本章目標

- 了解80x86的定址方式與指令使用
- 了解80x86的資料轉移指令與程式設計
- 了解80x86的算術運算指令與程式設計
- 了解80x86的分歧(跳躍)指令與程式設計
- 了解組合語言程式的迴路設計方法
- 了解組合語言程式的迴路指令與應用

80x86各種定址方式的格式

定址方式	格式	有效位址
立即資料定址	exp	
暫存器定址	reg	
直接定址	Var + exp/Var-exp	Var + exp/Var-exp
暫存器間接定址	[reg]	reg
暫存器相對定址	Var[reg + disp]或[reg + disp]	Var + reg + disp 或 reg + disp
基底指標定址	[reg1][reg2]	reg1 + reg2
基底指標相對定址	Var[reg1 + disp1][reg2 + disp2] 或[reg1 + disp1][reg2 + disp2]	Var + reg1 + disp1 + reg2 + disp2 或 reg1 + disp1 + reg2 + disp2
倍率指標定址	[reg*SF]	reg*SF
倍率指標相對定址	Var[reg*SF + disp]	Var + reg*SF + disp
倍率基底指標定址	[reg1][reg2*SF]	reg1 + reg2*SF
倍率基底指標相對定址	Var[reg1 + dsip1][reg2*SF + disp2]或 [reg1 + dsip1][reg2*SF + disp2]	Var + reg1 + disp1 + reg2*SF + disp2 或 reg1 + disp1 + reg2*SF + disp2

註: 1. disp, disp1, disp2 在 16 位元有效位址時可以爲 8 位元或 16 位元；在 32 位元有效位址時可以爲 8 位元或 32 位元。
2. SF 可以爲 1,2,4,或 8 。 3. reg, reg1,與 reg2 之選用，請參第 3.4 節。

定址方式與各種運算元長度

```
                        ;ex4.1-1.asm
                        ....
                              .386
                        ;32-bit operand size data segment
0000                      DATA32    SEGMENT PUBLIC 'DATA' USE16
0000 0047                   OPR1      DW    0047H
0002 00000047               OPR2      DD    0047H
0006                      DATA32    ENDS
                        ;32-bit addressing size code segment
0000                      CODE32    SEGMENT PUBLIC 'CODE' USE16
                        ASSUME    CS:CODE32,DS:DATA32
0000                      ADD_MODE  PROC  NEAR
                        ;16-bit addressing and 16-bit operand size.
0000  B8 0034                     MOV   AX,034H ;immediate
0003  8B D8                         MOV   BX,AX   ;register
0005  A1 0000 R                   MOV   AX,OPR1;direct
0008  8B 0F                      MOV   CX,[BX] ;register indirect
000A  8B 87 0000 R                  MOV   AX,OPR1[BX];register-relative
000E  8B 01                      MOV   AX,[BX][DI] ;based-indexed
                                 ;based-indexed-relative
0010  8B 81 0000 R                 MOV   AX,OPR1[BX][DI]
```


定址方式與各種運算元長度

```
                        ;32-bit addressing and 32-bit operand size.
0014  67& 66| 8B 1C CD              MOV   EBX,[ECX*8] ;scaled-indexed
      00000000
                              ;scaled-based-indexed
001D  67& 66| 8B 1C 78              MOV   EBX,[EAX][EDI*2];based-indexed
                        ;scaled-based-indexed-relative
0022  67& 66| 8B 9C B9              MOV   EBX,OPR2[ECX][EDI*4]
      00000002 R
                        ;Mixed use of  16/32-bit operand size
                        ;and addressing.
                        ;16-bit addressing and operand size
002B  8B 07                       MOV   AX,[BX]
                        ;32-bit addressing and 16-bit operand size
002D  67& 8B 03                     MOV   AX,[EBX]
                        ;16-bit addressing and 32-bit operand size
0030  66| 8B 07                    MOV   EAX,[BX]
0033                      ADD_MODE  ENDP
0033                      CODE32    ENDS
                              END
```

定址方式與各種運算元長度

```
                    ;ex4.1-2.asm
                    ;Some examples of the notations of
                    ;addressing modes
                    .386
                    ;32-bit operand size data segment
00000000                DATA32   SEGMENT PUBLIC 'DATA' USE32
00000000 00000047       OPR2     DD   0047H
0004                  DATA32   ENDS
                    ;32-bit addressing size code segment
00000000                CODE32   SEGMENT PUBLIC 'CODE' USE32
                    ASSUME   CS:CODE32,DS:DATA32
00000000                ADD_MODE  PROC  NEAR
                    ;32-bit addressing and 32-bit operand size.
00000000  B8 00000034            MOV   EAX,034H ;immediate
00000005  8B D8                 MOV   EBX,EAX   ;register
00000007  A1 00000000 R          MOV   EAX,OPR2;direct
0000000C  8B 1B                 MOV   EBX,[EBX] ;register indirect
0000000E  8B 9B 00000000 R       MOV   EBX,OPR2[EBX];register-relative
00000014  8B 1C 07              MOV   EBX,[EAX][EDI] ;based-indexed
```


定址方式與各種運算元長度

```
                    ;based-indexed-relative
00000017  8B 9C 0F              MOV   EBX,OPR2[ECX][EDI]
        00000000 R
0000001E  8B 1C CD              MOV   EBX,[ECX*8] ;scaled-indexed
        00000000
                        ;scaled-based-indexed
00000025  8B 1C 78              MOV   EBX,[EAX][EDI*2];based-indexed
                        ;scaled-based-indexed-relative
00000028  8B 9C B9              MOV   EBX,OPR2[ECX][EDI*4]
        00000000 R
                    ;Mixed use of  16/32-bit operand size
                    ;and addressing.
                    ;16-bit addressing and operand size
0000002F  67& 66| 8B 07          MOV   AX,[BX]
                    ;32-bit addressing and 16-bit operand size
00000033  66| 8B 03              MOV   AX,[EBX]
                    ;16-bit addressing and 32-bit operand size
00000036  67& 8B 07              MOV   EAX,[BX]
00000039              ADD_MODE  ENDP
0039                CODE32   ENDS
                        END
```

