第九章 NASM組合語言

- NASM 組合語言的英文名稱為 Netwide Assembler，簡稱 NASM，它是專為 Intel 公司出品的 80x86 處理器所組成的電腦而設計的，是一種可攜性高且具模組性的一種組合語言，支援多種目的檔格式，包括 Linux 作業系統的 a.out 及 ELF 格式、NetBSD/FreeBSD 格式、COFF 格式、微軟公司的十六位元 OBJ 格式、以及 Win32 格式等，也支援純二進位檔案的輸出格式。
- NASM 的語法支援 Pentium、P6 以及 MMX 機種的運算碼，同時具備巨集的功能。有關 NASM 的軟體及參考文件可從下列網站免費下載：
-
- http://sourceforge.net/projects/nasm

9.1 電腦硬體

- 電腦硬體其內部主要部份為處理器、記憶體、以及暫存器。暫存器其實是處理器中特殊的元件，用於儲存位址及資料。電腦硬體其外部主要部份為電腦的輸入輸出設備包括鍵盤、螢幕、磁碟機以及 CD-ROM 等等不勝枚舉。軟體包含作業系統、程式以及資料檔等，通常儲存於磁碟上。
- 要執行一個程式，作業系統將程式從外部的設備，通常為硬碟，拷貝進入記憶體裡，處理器執行這個程式的指令，而暫存器處理所需的算術運算、資料的搬移、以及位址的運算等工作。
- 個人電腦的腦部就是處理器，也稱為中央處理單元，對於 80x86 家族而言，處理器負責所有指令的執行以及資料的處理。各種處理器對於處理速度、記憶體容量、暫存器以及資料匯流排等均有所差異，本小節只作一般性的說明。

9.1.1 暫存器

- 處理器中的暫存器用於控制正在執行中的指令，包括處理記憶體定址以及提供算術運算的能力。在程式中要使用暫存器可直接使用它的名稱就行了，例如 CS、DS、及 SS 等。暫存器位元編號也是從 0 算起，由右往左排列。
- 段暫存器　CS 、DS 、SS 、 ES。
- 指標暫存器 IP 、SP 、BP。
- 通用暫存器 AX 、BX 、CX 、DX。
- 索引暫存器 SI 、DI。
- 旗標暫存器 CF 、PF 、AF 、 ZF 、SF 、TF 、
- IF、DF 、OF。

硬體插斷

- 某些事件導致處理器暫停正在執行的工作，轉而處理該事件，處理完畢後又繼續剛剛暫停的工作。有些事件是正常而經常發生的，例如從鍵盤輸入資料產生一個事件，導致處理器暫停正在執行的工作，轉而呼叫 BIOS 中從鍵盤輸入資料的程序，處理完畢後又繼續剛剛暫停的工作。有些事件是不正常的，例如您執行除法，但不小心讓除數為零值，導致處理器停止執行您的程式。
- 另一種為軟體插斷，您的程式本身發出一個請求，要將資料顯示在螢幕上，導致處理器暫停正在執行的工作，轉而呼叫 BIOS 中將資料顯示在螢幕上的程序，處理完畢後又繼續剛剛暫停的工作。

9.2 NASM指令格式

- 原始程式列格式為包含四欄的組合，這四欄分別為：標籤、指令、運算元、註解。
- **[標籤] [指令] [運算元] [註解]**
- 這四欄均為選用性質，方括號表示選用。每欄之間可以擺放一個或多個空白。
- ***標籤欄*** 由標籤識別字及冒號組成。標籤識別字由下列字元所組成：
- **_ $ # @ ~ . ? 英文字母 數字**
- ***標籤識別字*** 第一個字元只能是英文字母、句點「.」、底線「_」或是問號「?」，對於大小寫英文字母，認為是不同的識別字。
- ***指令欄*** 包含任何的機器指令，前方可以伴隨著 LOCK、REP、REPE/REPZ、REPNE/REPNZ 等等。

NASM指令格式

- ***運算元欄*** 包括下列的形式：
- **1. 暫存器**　　暫存器名稱，
- 　　例如 AX、EBX、CL 等。
- **2. 有效位址**　表示記憶體位址。
- **3. 常數**　　可為數字常數、字元常數、
- 　　字串常數、或浮點數常數。
- **4. 運算式**　　運算式為暫存器、有效位址、
- 　　常數以及運算子的結合，
- 計算的結果當為一個運算元。
- ***註解欄*** 以分號開始，其後跟隨著註解。

9.3 虛擬指令

- 虛擬指令並非真正的機器指令，它指示組合語言如何組譯程式的一種指令，它要擺放在原始程式列的指令欄位置，因此稱呼它為虛擬指令。NASM 所支援的虛擬指令有：DB、DW、DD、DQ、DT、RESB、RESW、RESD、RESQ、REST、INCBIN、EQU、TIMES。

