第六章 指令和定地址法介紹

指令簡介
定地址法簡介

計算機指令的分類

- 運算
  - 算術
  - 邏輯
  - 移位、旋轉
- 資料移動
- 控制

算術運算

ADC: 連同進位的加法
ADD: 加法
DEC, INC: 減一, 增一
DIV, IDIV: 除法(Unsigned), 整數(二補數)除法
MUL, IMUL: 乘法, 整數乘法
NEG: 二補數
SBB: 連同借位的減法
SUB: 減法
XADD: 交換, 相加

十進制(BCD)轉換和調整

AAA: 加法後的ASCII調整
AAD: 除法前的ASCII調整
AAM: 乘法後的ASCII調整
AAS: 減法後的ASCII調整
DAA: 加法後的十進制調整
DAS: 減法後的十進制調整

位元移動

RCL, RCR: 經進位旗標向左旋轉, 向右旋轉
ROL, ROR: 向左旋轉, 向右旋轉
SAL, SAR:算術性向左移位, 向右移位
SHL, SHR:邏輯性向左移位, 向右移位
SHLD, SHRD:雙字組向左移位, 向右移位

比較

BSF/BSR: 位元掃描
BT/BTC/BTR/BTS: 位元測試
CMP: 比較 (相減, 但不存結果, 只改變旗標)
CMPSn: 比較字串
CMPXCHG: 比較和交換
CMPXCHG8B:比較和交換
TEST: 測試位元(AND, 但不存結果, 只改變旗標)

資料移動

LDS, LES, LSS: 載入DS, ES, SS 暫存器
LEA: 載入有效地址 (將所指資料的地址放到暫存器)
LODSn: 載入字串 (SI所指位置內容放到暫存器A, 增減SI)
MOV: 移動資料
MOVSn: 移動字串 ([SI]→[DI},增減SI, DI)
MOVSX: 移動資料(到較大位置),延伸正負號
MOVZX:移動資料(到較大位置), 延伸0
STOSn: 貯存字串(暫存器A內容放到DI所指位置, 增減DI)
XCHG: 交換資料
XLAT: 翻譯 (以BX為基底, AL為索引, 取資料放到AL)

旗標操作

CLC, CLD, CLI: 清除進位、方向、插斷旗標
CMC: 將進位旗標反相 (補數)
LAHF: 將旗標載入AH
POPF: 由堆疊彈出旗標
PUSHF: 將旗標堆入堆疊
SAHF: 將AH存到旗標
STC, STD, STI: 設定進位、方向、插斷旗標

輸入/輸出

IN: 輸入
INSn: 輸入字串
OUT: 輸出
OUTSn: 輸出字串

邏輯運算(位元對位元)

AND:
NOT:
OR:
XOR:

迴圈

LOOP: CX減一; 若CX不是0, 跳到指定位置

LOOPE, LOOPZ: CX減一; 若CX不是0, 或 ZF=1, 跳到指定位置

LOOPNE, LOOPNZ: CX減一; 若CX不是0, 或ZF=0, 跳到指定位置

LOOPNEW, LOOPNZW:

處理機控制

HLT: 停止

LOCK: 鎖住匯流道

NOP: 無動作

WAIT: 等待TEST信號

堆疊操作

ENTER, LEAVE:

POP, PUSH: 彈出, 堆入

POPF, PUSHF:彈出, 堆入旗標

POPA, PUSHA:彈出, 堆入所有一般用途暫存器

字串操作

CMPSn: 比較字串

LODSn: 載入字串

MOVSn: 移動字串

SCASn: 掃描字串

STOSn: 貯存字串

REP: 重覆 (前置指令, 必須配合字串指令)

REPE, REPZ: 若相等(ZF=1), 則重覆

REPNE, PRENZ: 若不相等(ZF=0), 則重覆

程式移轉(有條件)

INTO: Interrupt on overflow
JA/JNBE: Jump if above (not below or equal)
JAE/JNB: Jump if above or equal
JB/JNAE: Jump if below
JBE/JNA: Jump if below or equal
JC: Jump if CF=1 (有進位)
JCXZ: Jump if CX=0

JE/JZ: Jump if ZF=1 (zero, equal)
JG/JNLE: Jump if greater than (not less than or equal)
JGE/JNL: Jump if greater than or equal
JL/JNGE: Jump if less than
JLE/JNG: Jump if less than or equal
JNC: Jump if CF=0
JNE/JNZ: Jump if ZF=0
JNO: Jump if OF=0 (no overflow)
JNP/JPE: Jump if PF=0 (parity even)

JNS: Jump if SF=0 (正數)
JO: Jump if OF=1 (Overflow)
JP/JPO: Jump if PF=1 (Parity Odd)
JS: Jump if SF=1 (負數)

程式移轉(無條件)

CALL: 呼叫副程式
INT: 插斷
IRET: 插斷回轉
JMP: 跳越
RET: 回轉
RETN/RETF: 回轉(近距)/(遠距)

資料型式轉換

CBW: Byte → word
CDQ: Word → quadword
CWD: Word → doubleword
CWDE: Word → extended doubleword

資料移動指令

MOV
MOVSX, MOVZX
XCHG
LEA

MOV 指令

格式

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

```
BYTEFLD      DB  ?
WORDFLD      DW  ?
             MOV EDX,ECX
             MOV BYTEFLD,DH
             MOV [DI],BX
```

```
        MOV     CX,40H
        MOV     BYTEFLD,25
        MOV     WORDFLD[BX],16H
        MOV     CH,BYTEFLD
        MOV     CX,[WORDFLD[BX]
        MOV     AX,DS
        MOV     WORDFLD,DS
```