80x86資料轉移指令

指令	動作	OF	SF	ZF	AF	PF	CF
MOV mem,ACC	(mem)← ACC	-	-	-	-	-	-
MOV ACC,mem	ACC←(mem)	-	-	-	-	-	-
MOV reg1,reg2	reg1←reg2	-	-	-	-	-	-
MOV mem,reg	(mem)← reg	-	-	-	-	-	-
MOV reg,mem	reg←(mem)	-	-	-	-	-	-
MOV reg,data	reg←data	-	-	-	-	-	-
MOV mem,data	(mem)← data	-	-	-	-	-	-
MOV sreg,reg16	sreg← reg16	-	-	-	-	-	-
MOV sreg,m16	sreg← (m16)	-	-	-	-	-	-
MOV reg16,sreg	reg16← sreg	-	-	-	-	-	-
MOV m16,sreg	(m16)← sreg	-	-	-	-	-	-

80x86資料轉移指令

XCHG ACC,reg	ACC ↔ reg	-	-	-	-	-	-
XCHG mem,reg	(mem) ↔ reg	-	-	-	-	-	-
XCHG reg1,reg2	reg1 ↔ reg2	-	-	-	-	-	-
BSWAP r32	TEMP ← r32 r32(7:0) ↔ TEMP(31:24) r32(15:8) ↔ TEMP(23:16)	-	-	-	-	-	-
LEA r16/32,m16/32	r16/32 ← (m16/32)	-	-	-	-	-	-
Lsreg r16,m16:16	r16 ← (m16:16) sreg ← (m16:16 + 2)	-	-	-	-	-	-
Lsreg r32,m16:32	r32 ← (m16:32) sreg ← (m16:32 + 4)	-	-	-	-	-	-

MOV 指令與直接定址

```
                    ;ex4.2-2.asm
0000                  DATA     SEGMENT PUBLIC 'DATA'
0000 0047               OPR1     DW    0047H
0002 0023               OPR2     DW    0023H
0004                  DATA     ENDS
                    ;Exchange two words in memory
                    ;using absolute addressing mode
0000                  CODE     SEGMENT PUBLIC 'CODE'
                    ASSUME   CS:CODE,DS:DATA
0000                  SWAP     PROC  NEAR
0000  B8 ---- R                MOV   AX,DATA ;load DS
0003  8E D8                    MOV   DS,AX
0005  A1 0000 R                 MOV   AX,OPR1 ;get opr1
0008  8B 1E 0002 R               MOV   BX,OPR2 ;get opr2
000C  A3 0002 R                 MOV   OPR2,AX ;save opr1
000F  89 1E 0000 R               MOV   OPR1,BX ;save opr2
0013  C3                      RET
0014                  SWAP     ENDP
0014                  CODE     ENDS
                          END   SWAP
```


MOV指令與暫存器間接定址

```
                    ;ex4.2-3.asm
0000                  DATA     SEGMENT PUBLIC 'DATA'
0000 0047               OPR1     DW    0047H
0002 0023               OPR2     DW    0023H
0004                  DATA     ENDS
                          ....
                          ....
0000  B8 ---- R                MOV   AX,DATA ;load DS
0003  8E D8                    MOV   DS,AX
0005  8D 36 0000 R               LEA   SI,OPR1 ;set up pointers
0009  8D 3E 0002 R               LEA   DI,OPR2
000D  8B 04                    MOV   AX,[SI] ;get opr1
000F  8B 1D                    MOV   BX,[DI] ;get opr2
0011  89 05                   MOV   [DI],AX ;save opr1
0013  89 1C                    MOV   [SI],BX ;save opr2
0015  C3                      RET
0016                  SWAP     ENDP
0016                  CODE     ENDS
                          END   SWAP
```

MOV指令與暫存器相對定址

```
                    ;ex4.2-4.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 0047             OPR1    DW   0047H
0002 0023             OPR2    DW   0023H
0004                DATA    ENDS
                        ....
                        ....
0000                SWAP    PROC  NEAR
0000  B8 ---- R             MOV   AX,DATA ;load DS
0003  8E D8                 MOV   DS,AX
0005  BE 0000                MOV   SI,00H  ;zero SI
0008  8B 84 0000 R             MOV   AX,OPR1[SI] ;get opr1
000C  8B 9C 0002 R              MOV   BX,OPR2[SI] ;get opr2
0010  89 84 0002 R              MOV   OPR2[SI],AX ;save opr1
0014  89 9C 0000 R              MOV   OPR1[SI],BX ;save opr2
0018  C3                      RET
0019                SWAP    ENDP
0019                CODE    ENDS
                        END   SWAP
```


MOV指令與基底指標定址

```
                    ;ex4.2-5.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 0047             OPR1    DW   0047H
0002 0023             OPR2    DW   0023H
0004                DATA    ENDS
                        ....
                        ....
0000  B8 ---- R             MOV   AX,DATA ;load DS
0003  8E D8                 MOV   DS,AX
0005  8D 36 0000 R            LEA   SI,OPR1 ;set up pointers
0009  8D 3E 0002 R            LEA   DI,OPR2
000D  BB 0000                MOV   BX,00H  ;zero BX
0010  8B 00                  MOV   AX,[BX][SI] ;get opr1
0012  8B 09                  MOV   CX,[BX][DI] ;get opr2
0014  89 01                  MOV   [BX][DI],AX ;save opr1
0016  89 08                  MOV   [BX][SI],CX ;save opr2
0018  C3                    RET
0019                SWAP    ENDP
0019                CODE    ENDS
                        END   SWAP
```

