第七章 程式流程控制和邏輯運算

跳越 (Jump)

迴圈 (Loop)

比較

條件式跳越 (Conditional jump)

邏輯運算

移位

旋轉

相關指令

比較運算	程式轉移動作	邏輯運算	移位和旋轉
CMP	CALL	AND	SAR/SHR
TEST	JMP	NOT	SAL/SHL
	Jnnn*	OR	RCR/ROR
	LOOP	XOR	RCL/ROL
	RETn		
* nnn 表示條件, 例如 JNE 和 JL			

短距、近距和遠距地址

	短距	近距	遠距
指令	-128 到 127 相同分段	-32,768 到 32,767 相同分段	超過 32K 或在另一分段
JMP	是	是	是
Jnnn	是	是 (386+)	否
LOOP	是	否	否
CALL	沒有	是	是

跳越

格式

[Label:] JMP short/near/far address

往回跳

```
L10:
     …
     jmp       L10
```

往前跳

```
     jmp       L20
     …
L20:
```

使用JMP的程式例子

```
titlea07jump (COM) using jump for looping
     .model    small
     .code
     org  100h       ;start at end of PSP
a10main    proc     near
     mov  ax,0       ;initialize ax
     mov  bx,0       ; and bx to zero,
     mov  cx,1       ; cx to 1
a20:
     add  ax,1       ;increment ax
     add  bx,ax      ;add ax to bx
     shl  cx,1       ;double cx
     jmp  a20        ;jump to a20 label
a10main    endp
     end  a10main
```

迴圈

格式

[Label:] LOOP short address

CX減1;

若結果不是0, 跳到指定位置

若結果是0, 作接著的指令

使用LOOP的程式例子

```
titlea07loop (COM) illustration of loop
     .model    small
     .code
     org  100h
a10main    proc      near
     mov  ax,0       ;initialize ax and
     mov  bx,0       ; bx to zero,
     mov  dx,1       ; dx to 1
     mov  cx,8       ;initialize for
a20:                 ; 8 loops
     inc  ax         ;increment ax
     add  bx,ax      ;add ax to bx
     shl  dx,1       ;double dx
     loop a20        ;decrement cx,
                     ; loop if nonzero
     mov  ax,4c00h   ;end processing
     int  21h
a10main   endp
     end  a10main
```

CMP指令

格式

[Label:] CMP register/memory,register/memory/immediate

```
    cmp      dx,0
    je       L10
    …
L10:
```

條件式跳越

格式

[Label:] Jnnn short address

```
DEC     CX
JNZ     A20
```

無正負號數的比較

大於: Above

小於: Below

等於: Equal, Zero

指令名稱	說明	測試旗標
JE/JZ	相等或是零則跳越	ZF
JNE/JNZ	不相等或不是零則跳越	ZF
JA/JNBE	大於或不小於/等於則跳越	CF, ZF
JAE/JNB	大於/等於或不小於則跳越	CF
JB/JNAE	小於或不大於/等於則跳越	CF
JBE/JNA	小於/等於或不大於則跳越	ZF, CF

有正負號數的比較

大於: Greater than

小於: Less than

等於: Equal, Zero

指令名稱	說明	測試旗標
JE/JZ	相等或是零則跳越	ZF
JNE/JNZ	不相等或不是零則跳越	ZF
JG/JNLE	大於或不小於/等於則跳越	OF, SF, ZF
JGE/JNL	大於/等於或不小於則跳越	OF, SF
JL/JNGE	小於或不大於/等於則跳越	OF, SF
JLE/JNG	小於/等於或不大於則跳越	OF, SF, ZF

單獨旗標測試

指令名稱	說明	測試旗標
JCXZ	CX 是零則跳越	無
JC	進位是 1 則跳越	CF
JNC	進位是 0 則跳越	CF
JO	OF 是 1 則跳越	OF
JNO	OF 是 0 則跳越	OF
JP/JPE	PF 是 1 則跳越	PF
JNP/JPO	PF 是 0 則跳越	PF
JS	SF 是 1 則跳越	SF
JNS	SF 是 1 則跳越	SF