錯誤的敘述

```
MOV     DL,WORDFLD
MOV     CX,BYTEFLD
MOV     WORDFLD,EBX
MOV     BYTE_VAL1,BYTEVAL2
MOV     ES,225
MOV     ES,DS
```

MOVSX 和 MOVZX

格式

[Label:] MOVSX/MOVZX register/memory, register/memory/immediate

```
MOVSX   CX,10110000B    ;CX=FFB0H
MOVZX   CX,10110000B    ;CX=00B0H
```

```
BYTE1   DB  25
WORD1   DW  40
DWORD1  DD  160
.386
        MOVSX   CX,BYTE1
        MOVZX   WORD1,BH
        MOVZX   EBX,WORD1
        MOVZX   DWORD1,CX
```

XCHG 指令

[Label:] XCHG register/memory, register/memory

```
WORDQ   DW  ?
        …
        XCHG    CL,BH
        XCHG    CX,WORDQ
```

LEA 指令

[Label:] LEA register, memory

```
DATATBL     DB  25 DUP (?)
BYTEFLD     DB  ?
            …
            LEA BX,DATATBL
            MOV BYTEFLD,[BX]
```

記憶體到記憶體??

相當於
```
MOV  BX,OFFSET DATATBL
```

基本算術指令

INC, DEC

ADD, SUB

INC, DEC指令

[Label:] INC/DEC register/memory

會改變 OF, SF 和 ZF

ADD, SUB指令

[Label:] ADD/SUB register/memory, register/memory/immediate

```
ADD  AX,CX
ADD  EBX,DBLWORD
SUB  BL,10
```

程式例子: 移動多個位置資料

```
title a06move (exe) extended move operations
;-----------------------------------------------
  .model    small
  .stack    64
;-----------------------------------------------
  .data
headng1     db    'InterTech'
headng2     db    9 dup ('*'), '$'
;-----------------------------------------------
  .code
a10main proc      far
  mov ax,@data    ;initialize segment
  mov ds,ax       ; registers
  mov es,ax
;
  mov cx,9        ;initialize to move 9 chars
  lea si,headng1  ;initialize address of headng1
  lea di,headng2  ; and headng2
```

```
 a20:
   mov al,[si]     ;get character from headng1
   mov [di],al     ; move it to headng2
   inc si          ;incr next char in headng1
   inc di          ;incr next pos'n in headng2
   dec cx          ;decrement count for loop
   jnz a20         ;count not zero? yes, loop
                   ; finished
   mov ah,9        ;request display
   lea dx,headng2  ;of headng2
   int 21h
 ;
   mov ax,4c00h    ;end processing
   int 21h
 a10main endp                  ;end of a procedure
   end a10main     ;end of program
```

定地址模式 (Addressing Mode)

指示運算元的位置或值的方式, 有

暫存器: 暫存器的內容

立即 (Immediate): 運算元的值在指令中

直接 (Direct): 運算元的地址在指令中

間接 (Indirect):運算元的地址在暫存器中(可能還有一些計算)

一個指令中可能有多個不同定地址模式的運算元

暫存器運算元

```
WORDA   DW      ?
        …
        MOV     DX,WORDA
        MOV     WORDA,CX
        MOV     EDX,EBX
```

立即運算元

```
COUNT   DB      ?
        …
        ADD     BX,25
        MOV     COUNT,50
```

直接地址

```
WORDA   DW      0
BYTEA   DB      0
        …
        MOV     BX,WORDA
        ADD     BYTEA,DL
        MOV     CX,DS:[38B0H]
        INC     BYTE PTR [1B0H]
```

間接地址

```
DATAVAL   DB      50
          …
          MOV     BX,OFFSET DATAVAL
          MOV     BYTE PTR [BX],25
          MOV     BYTE PTR [BX+2],0
          ADD     CL,[BX]
          MOV     BYTE PTR [DI],25
          ADD     [BP],CL
```

基底位移定地址法

```
DATA_TBL  DB      365 DUP (?)
          …
          MOV     BX,OFFSET DATA_TBL
          MOV     BYTE PTR [BX+2],0
          ADD     CL,[DI+12]
          SUB     DATA_TBL[SI],25
          MOV     DATA_TBL[DI],DL
```

基底索引定地址法

```
MOV     AX,[BX+SI]
ADD     [BX+DI],CL
```

基底索引加位移定地址法

```
        MOV     AX,[BX+SI+10]
        MOV     CL,DATA_TBL[BX+DI]
.386
        MOV     EBX,[ECX*2+ESP+4]
```

Segment Override Prefix

```
MOV     DX,ES:[BX]
MOV     ES:[SI+36],CL
```

組譯程式在指令的程式碼之前加前置碼 26H; 相同於

```
ES:MOV  DX,[BX]
ES:MOV  [SI+36],CL
```

Near 和 Far 地址

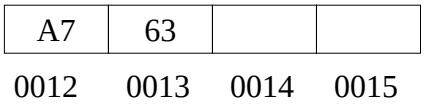
Near 地址: 16位元 (Offset)

Far 地址: 32位元 (Segment號碼和Offset)

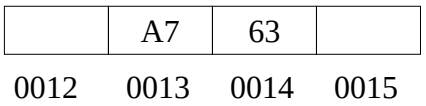
資料的對齊 (Align)

AX=63A7H, SI=0012H, DI=0013H

MOV [SI],AX



MOV [DI],AX



資料的對齊

在資料段中, 依次放四字組、雙字組、字組和位元組