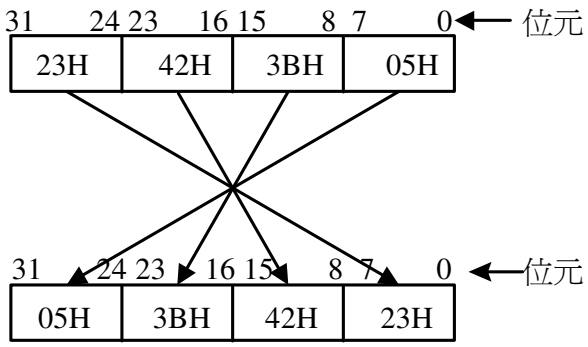
MOV指令與基底指標相對定址

```
                        ;ex4.2-6.asm
0000                    DATA    SEGMENT PUBLIC 'DATA'
0000 0047                 OPR1    DW   0047H
0002 0023                 OPR2    DW   0023H
0004                    DATA    ENDS
                            ....
                            ....
0000                    SWAP    PROC  NEAR
0000  B8 ---- R                 MOV   AX,DATA ;load DS
0003  8E D8                     MOV   DS,AX
0005  BE 0000                   MOV   SI,00H  ;zero SI
0008  8B DE                     MOV   BX,SI   ;zero BX
000A  8B 80 0000 R                MOV   AX,OPR1[BX][SI] ;get opr1
000E  8B 88 0002 R                MOV   CX,OPR2[BX][SI] ;get opr2
0012  89 80 0002 R                MOV   OPR2[BX][SI],AX ;save opr1
0016  89 88 0000 R                MOV   OPR1[BX][SI],CX ;save opr2
001A  C3                        RET
001B                    SWAP    ENDP
001B                    CODE    ENDS
                            END   SWAP
```


節區超越前標的使用

```
                        ;ex4.2-7.asm
0000                    DATA    SEGMENT PUBLIC 'DATA'
0000 0047                 OPR1    DW   0047H
0002 0023                 OPR2    DW   0023H
0004                    DATA    ENDS
                            ....
                            ....
0000                    SWAP    PROC  NEAR
0000  B8 ---- R                 MOV   AX,DATA ;load DS
0003  8E D8                     MOV   DS,AX
0005  BD 0000                   MOV   BP,00H  ;zero BP
0008  3E: 8B 86 0000 R            MOV   AX,DS:OPR1[BP] ;get opr1
000D  3E: 8B 9E 0002 R            MOV   BX,DS:OPR2[BP] ;get opr2
0012  3E: 89 86 0002 R            MOV   DS:OPR2[BP],AX ;save opr1
0017  3E: 89 9E 0000 R            MOV   DS:OPR1[BP],BX ;save opr2
001C  C3                        RET
001D                    SWAP    ENDP
001D                    CODE    ENDS
                            END   SWAP
```

指令BSWAP的動作



加法運算與旗號位元的關係

運算	SF ZF OF CF	帶號數	未帶號數
01000000 + 00101110 01101110	0 0 0 0	+64 + +46 + 110	64 + 46 110
01000110 + 01010000 10010110	1 0 1 0	+70 + +80 - 106(溢位)	70 + 80 150
01001110 + 10110010 100000000	0 1 0 1	+78 + -78 + 0	78 + 178 256(C=1)
10110010 + 10100000 101010010	0 0 1 1	-78 + -96 + 82(溢位)	178 + 160 82(C=1)

80x86二進制加法與減法指令

指令	動作	OF	SF	ZF	AF	PF	CF
ADD reg1,reg2	reg1 ← reg1 + reg2	*	*	*	*	*	*
ADD reg,mem	reg ← reg + (mem)	*	*	*	*	*	*
ADD mem,reg	(mem) ← reg + (mem)	*	*	*	*	*	*
ADD reg,data	reg ← reg + data	*	*	*	*	*	*
ADD mem,data	(mem) ← (mem) + data	*	*	*	*	*	*
ADD ACC,data	ACC ← ACC + data	*	*	*	*	*	*
ADC reg1,reg2	reg1 ← reg1 + reg2 + C	*	*	*	*	*	*
ADC reg,mem	reg ← reg + (mem) + C	*	*	*	*	*	*
ADC mem,reg	(mem) ← reg + (mem) + C	*	*	*	*	*	*
ADC reg,data	reg ← reg + data + C	*	*	*	*	*	*
ADC mem,data	(mem) ← (mem) + data + C	*	*	*	*	*	*
ADC ACC,data	ACC ← ACC + data + C	*	*	*	*	*	*

80x86二進制加法與減法指令

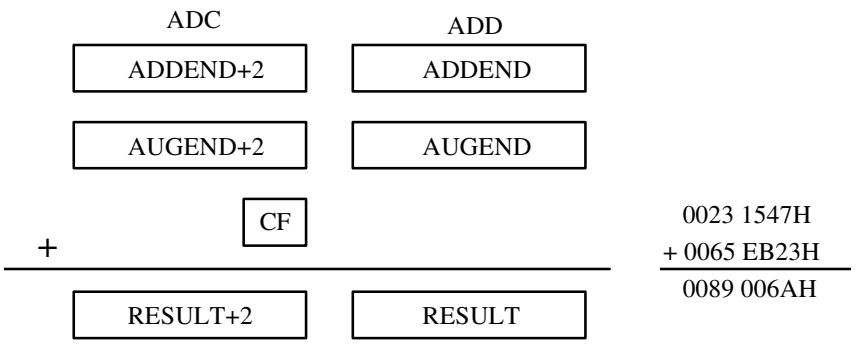
SUB reg1,reg2	reg1 ← reg1 - reg2	*	*	*	*	*	*
SUB reg,mem	reg ← reg - (mem)	*	*	*	*	*	*
SUB mem,reg	(mem) ← (mem) - reg	*	*	*	*	*	*
SUB reg,data	reg ← reg - data	*	*	*	*	*	*
SUB mem,data	(mem) ← (mem) - data	*	*	*	*	*	*
SUB ACC,data	ACC ← ACC - data	*	*	*	*	*	*
SBB reg1,reg2	reg1 ← reg1 - reg2 - C	*	*	*	*	*	*
SBB reg,mem	reg ← reg - (mem) - C	*	*	*	*	*	*
SBB mem,reg	(mem) ← (mem) - reg - C	*	*	*	*	*	*
SBB reg,data	reg ← reg - data - C	*	*	*	*	*	*
SBB mem,data	(mem) ← (mem) - data - C	*	*	*	*	*	*
SBB ACC,data	ACC ← ACC - data - C	*	*	*	*	*	*
CMP reg1,reg2	reg1 - reg2	*	*	*	*	*	*
CMP reg,mem	reg - (mem)	*	*	*	*	*	*
CMP mem,reg	(mem) - reg	*	*	*	*	*	*
CMP reg,data	reg - data	*	*	*	*	*	*
CMP mem,data	(mem) - data	*	*	*	*	*	*
CMP ACC,data	ACC - data	*	*	*	*	*	*