9.3.1 宣告含有初值的資料

- 虛擬指令 DB 宣告含有初值的位元組資料，DW 宣告含有初值的字組資料，DD 宣告含有初值的雙字組資料，DQ 宣告含有初值的四字組資料，DT 宣告含有初值的十位元組資料。下列均為合乎語法的宣告：
-
- DB 31H, 32H, 33H ;初值為十六進位數31、32、33
- DB 'a', 62H ;初值為字元常數'a'及十六進位數62
- DB 'Hello',13,10,'$‘ ;初值為字串常數'Hello'及十進位
- ;數13、10及字元常數'$'
- DW 1234H, 5678H ;初值為十六進位數1234及5678

9.3.2 宣告不含初值的資料

- RESB、RESW、RESD、RESQ、及 REST 等虛擬指令宣告不含初值的資料，事實上這些虛擬指令只保留一些記憶體空間而已，RES 為 reserve 保留的縮寫，B 表示 Byte 位元組，W 表示 Word 字組，D 表示 Double word 雙字組，Q 表示 Quad word 四字組，T 表示 Ten byte 十個位元組的空間。下列均為合乎語法的宣告：
-
- buf RESB 64 ;保留64個位元組的記憶體空間
- wvar RESW 1 ;保留一個字組的記憶體空間
- rbuf RESQ 10 ;保留10個四字組的記憶體空間

9.3.3 INCBIN 虛擬指令

- INCBIN 虛擬指令引入二進位檔，引入的二進位檔可以是圖片檔或聲音檔，這對於遊戲程式的設計有很大的幫助。INCBIN 虛擬指令的使用有下列三種格式：
-
- **INCBIN "pic.dat" ;引入全部檔案**
- **INCBIN "pic.dat",1024 ;跳過前面1024個位元組**
- **INCBIN "pic.dat",1024,512 ;跳過前面1024個位元組**
- **;最多引入512個位元組**

9.3.4 EQU 虛擬指令

- EQU 虛擬指令定義一個常數給一個符號，當您使用 EQU 虛擬指令時，原始程式列必須包含一個標籤，這個標籤就是所謂的符號。這個符號一經定義後就不能改變了。例如：
-
- msg DB 'Hello, world'
- msglen EQU $-msg
-
- 「$」符號表示這一原始程式列開始的位址，它減去 msg 的位址後，得到的值為 12，它表示 msg 這個字串的長度，在這個程式中 msglen 的值永遠為 12，不能改變。

9.3.5 TIMES 虛擬指令

- TIMES 虛擬指令的格式如下：
- **TIMES 次數 指令**
- TIMES 虛擬指令將隨後的「指令」執行指定的「次數」。如下例：
- **buf: TIMES 64 DB 0**
- 宣告 64 個位元組，每個位元組的內含為 0。次數可以是常數，也可以使用運算式，如下例：
- **buf: DB 'Hi'**
- **TIMES 64-$+buf DB ' '**
- 宣告一個 64 個位元組的空間，前面的 12 個位元組的內含為 'Hi'，其他 62 個位元組的內含為空白。
- **label2: TIMES 100 MOVSB**
- 將 MOVSB 指令重複 100 次。
- TIMES 虛擬指令不能使用在 MACRO 巨集裡。

9.4 有效位址

- 有效位址為指令的一個運算元，用於參考到一個記憶體位址。對於 NASM 來講，有效位址的計算非常簡單，將在方括號內的運算式計值，該值就是一個有效位址。如下例：

```
        var     DB      'a', 'b', 'c', 'd'
        buf     DB      '1', '2', '3', '4'
                MOV   AL, [var+1]
                MOV   [buf], AL
```

9.5 常數

- NASM 支援四種不同的型態：數字常數、字元常數、字串常數以及浮點數常數。
- ***數字常數*** 簡單地講就是數字，NASM 支援數種基底的數字常數，只要 在數字的後頭附上「H」或「h」表示該數字為十六進位數，附上「Q」 或「q」表示該數字為八進位數，附上「B」或「b」表示該數字為二 進位數，若沒有附上則表示十進位數。NASM 也支援數字前頭附上 「0X」或「0x」或「$」表示該數字為十六進位數。

字元常數

- 字元常數最多包括四個字元，可使用單引號或雙引號將它們括起來。 使用單引號或雙引號對於 NASM 都表示同一個意思，它的好處是當您 使用單引號時，雙引號就可以當資料字元來使用，同理當您使用雙引號時，單引號就可以當資料字元來使用。字元常數若超過一個字元時 ，要注意前面的字元擺在低位址，後面的字元擺在高位址。

