DS,AX   ;ES and DS use
0005  8E C0                      MOV   ES,AX   ;the same segment
0007  B9 0008                    MOV   CX,LENTH;get length
000A  8D 36 0000 R                LEA   SI,SRCA ;set source pointer
000E  8D 3E 0008 R                LEA   DI,DSTA ;set dest. pointer
0012  FC                        CLD   ;set auto-increment mode
0013  F3/ A4              MLOOP:   REP   MOVSB   ;transfer data
0015  C3                        RET
```

80x86表格轉換指令

指令	動作	OF	SF	ZF	AF	PF	CF
XLATB	AL ← ((E)BX+零位元擴展之 AL)	-	-	-	-	-	-

80x86 CPU控制指令

指令	動作	OF	SF	ZF	AF	PF	CF
HLT	令 CPU 停止動作。	-	-	-	-	-	-
NOP	CPU 不做動作。	-	-	-	-	-	-
ESC data,mem	浮點運算處理器指令。	-	-	-	-	-	-
ESC data,reg	浮點運算處理器指令。	-	-	-	-	-	-
CPUID	讀取 CPU 相關的識別資訊	-	-	-	-	-	-

軟體延遲程式(8088)

```
                        ;ex5.6-1.asm
                        ;subroutine to delay a given clocks.
                        ;the delay time is calculated as follow:
                        ;delay time = 4+(3+3+17)(N-1)+(3+3+5)+8
                        ;           = 23N (clocks)
0000                    CODE      SEGMENT PUBLIC 'CODE'
= 0064                   N         EQU   100
                        ASSUME    CS:CODE
0000                     DELAY     PROC  NEAR
0000  B9 0064                     MOV   CX,N    ;  4
0003  90                 KTIME:    NOP           ;  3
0004  90                        NOP           ;  3
0005  E2 FC                       LOOP  KTIME   ;5(B=0);17(B<>0)
0007  C3                        RET           ;  8
0008                     DELAY     ENDP
0008                     CODE      ENDS
                             END   DELAY
```


80x86旗號位元運算指令

指令	動作	DF	TF	IF	OF	SF	ZF	AF	PF	CF
CLC	CF ← 0	-	-	-	-	-	-	-	-	0
CMC	CF ← $\overline{CF}$	-	-	-	-	-	-	-	-	$\overline{CF}$
STC	CF ← 1	-	-	-	-	-	-	-	-	1
CLD	DF ← 0	0	-	-	-	-	-	-	-	-
STD	DF ← 1	1	-	-	-	-	-	-	-	-
CLI	IF ← 0	-	-	0	-	-	-	-	-	-
STI	IF ← 1	-	-	1	-	-	-	-	-	-
LAHF	AH ← 旗號位元組(7:0)	-	-	-	-	-	-	-	-	-
SAHF	旗號位元組(7:0)← AH	-	-	-	-	*	*	*	*	*