註：在旗號欄中的“*“表示該旗號的狀態會受指令的影響。

單精確制加法

```
                    ;ex4.3-1.asm
0000                  DATA      SEGMENT PUBLIC 'DATA'
0000 0047               ADDEND    DW    0047H     ;addend
0002 0023               AUGEND    DW    0023H     ;augend
0004 0000               RESULT    DW    0000H     ;result
0006                  DATA      ENDS
                    ;single-precision addition
0000                  CODE      SEGMENT PUBLIC 'CODE'
                    ASSUME    CS:CODE,DS:DATA
0000                  ADD16     PROC  NEAR
0000  B8 ---- R                 MOV   AX,DATA    ;load DS
0003  8E D8                     MOV   DS,AX
0005  A1 0000 R                   MOV   AX,ADDEND  ;get addend
0008  03 06 0002 R                 ADD   AX,AUGEND  ;add them
000C  A3 0004 R                   MOV   RESULT,AX  ;store result
000F  C3                        RET
0010                  ADD16     ENDP
0010                  CODE      ENDS
                          END   ADD16
```


雙精確制加法動作



雙精確制(32位元)加法

```
                    ;ex4.3-2.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 0047             ADDEND  DW  0047H     ;addend
0002  0023                    DW  0023H
0004 0023             AUGEND  DW  0023H     ;augend
0006  0065                    DW  0065H
0008 0000             RESULT  DW  0000H     ;result
000A  0000                    DW  0000H
                        ....
0000  B8 ---- R               MOV AX,DATA   ;load DS
0003  8E D8                   MOV DS,AX
0005  A1 0000 R               MOV AX,ADDEND ;get addend(lo)
0008  03 06 0004 R            ADD AX,AUGEND ;add them
000C  A3 0008 R               MOV RESULT,AX ;store result(lo)
000F  A1 0002 R               MOV AX,ADDEND+2;get addend(hi)
0012  13 06 0006 R            ADC AX,AUGEND+2;add them
0016  A3 000A R               MOV RESULT+2,AX;save result(hi)
0019  C3                    RET
001A                ADD32   ENDP
001A                CODE    ENDS
                        END ADD32
```

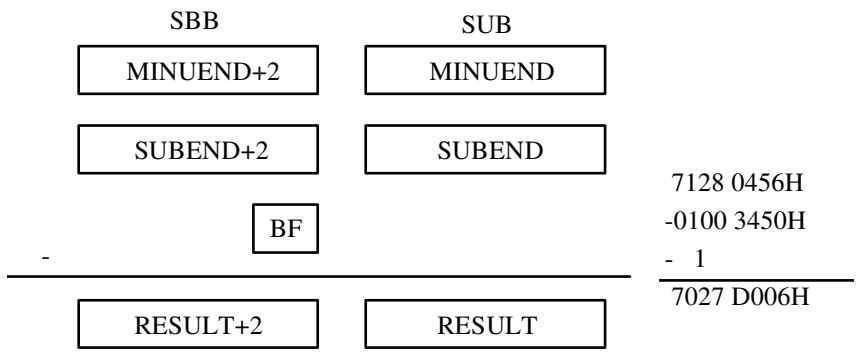

減法運算與旗號位元的關係

運算	SF ZF OF CF	帶號數	未帶號數
01000000 - 00101110 100010010	0 0 0 0	+64(大) - +46(小) + 18	64(大) - 46(小) 18
01000110 - 01010000 011110110	1 0 0 1	+70(小) - +80(大) - 10	70(小) - 80(大) 246(B=1)
01001110 - 10110010 010011100	1 0 1 1	+78 - -78 - 100(溢位)	78 - 178 156
10110010 - 10100000 100010010	0 0 0 0	-78(大) - -96(小) + 18	178(大) - 160(小) 18
10110010 - 11000100 011101110	1 0 0 1	-78(小) - -60(大) - 18	178(小) - 196(大) 238(B=1)
01100000 - 10110010 010101110	1 0 1 1	+96(大) - -78(小) - 82(溢位)	96(小) - 178(大) 238(B=1)
10110010 - 01111111 100110011	0 0 1 0	-78(小) - +127(大) + 51(溢位)	178(大) - 127(小) 51

單精確制減法

```
                    ;ex4.3-3.asm
0000                  DATA     SEGMENT PUBLIC 'DATA'
0000 0023               MINUEND  DW   0023H    ;minuend
0002 0047               SUBEND   DW   0047H    ;subend
0004 0000               RESULT   DW   0000H    ;result
0006                  DATA     ENDS
                    ;single-precision subtraction
0000                  CODE     SEGMENT PUBLIC 'CODE'
                    ASSUME   CS:CODE,DS:DATA
0000                  SUB16    PROC  NEAR
0000  B8 ---- R               MOV   AX,DATA   ;load DS
0003  8E D8                   MOV   DS,AX
0005  A1 0000 R                MOV   AX,MINUEND ;get minuend
0008  2B 06 0002 R              SUB   AX,SUBEND  ;subtract them
000C  A3 0004 R                 MOV   RESULT,AX  ;store result
000F  C3                      RET
0010                  SUB16    ENDP
0010                  CODE     ENDS
                          END   SUB16
```