字串常數

- 字串常數只適用於虛擬指令 DB 一族以及 INCBIN 指令。字串常數看起來像字元常數，只不過長一點而已。
-
- DB 'Hello' ;字串常數
- DB 'H', 'e', 'l', 'l', 'o' ;字元常數相當於上例字串常數
-
- 下列三個指令也是相當的：
-
- DD 'ninechars' ;雙字組字串常數
- DD 'nine', 'char', 's' ;相當於三個雙字組字元常數
- DB 'ninechars', 0,0,0 ;相當於12個位元組

浮點數常數

- 浮點數常數只適用於虛擬指令 DD、DQ 及 DT。它們以傳統的方式表示出來，它的格式如下：
-
- 數字 . [數字] [[E | e] [+ | -] 指數]
-
- 方括號表示選用。如下例：
-
- DD 365.25 ;浮點數常數365.25
- DD 3.6525e+2 ;浮點數常數365.25

9.6 運算式

- NASM 的運算式語法類似 ANSI C 語言的語法。NASM 的運算式支援兩個特殊的符號：$ 及 $$。$ 表示包含此運算式原始程式列開始的位址。而 $$ 表示目前的段址。$ - $$ 表示包含此運算式原始程式列開始的位址離開該段的距離（位元組數）。
-

- 逐位元 OR 運算子 「|」與指令「OR」的作用相同。
- 逐位元 XOR 運算子 「^」與指令「XOR」的作用相同。
- 逐位元 AND 運算子 「&」與指令「AND」的作用相同。
- 移位運算子 「<<」位元往左移位，「>>」位元往右移位。
- 加及減運算子 加「+」表示算術加法，減「-」表示算術減法。
- 乘及除運算子 乘「*」表示乘法，除「/」表示無號整數除法。「//」表示有號整數除法，餘數「%」表示取無號整數除法的
- 餘數 「%%」運算子表示取有號整數除法的餘數。
- 單元運算子 「+」、「-」、「~」、及 SEG 運算子。

9.7 區域標籤

- 以句點「.」起頭的符號 NASM 會作特殊處理，將它視為區域標籤，必須與非區域標籤結合才可當成獨立的標籤。如下例：
- label2: ;某些程式碼
- .loop ;某些程式碼
- JNE .loop ;事實上跳至 label2.loop
- RET
- label3: ;某些程式碼
- .loop ;某些程式碼
- JNE .loop ;事實上跳至 label3.loop
- RET
- 上例的程式碼中，兩個區域標籤均命名為「.loop」，但它的位置並不相同，上一個區域標籤「.loop」與它的前一個非區域標籤「label2」相結合成「label2.loop」。下一個區域標籤「.loop」與它的前一個非區域標籤「label3」相結合成「label3.loop」。

9.8 前置處理器

- 所有的前置處理器指引指令均以「%」開頭。包括單列巨集、多列巨集、條件組譯、前置處理迴圈、引入檔、標準巨集等。

單列巨集 %define 指令

- 單列巨集包括 %define、%undef、及 %assign 指令，說明如下
- 單列巨集以前置處理器指引指令 %define 來定義。例如：
- **%define sum(a,b) a+b**
- **MOV DL, sum(64,1) ;將 64+1 的結果存入 DL**
- 單列巨集也可以定義沒有參數的巨集，例如：
- **%define pi 3.14159**
- 定義符號 pi 為 3.14159。

單列巨集 %undef 指令

- %undef 指令
-
- %undef 指令取消 %define 所定義的巨集，例如：
-
- %define pi 3.14159 ;**定義pi符號**
- … ;**其他程式碼**
- %undef pi ;**取消%define所定義的巨集**pi

單列巨集 %assign 指令

- %assign 指令定義一個單列巨集，沒有參數但有一個變數值。這個數值可以運算式方式表示，當 %assign 指令被執行時計值一次。%assign 所定義的變數可以重新定義，如下例：
-
- %assign i 0 ;**指定變數i初值為0**
- %rep 3 ;**迴圈開始(重複三次)**
- DB i ;**宣告一個位元組初值為i之值**
- *%assign i i+1 ;**指定變數i的值為(i+1)***
- %endrep ;**迴圈結束**

多列巨集

- 多列巨集的格式如下：
- **%MACRO 巨集名稱 參數個數**
- **巨集本體**
- **%ENDMACRO**
- 如下例：

```
%MACRO  readchr  1      ;巨集名稱readchr,一個參數
        PUSH  AX        ;疊入AX
        MOV   AH, 01H   ;從鍵盤讀取一個字元
        INT   21H       ;至暫存器AL
        MOV   [%1], AL  ;將AL值存入參數所指位址
        POP   AX        ;疊出AX
%ENDMACRO               ;巨集結束
```