多個條件的測試

只要任一個符合

```
        CMP     AL,BL
        JE      EQUAL
        CMP     AL,BH
        JE      EQUAL
        CMP     AL,CL
        JE      EQUAL
NOT_EQUAL:  …
EQUAL:      …
```

多個條件的測試

所有條件符合

```
        CMP     AL,BL
        JNE     NOT_EQUAL
        CMP     AL,BH
        JNE     NOT_EQUAL
        CMP     AL,CL
        JNE     NOT_EQUAL
EQUAL:  …
NOT_EQUAL:      …
```

呼叫程序 (Procedure)

結構化的程式設計
經由呼叫副程式, 節省程式空間
容易測試和維護

```
PROC_NAME  PROC  FAR
…
PROC_NAME  ENDP
```

CALL 和 RETn 指令

格式

[Label:] CALL procedure-name

[Label:] RET[n] [immediate]

n: N 或 F 代表 Near 或 Far。
RETN 將POP IP, SP 加 2
RETF 將 POP IP和CS, SP 加 4
沒有n, 組譯程式將依PROC敘述中定義的, 自動產生合適的碼
呼叫procedure時, 亦是如此 (Near call, Far call)
Immediate表示SP應再加上的值

程序呼叫的例子

```
   .model     small
   .stack     64
   .data
;-------------------------------------------
   .code
a10main proc  far
       call   b10           ;call  b10
;  ...
       mov    ax,4c00h      ;end processing
       int    21h
a10main endp
;-------------------------------------------
b10    proc   near
       call   c10    ;call c10
;  ...
       ret           ;return to caller
b10    endp
;-------------------------------------------
c10    proc   near
;  ...
       ret           ;return to caller
c10    endp
;-------------------------------------------
       end    a10main
```

堆疊的改變狀況

最早時, SP=40H (定義了64個位置)

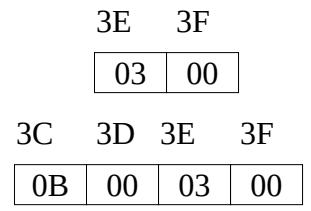
1. call b10 時, IP=3 (已指到下一個指令的地址)

b10是 near procedure

SP減2(變為3EH), 存IP到堆疊

2. Call c10時, IP=000BH

SP減2(變為3CH), 存IP到堆疊



3. 由 Procedure c10中 return
將堆疊頂端兩位元組pop到 IP (IP=000BH)
SP加2, 變成3EH
4. 由 Procedure b10中 return
將堆疊頂端兩位元組pop到 IP (IP=0003H)
SP加2, 變成40H

呼叫副程式時傳遞參數

傳值或傳地址
傳陣列通常傳地址
使用暫存器傳或使用堆疊傳

例子: 使用暫存器傳參數

```
          LEA       BX,MULTICAND
          LEA       SI,MULTIPLER
          CALL      M30MULT
     …
M30MULT   PROC      NEAR
          MOV       AX,[BX]
          MUL       WORD PTR [SI]
          RET
M30MULT   ENDP
```

例子: 使用堆疊傳參數

```
.386
          PUSH        OFFSET MULTICAND
          PUSH        OFFSET MULTIPLER
          CALL        M30MULT
          …
M30MULT   PROC  NEAR
          PUSH  BP
          MOV   BP,SP
          MOV   BX,[BP+6]
          MOV   DI,[BP+4]
          MOV   AX,[BX]
          MUL   WORD PTR [DI]
          POP   BP
          RET   4
M30MULT   ENDP
```

C 程式呼叫組合語言副程式

C程式中

```
Adds(value_1, value2);
```

組合語言的Adds副程式

```
PUSH  BP
MOV   BP,SP
MOV   DH,[BP+4]
MOV   DL,[BP+6]
…
POP   BP
RET
```