雙精確制減法動作



7128 0456H
-0100 3450H
- 1
7027 D006H

雙精確制減法

```
                    ;ex4.3-4.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 0023             MINUEND   DW    0023H     ;minuend
0002  0065                      DW    0065H
0004 0047             SUBEND    DW    0047H     ;subtrahend
0006  0023                      DW    0023H
0008 0000             RESULT    DW    0000H     ;result
000A  0000                      DW    0000H
                        ....
0000  B8 ---- R                 MOV   AX,DATA   ;load DS
0003  8E D8                     MOV   DS,AX
0005  A1 0000 R                 MOV   AX,MINUEND;get minuend(low)
0008  2B 06 0004 R               SUB   AX,SUBEND ;subtract them
000C  A3 0008 R                 MOV   RESULT,AX ;store result(low)
000F  A1 0002 R                 MOV   AX,MINUEND+2;get minuend(high)
0012  1B 06 0006 R               SBB   AX,SUBEND+2 ;subtract them
0016  A3 000A R                 MOV   RESULT+2,AX;save result(high)
0019  C3                      RET
001A                SUB32     ENDP
001A                CODE      ENDS
                        END   SUB32
```


80x86單運算元指令

指令	動作	OF	SF	ZF	AF	PF	CF
INC reg	reg ← reg + 1	*	*	*	*	*	-
INC mem	(mem) ← (mem) + 1	*	*	*	*	*	-
DEC reg	reg ← reg-1	*	*	*	*	*	-
DEC mem	(mem) ← (mem) - 1	*	*	*	*	*	-
NEG reg	reg ← 0 - reg	*	*	*	*	*	*
NEG mem	(mem) ← 0 - (mem)	*	*	*	*	*	*
NOT reg	reg ← $\overline{\text{reg}}$	-	-	-	-	-	-
NOT mem	(mem) ← $\overline{\text{(mem)}}$	-	-	-	-	-	-

80x86乘法與除法運算指令

指令	動作	OF	SF	ZF	AF	PF	CF
MUL r8/m8	AX←AL×r8/(m8)	*	U	U	U	U	*
MUL r16/m16	DX:AX←AX×r16/(m16)	*	U	U	U	U	*
MUL r32/m32	EDX:EAX←EAX×r32/(m32)	*	U	U	U	U	*
IMUL r8/m8	AX←AL×r8/(m8)	*	U	U	U	U	*
IMUL r16/m16	DX:AX←AX×r16/(m16)	*	U	U	U	U	*
IMUL r32/m32	EDX:EAX←EAX×r32/(m32)	*	U	U	U	U	*
IMUL r16,r16/m16	r16←r16×r16/(m16)	*	U	U	U	U	*
IMUL r32,r32/m32	r32←r32×r32/(m32)	*	U	U	U	U	*
IMUL r16,r16/m16,imm16	r16←r16/(m16)×imm16	*	U	U	U	U	*
IMUL r32,r32/m32,imm32	r32←r32/(m32)×imm32	*	U	U	U	U	*
IMUL r16,imm16	r16←r16×imm16	*	U	U	U	U	*
IMUL r32,imm32	r32←r32×imm32	*	U	U	U	U	*
DIV r8/m8	AH:AL←AX÷r8/(m8)	U	U	U	U	U	U
DIV r16/m16	DX:AX←DX:AX÷r16/(m16)	U	U	U	U	U	U
DIV r32/m32	EDX:EAX←EDX:EAX÷r32/(m32)	U	U	U	U	U	U
IDIV r8/m8	AH:AL←AX÷r8/(m8)	U	U	U	U	U	U
IDIV r16/m16	DX:AX←DX:AX÷r16/(m16)	U	U	U	U	U	U
IDIV r32/m32	EDX:EAX←EDX:EAX÷r32/(m32)	U	U	U	U	U	U

註: 在旗號欄中的"U"表示在指令執行後該旗號爲不確定狀態。

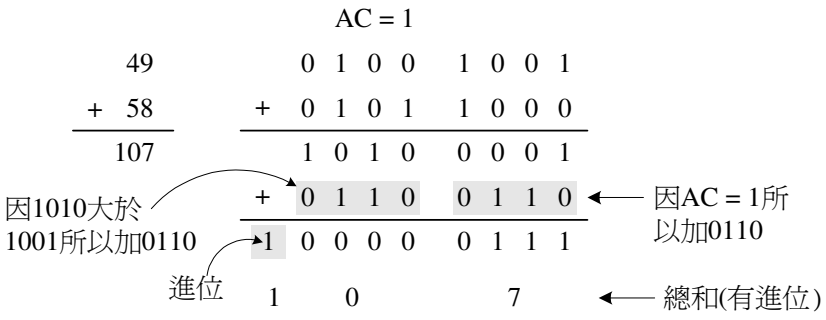
16位元乘法

```
                    ;ex4.3-5.asm
0000                  DATA      SEGMENT PUBLIC 'DATA'
0000 0047               MULTER    DW    0047H     ;multer
0002 0023               MULTEND   DW    0023H     ;multend
0004 0000               RESULT    DW    0000H     ;result(low)
0006 0000                         DW    0000H     ;result(high)
0008                  DATA      ENDS
                    ;single-precision multiplication
0000                  CODE      SEGMENT PUBLIC 'CODE'
                    ASSUME    CS:CODE,DS:DATA
0000                  MUL16     PROC  NEAR
0000 B8 ---- R                  MOV   AX,DATA    ;load DS
0003 8E D8                      MOV   DS,AX
0005 A1 0000 R                    MOV   AX,MULTER  ;get multer
0008 F7 26 0002 R                 MUL   MULTEND    ;multiply them
000C A3 0004 R                    MOV   RESULT,AX  ;save result(lo)
000F 89 16 0006 R                 MOV   RESULT+2,DX;save result(hi)
0013 C3                         RET
0014                  MUL16     ENDP
0014                  CODE      ENDS
                          END   MUL16
```