- 上例巨集名稱為 readchr，只有一個參數。首先疊入 AX，表示要將現在的值保存在堆疊的頂端，將 01H 存入 AH 後呼叫 21H 號插斷，表示要從鍵盤讀取一個字元至暫存器 AL，然後將 AL 值存入第一個參數所指位址，最後從堆疊頂端疊出原來的 AX 值後才返回呼叫此巨集的程式。
- 呼叫的程式可作如下的設計：
- ***ch2 DB ' '***
- ***...***
- ***readchr ch2***
- 呼叫 readchr 巨集從鍵盤讀取一個字元至記憶體變數 ch2 處。例如您輸入 'C' 那麼 ch2 變數的值就變為 'C' 了。

9.9 條件組譯

- 條件組譯的格式常用的有下列兩個：
- 格式一：
- ***%ifdef 符號***
- ***只有符號有定義時才被組譯的程式碼***
- ***%endif***
- 格式二：
- ***%if expr***
- ***只有運算式 expr 的值為非零時才被組譯的程式碼***
- ***%endif***

9.10 前置處理迴圈

- NASM 的 TIMES 指令雖然功能強大，但卻不能多次執行多列巨集，因此又提供一個 %rep 前置處理迴圈指引。它以 %rep 開始，以 %endrep 結束，它的語法如下：
- ***%rep 迴圈次數***
- ***迴圈指令***
- ***%endrep***
- 如下例：

```
%assign i 0     ;指定變數i初值為0
%rep 3          ;迴圈開始(重複三次)
   DW i         ;宣告一個字組初值為變數i之值
   %assign i i+1  ;指定變數i的值為(i+1)
%endrep         ;迴圈結束
```

9.11 引入檔指引

- 引入檔指引提供您一個方法將其他的原始程式檔引入至您的程式中，標明引入檔指引之處。其格式如下：
- ***%include　引入檔名***
- 例如您想將 readchr.mac 引入至您的程式中，其指令如下：
-
- ***%include "C:\plone\ch09\mymacro\readchr.mac"***
- ***;您的其他程式指令***

9.12 組合語言指引

- NASM 組合語言的指引只有少數幾個而已。NASM 內定的組合語言指引均以方括號括起來，這裡只介紹常用的兩個組合語言指引 BITS 及 SECTION。
- BITS 組合語言指引的模式宣告如下：
- [BITS 16] ;十六位元模式
- [BITS 32] ;三十二位元模式

組合語言指引SECTION

- SECTION 組合語言指引標示程式碼組譯時所需的段址，對於某些目的檔格式，段的名稱及數目是固定的，對於另外的目的檔格式，段的名稱及數目是要您自己設定的，您若設定不存在的段址就會產生錯誤。對於 及 bin 目的檔案格式而言，NASM 支援三個標準的段名：'.text'、'.data' 以及 '.bss'。.text 是程式碼段，.data 是資料段，.bss 則是沒有初值的資料段。對於 obj 目的檔案格式而言，NASM 並不認識這三個段名，段名必須您自己去設定。SEGMENT 與 SECTION 同義。
- SECTION 組合語言指引的宣告如下：
- [SECTION .text]
- [SECTION .data]
- [SECTION .bbs]

9.13 目的檔格式

- NASM 是一個跨平台的組合語言，組譯後的目的檔可在支援 ANSI C 語言的平台上執行，當然那個平台的處理器必須是 Intel 公司的 80x86 系列的才行。為了達到這個目的，它支援許多的目的檔格式，您可在組譯時使用 -f 選項選定目的檔格式，為了簡單起見本小節只介紹 bin 純二進位輸出格式。
- bin 格式並不產生目的檔，它只產生您程式指令相當的機器碼檔案而已，這種格式用於 MS-DOS 的「.COM」執行檔以及「.SYS」設備驅動程式。bin 格式支援三個標準的段名「.text」、「.data」及「.bss」。輸出的執行檔首先包含「.text」段的指令碼，然後跟隨著「.data」段的資料碼，段與段間以四個位元組對齊，並不包含「.bss」段的資料。若您不標明 SECTION 那麼您的程式碼預設使用「.text」段，若您不標明 BITS 那麼您的程式碼模式預設為十六位元模式。

目的檔格式

- bin 格式額外提供一個 ORG 指引，它告訴 NASM 目的程式執行時的起始位址，對於「.COM」執行檔而言，前面的二百五十六個位元組必須保留給 PSP 程式段，您的程式碼必須從第二百五十六個位元組開始擺放，您可以使用 ORG 指令達到這個目的。
-
- **ORG 0100H ;十六進位數0100相當十進位數256**

9.14 NASM 組譯程式安裝

- NASM 組合語言是一種可攜性高且具模組性的一種組合語言，它的相關軟體可從下列網址免費下載，該網址為：
- ***http://sourceforge.net/projects/nasm***