Old BP	BP+0
Return address	BP+2
value_1	BP+4
value_2	BP+6

邏輯運算

[Label:] operation register/memory,register/memory/immediate

OPERATIONS:

AND

OR

XOR

TEST

NOT

AND 指令

用以將暫存器或記憶體的部分位元變為0 (保留其它位元), 或測試其它位元是否都是0

例子: BL=0011 1010, CH=1010 0011

```
AND BL,0FH ;將BL的高階四位元變為0 (低階四位元不變), 亦可能是檢查低階是位元是否都是0 (若是, 則結果是0, ZF=1)
AND CH, 02H ;
```

OR 指令

用以將暫存器或記憶體的部分位元變為1 (其它位元不變)

例子: BL=0011 1010, CH=1010 0011

```
OR BL,CH ;BL=1011 1011
OR CH, CH ;若CH=0, 則ZF=1; 否則, ZF=0
          ;測試CH是否是0
```

XOR 指令

用以將暫存器或記憶體的部分位元反相
(1→0,0→1; 其它位元不變)

例子: BL=0011 1010

XOR BL,0FH ;BL=0011 0101

XOR CH, CH ;CH=0

TEST 指令

動作與AND指令相同, 但不改變目的暫存器的值, 只改變旗標值

例子:

TEST CX,0FFH

TEST BL,01H

TEST CL,11110000B

NOT 指令

[Label:] NOT register/memory

用以將暫存器或記憶體的所有位元反相

例子: BL=0011 1010
NOT BL ;BL=1100 0101

程式例子: 大小寫轉換

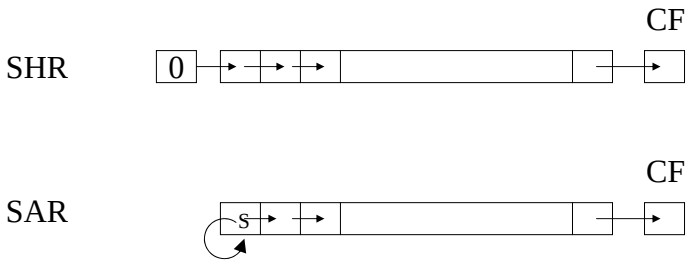
英文字母的 ASCII 碼
大寫 A: 41H (0100 0001), …, Z: 5AH (0101 1010)
小寫 a: 61H (0110 0001), …, z: 7AH (0111 1010)
大小寫字碼的差別在位元5; 大寫是0, 小寫是1

欲作大小寫互換, 應將字碼與0010 0000作XOR
欲使所有字變小寫, 應將字碼與0010 0000作OR
欲使所有字變大寫, 應將字碼與1101 1111作AND

```
title   a07case (COM) change uppercase to lowercase
        .model small
        .code
        org     100h
begin:  jmp     a10main
;-----------------------------------------------
coname  db      'LASER-12 SYSTEMS', '$'
;-----------------------------------------------
a10main proc    near
        lea     bx,coname+1     ;1st char to change
        mov     cx,15           ;no. of chars to change
a20:
        mov     ah,[bx]         ;character from coname
        cmp     ah,41h          ;is it
        jb      a30             ; uppercase
        cmp     ah,5ah          ; case
        ja      a30             ; letter?
        xor     ah,00100000b    ;yes, convert
        mov     [bx],ah         ;restore in coname
a30:
        inc     bx              ;set for next char
        loop    a20             ;loop 15 times
                                ;done,
        mov     ah,9            ; display
        lea     dx,coname       ; coname
        int     21h
        mov     ax,4c00h        ;end processing
        int     21h
a10main endp
        end     begin
```