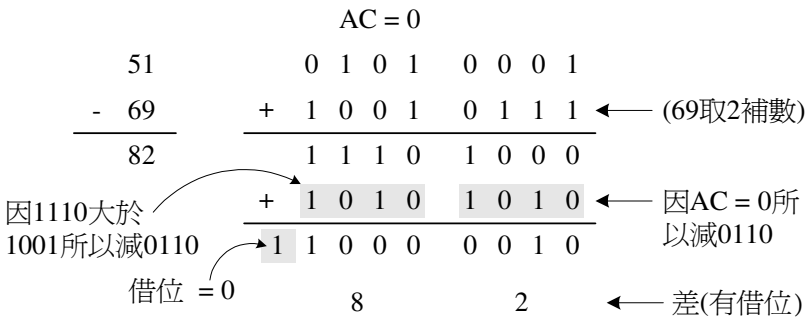
16位元除法

```
                    ;ex4.3-6.asm
0000                DATA      SEGMENT PUBLIC 'DATA'
0000 0023             DIVIDEND  DW    0023H  ;dividend
0002  0010                    DW    0010H
0004 0047             DIVISOR   DW    0047H  ;divisor
0006 0000             QUOTIENT  DW    0000H  ;result---quotient
0008 0000             REMAINDER DW    0000H  ;result---remainder
000A                DATA      ENDS
                        ....
0000  B8 ---- R               MOV   AX,DATA    ;load DS
0003  8E D8                   MOV   DS,AX
0005  8B 16 0002 R              MOV   DX,DIVIDEND+2 ;get dividend
0009  A1 0000 R                MOV   AX,DIVIDEND   ;
000C  F7 36 0004 R               DIV   DIVISOR       ;divide
0010  A3 0006 R                 MOV   QUOTIENT,AX ;save quotient
0013  89 16 0008 R               MOV   REMAINDER,DX;save remainder
0017  C3                      RET
0018                DIV16     ENDP
0018                CODE      ENDS
                        END   DIV16
```


併裝BCD加法的調整程序



併裝BCD減法的調整程序



80x86併裝BCD調整指令

指令	動作	OF	SF	ZF	AF	PF	CF
DAA	在加算之後調整 AL 內容爲十進制。	U	*	*	*	*	*
DAS	在減算之後調整 AL 內容爲十進制。	U	*	*	*	*	*

單精確制併裝BCD加法

```
                    ;ex4.3-9.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 47               ADDEND  DB  47H     ;addend
0001 23               AUGEND  DB  23H     ;augend
0002 00 00            RESULT  DB  00H,00H ;result
0004                DATA    ENDS
                        ....
0000  B8 ---- R             MOV AX,DATA  ;load DS
0003  8E D8                 MOV DS,AX
0005  A0 0000 R             MOV AL,ADDEND ;get addend
0008  B4 00                 MOV AH,00H   ;clear AH
000A  02 06 0001 R          ADD AL,AUGEND ;add them
000E  27                  DAA           ;decimal adjust
000F  A2 0002 R             MOV RESULT,AL ;save result(lo)
0012  80 D4 00              ADC AH,00H   ;get carry
0015  88 26 0003 R          MOV RESULT+1,AH;save result(hi)
0019  C3                  RET
001A                BCDADD  ENDP
001A                CODE    ENDS
                        END BCDADD
```


單精確制併裝BCD減法

```
                    ;ex4.3-10.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 47               MINUEND DB  47H     ;minuend
0001 23               SUBEND  DB  23H     ;subend
0002 00               RESULT  DB  00H     ;result
0003                DATA    ENDS
                        ....
0000  B8 ---- R             MOV AX,DATA  ;load DS
0003  8E D8                 MOV DS,AX
0005  A0 0000 R             MOV AL,MINUEND ;get minuend
0008  2A 06 0001 R          SUB AL,SUBEND ;subtract them
000C  2F                  DAS           ;decimal adjust
000D  A2 0002 R             MOV RESULT,AL ;store result
0010  C3                  RET
0011                BCDSUB  ENDP
0011                CODE    ENDS
                        END BCDSUB
```

80x86未併裝BCD算術調整指令

指令	動作	OF	SF	ZF	AF	PF	CF
AAA	在加算之後調整 AL 內容爲未併裝 BCD 。	U	U	U	*	U	*
AAS	在減算之後調整 AL 內容爲未併裝 BCD 。	U	U	U	*	U	*
AAM	在乘算之後調整 AL 內容爲未併裝 BCD 。	U	*	*	U	*	U
AAD	在除算之前調整 AH:AL 的未併裝 BCD 爲二進制值。	U	*	*	U	*	U

未併裝BCD加法

```
                    ;ex4.3-11.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 37               ADDEND  DB  37H     ;ASCII '7'
0001 33               AUGEND  DB  33H     ;ASCII '3'
0002 0000             RESULT  DW  00H     ;result
0004                DATA    ENDS
                      ....
0000                ASCIIA  PROC  NEAR
0000  B8 ---- R             MOV   AX,DATA   ;load DS
0003  8E D8                 MOV   DS,AX
0005  A0 0000 R               MOV   AL,ADDEND  ;get addend
0008  B4 00                 MOV   AH,00H    ;clear AH
000A  02 06 0001 R             ADD   AL,AUGEND  ;add them
000E  37                    AAA           ;decimal adjust
000F  A3 0002 R               MOV   RESULT,AX  ;store result
0012  C3                    RET
0013                ASCIIA  ENDP
0013                CODE    ENDS
                        END   ASCIIA
```

未併裝BCD乘法

```
                    ;ex4.3-12.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 37               MULTPLR  DB   37H      ;ASCII '7'
0001 33               MULTEND  DB   33H      ;ASCII '3'
0002 0000             RESULT   DW   00H      ;result
0004                DATA    ENDS
                        ....
0000  B8 ---- R             MOV  AX,DATA   ;load DS
0003  8E D8                 MOV  DS,AX
0005  A0 0000 R               MOV  AL,MULTPLR ;get multiplier
0008  24 0F                 AND  AL,0FH  ;clear upper nibble
000A  8A 1E 0001 R             MOV  BL,MULTEND ;get multend
000E  80 E3 0F               AND  BL,0FH  ;clear upper nibble
0011  F6 E3                 MUL  BL     ;multiply them
0013  D4 0A                  AAM          ;adjust result
0015  A3 0002 R               MOV  RESULT,AX  ;store result
0018  C3                    RET
0019                ASCIIM  ENDP
0019                CODE    ENDS
                        END  ASCIIM
```


未併裝BCD除法

```
                    ;ex4.3-13.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 3733             DIVIDND  DW   3733H    ;ASCII '73'
0002 33               DIVISOR  DB   33H      ;ASCII '3'
0003                RESULT  LABEL WORD
0003 00               QUOTIENT DB   00H      ;result(18H)
0004 00               REMAINDR DB   00H      ;(01H)
                        ....
0000  B8 ---- R             MOV  AX,DATA   ;load DS
0003  8E D8                 MOV  DS,AX
0005  8A 1E 0002 R             MOV  BL,DIVISOR ;get divisor
0009  80 E3 0F               AND  BL,0FH  ;clear upper nibble
000C  A1 0000 R               MOV  AX,DIVIDND ;get dividend
000F  25 0F0F                AND  AX,0F0FH;clear upper nibbles
0012  D5 0A                  AAD          ;adjust using AH
0014  F6 F3                 DIV  BL     ;divide BL
0016  A3 0003 R               MOV  RESULT,AX  ;store result
0019  C3                    RET
001A                ASCIID  ENDP
001A                CODE    ENDS
                        END  ASCIID
```

80x86條件性分歧指令

指令	動作	OF	SF	ZF	AF	PF	CF
Jcc rel8 Jcc rel16 Jcc rel32	在 cc 條件成立時 IP ← IP + 符號擴展(rel8)(80286↓) EIP ← EIP + 符號擴展之(rel8/16/32) 若運算元長度爲 16 位元則 EIP ← EIP AND 0000FFFFH 否則繼續執行下一個指令。	-	-	-	-	-	-
註:cc = Z/E, NZ/NE, S, NS, O, NO, P/PE, NP/PO, B/NAE, C, NB/AE, L/NGE, GE/NL, LE/NG, G/NLE, NC, BE/NA, NBE/A 。							

80x86條件性分歧指令的測試條件(cc)

類型	符號	若…則分歧	條件
單一位元	C	進位	CF = 1
	NC	未進位	CF = 0
	S	爲負	SF = 1
	NS	爲正	SF = 0
	E/Z	相等(=)/爲零	ZF = 1
	NE/NZ	不相等(≠)/不爲零	ZF = 0
	O	溢位	OF = 1
	NO	未溢位	OF = 0
	P/PE	偶同位	PF = 1
	NP/PO	奇同位	PF = 0
帶號數	L/NGE	<	SF ⊕ OF = 1
	NL/GE	≥	SF ⊕ OF = 0
	LE/NG	≤	(SF ⊕ OF) ∨ ZF = 1
	NLE/G	>	(SF ⊕ OF) ∨ ZF = 0
未帶號數	B/NAE	<	CF = 1
	NB/AE	≥	CF = 0
	BE/NA	≤	CF ∨ ZF = 1
	NBE/A	>	CF ∨ ZF = 0

陣列資料搬移

```
                        ;ex4.4-1.asm
0000                    DATA    SEGMENT PUBLIC 'DATA'
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
0010                    DATA    ENDS
                            ....
0000  B8 ---- R                   MOV  AX,DATA ;load DS
0003  8E D8                       MOV  DS,AX
0005  A0 0000 R                   MOV  AL,SRCA ;transfer 1st byte
0008  A2 0008 R                   MOV  DSTA,AL
000B  A0 0001 R                   MOV  AL,SRCA+1;transfer 2nd byte
000E  A2 0009 R                   MOV  DSTA+1,AL
                            ....
002F  A0 0007 R                   MOV  AL,SRCA+7;transfer 8th byte
0032  A2 000F R                   MOV  DSTA+7,AL
0035  C3                        RET
0036                    BLKMOV  ENDP
0036                    CODE    ENDS
                            END  BLKMOV
```


利用迴路的陣列資料搬移

```
                        ;ex4.4-2.asm
0000                    DATA    SEGMENT PUBLIC 'DATA'
= 0008                  LENTH   EQU  08H    ;bytes of array
0000 12 23                SRCA    DB   12H,23H ;source array
                            ....
0006  23 45                       DB   23H,45H
0008  0008 [ 00 ]         DSTA    DB   8 DUP(00);destination array
0010                    DATA    ENDS
                            ....
0005  B9 0008                     MOV  CX,LENTH;set count
0008  8D 36 0000 R                LEA  SI,SRCA ;set source pointer
000C  8D 3E 0008 R                LEA  DI,DSTA ;set dest. pointer
0010  8A 04             MLOOP:   MOV  AL,[SI] ;transfer them
0012  88 05                       MOV  [DI],AL
0014  46                        INC  SI     ;point to next
0015  47                        INC  DI     ;entry
0016  49                        DEC  CX
0017  75 F7                      JNZ  MLOOP  ;repeat count times
0019  C3                        RET
                            ....
                            END  BLKMOV
```