移位: 右移

[Label:] SHR/SAR register/memory, CL/immediate

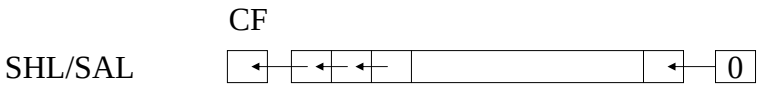


SHR/SAR 的例子

```
    MOV  BH,10110111B   ;(183)
    SHR  BH,1           ;01011011 (91), CF=1
    MOV  CL,2
    SHR  BH,CL          ;00010110 (22), CF=1
    SHR  BH,2           ;00000101 (5), CF=1
;
    MOV  BH,10110111B   ;(-73)
    SAR  BH,1           ;11011011 (-37), CF=1
    MOV  CL,2
    SAR  BH,CL          ;11110110 (-10), CF=1
    SAR  BH,2           ;11111101 (-3), CF=1
```

移位: 左移

[Label:] SHL/SAL register/memory, CL/immediate

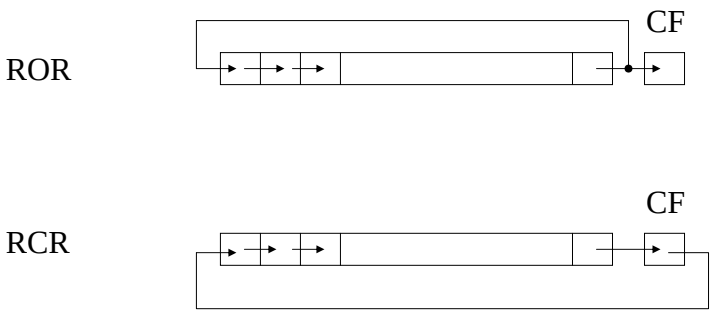


SHL/SAL 的例子

```
MOV  BH,00000101B   ;(5)
SHL  BH,1           ;00001010 (10), CF=0
MOV  CL,2
SHL  BH,CL          ;00101000 (40), CF=0
SHL  BH,2           ;10100000 (160), CF=0
```

旋轉: 右轉

[Label:] ROR/RCR register/memory, CL/immediate

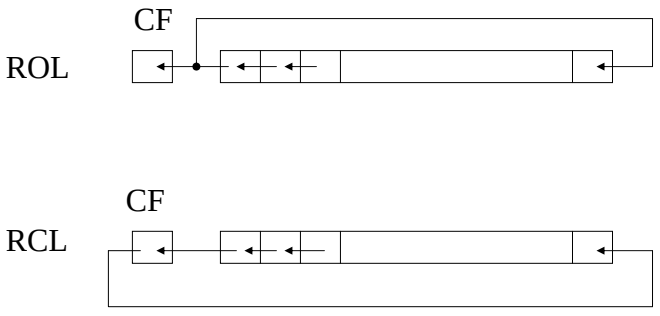


ROR/RCR 的例子

```
     MOV  BH,10110111B   ;
     ROR  BH,1           ;11011011, CF=1
     MOV  CL,3
     ROR  BH,CL          ;01111011, CF=0
     ROR  BH,3           ;01101111, CF=0
;
     CLC                 ;CF=0
     MOV  BH,10110111B   ;
     RCR  BH,1           ;01011011, CF=1
     MOV  CL,3
     RCR  BH,CL          ;11101011, CF=0
     RCR  BH,3           ;11011101, CF=0
```

旋轉: 左轉

[Label:] ROL/RCL register/memory, CL/immediate



ROL/RCL 的例子

```
        MOV  BH,10110111B   ;
        ROL  BH,1           ;01101111, CF=1
        MOV  CL,3
        ROL  BH,CL          ;01111011, CF=1
        ROL  BH,3           ;11011011, CF=1
;
        CLC                 ;CF=0
        MOV  BH,10110111B   ;
        RCL  BH,1           ;01101110, CF=1
        MOV  CL,3
        RCL  BH,CL          ;01110101, CF=1
        RCL  BH,3           ;10101101, CF=1
```