旗號迴路

```
                        ;ex4.4-4.asm
0000                    DATA    SEGMENT PUBLIC 'DATA'
0000 12 23                SRCA    DB   12H,23H ;source array
                            ....
0006  23 45                     DB   23H,45H
0008  FF                      DB   0FFH   ;end flag
0009  0008 [ 00 ]          DSTA    DB   8 DUP(00);destination array
                            ....
0005  8D 36 0000 R              LEA  SI,SRCA ;set source pointer
0009  8D 3E 0009 R              LEA  DI,DSTA ;set dest. pointer
000D  8A 04               MLOOP:  MOV  AL,[SI] ;transfer them
000F  3C FF                     CMP  AL,0FFH ;end ?
0011  74 06                     JE   DONE    ;yes, done
0013  88 05                     MOV  [DI],AL
0015  46                      INC  SI      ;point to next
0016  47                      INC  DI      ;entry
0017  EB F4                     JMP  MLOOP   ;continue
0019  C3                  DONE:   RET
001A                    BLKMOV  ENDP
001A                    CODE    ENDS
                          END   BLKMOV
```


80x86無條件分歧與跳躍指令

指令	動作	OF	SF	ZF	AF	PF	CF
(節區內：直接定址) JMP rel8 JMP rel16 JMP rel32 (80386↑)	IP ← IP + rel8/16 (80286↓) EIP ← EIP + rel8/16/32 (80386↑) 若運算元長度爲 16 位元則 EIP ← EIP AND 0000FFFFH	-	-	-	-	-	-
(節區內：間接定址) JMP r16/m16 JMP r32/m32 (80386↑)	IP ← r16/(m16) (80286↓) 若運算元長度爲 16 位元則 (80386↑) EIP ← r16/(m16) AND 0000FFFFH 否則 EIP ← r32/(m32)	-	-	-	-	-	-
(節區間：直接定址) JMP ptr16:16 JMP ptr16:32 (80386↑)	CS:IP← ptr16:16 (80286↓) 若運算元長度爲 16 位元則 (80386↑) CS:IP← ptr16:16 EIP ← EIP AND 0000FFFFH 否則(運算元長度爲 32 位元) CS:EIP ← ptr16:32	-	-	-	-	-	-
(節區間：間接定址) JMP m16:16 JMP m16:32 (80386↑)	CS:IP← (m16:16) (80286↓) 若運算元長度爲 16 位元則 (80386↑) CS:IP← (m16:16) EIP ← EIP AND 0000FFFFH 否則(運算元長度爲 32 位元) CS:EIP ← (m16:32)	-	-	-	-	-	-

整數平方根

```
   25          1
 -  1
   24          2
 -  3
   21          3
 -  5
   16          4
 -  7
    9          5 ←— 結果
 -  9
    0
```

```
   35          1
 -  1
   34          2
 -  3
   31          3
 -  5
   26          4
 -  7
   19          5 ←— 結果
 -  9
   10
 - 11
  - 1
```


整數平方根

```
                    ;ex4.4-6.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
0000 007B             NUMBER  DW  123   ;test number
0002 0000             SQRT    DW  ?     ;square root value
0004                DATA    ENDS
                        ....
0005  A1 0000 R             MOV  AX,NUMBER ;get test number
0008  B9 0001               MOV  CX,01H ;start value
000B  BB 0000               MOV  BX,00H ;clear count
000E  2B C1           AGAIN:  SUB  AX,CX  ;we have done it
0010  72 06                 JB   DONE   ;when AX < CX
0012  43                    INC  BX     ;increase count
0013  83 C1 02              ADD  CX,02H ;get next odd number
0016  EB F6                 JMP  AGAIN  ;continue
0018  89 1E 0002 R    DONE:   MOV  SQRT,BX;save result
001C  C3                    RET
001D                SQRT_FD ENDP
001D                CODE    ENDS
                        END  SQRT_FD
```

80x86迴路指令

指令	動作	OF	SF	ZF	AF	PF	CF
LOOP rel8 LOOPD rel8	計數器← 計數器 - 1; 若計數器不爲零則 IP ← IP + 符號擴展(rel8) 否則繼續執行下一個指令	-	-	-	-	-	-
LOOPE rel8 LOOPZ rel8 LOOPED rel8 LOOPZD rel8	計數器← 計數器 - 1; 若計數器不爲零並且 ZF = 1 則 IP ← IP + 符號擴展(rel8) 否則繼續執行下一個指令	-	-	-	-	-	-
LOOPNE rel8 LOOPNZ rel8 LOOPNED rel8 LOOPNZD rel8	計數器← 計數器 - 1; 若計數器不爲零並且 ZF = 0 則 IP ← IP + 符號擴展(rel8) 否則繼續執行下一個指令	-	-	-	-	-	-
JcounterZ rel8	若計數器不爲零則 IP ← IP + 符號擴展(rel8) 否則繼續執行下一個指令	-	-	-	-	-	-

註: 若位址長度爲 16 位元則計數器爲 CX ；若位址長度爲 32 位元則計數器爲 ECX 並且表中之 IP 更改爲 EIP 。

LOOP指令的使用例

```
                    ;ex4.4-7.asm
0000                DATA    SEGMENT PUBLIC 'DATA'
= 0008              LENTH   EQU  08H    ;bytes of array
0000 12 23          SRCA    DB   12H,23H ;source array
                      ....
0006  23 45               DB   23H,45H
0008  0008 [ 00 ]     DSTA    DB   8 DUP(00);destination array
0010                DATA    ENDS
                      ....
0003  8E D8               MOV  DS,AX
0005  B9 0008             MOV  CX,LENTH;set count
0008  8D 36 0000 R          LEA  SI,SRCA ;set source pointer
000C  8D 3E 0008 R          LEA  DI,DSTA ;set dest. pointer
0010  8A 04           MLOOP: MOV  AL,[SI] ;transfer them
0012  88 05               MOV  [DI],AL
0014  46                  INC  SI     ;point to next
0015  47                  INC  DI     ;entry
0016  E2 F8               LOOP MLOOP  ;repeat count times
0018  C3                  RET
0019                BLKMOV  ENDP
0019                CODE    ENDS
                        END  BLKMOV
```