

WILLIAM H. MURRAY, III AND CHRIS H. PAPPAS

80386/80286

組合語言程式設計

ASSEMBLY LANGUAGE PROGRAMMING

王亮惟 譯

松崗電腦圖書資料有限公司

80386/80286

組合語言程式設計

ASSEMBLY LANGUAGE PROGRAMMING

王亮惟 譯

松崗電腦圖書資料有限公司 印行

松崗電腦圖書資料有限公司已
聘任本律師為常年法律顧問，
如有侵害其著作權或其他權益
者，本律師當依法保障之。

長立國際法律事務所

陳 長 律師



80386/80286 組合語言程式設計
ASSEMBLY LANGUAGE PROGRAMMING

譯 者：王 亮 惟

發行人：朱 小 珍

發行所：松崗電腦圖書資料有限公司

台北市敦化南路五九三號五樓

電 話：(02) 7082125(代表號)

郵政劃撥：0109030-8

印刷者：建發印刷設計公司

中華民國七十六年三月初版

中華民國七十六年七月第二版

本出版社經行政院新聞局核准登記，登記號碼為局版台業字第三一九六號

版權所有



翻印必究

每本定價 270 元整

書號：2101245

譯 序

本書說理清晰、內容廣博，且不殫煩複、力求詳盡，實為介紹性質的好書。筆者有幸，得覽是書，且為之譯，以饗讀者。積數月之功，此書乃成，原書中或有疏誤，筆者悉一一校之，以正其訛誤，補其疏漏。間或有失，皆筆者之罪；倘蒙指正，則甚幸。是為序。

王亮惟

七五年歲末

簡 介

本書設計方針有三大目標：(1)介紹組合語言觀念，包括 80386 / 80286 微處理器及 80387 / 80287 共同處理器等組合語言；(2)指導不同深度的組合語言程式設計；(3)作為指令與程式設計示範的參考書。

80386 / 80286 組合語言脫胎於早期 8088 / 8086 組合語言，而且青出於藍、更勝於藍。由於 Intel 公司的 80xxx 系列都向上相容於 80xx 系列，所以 80xx 系列的程式大都包容於 80xxx 的指令集裏，因此基本的指令與特色大致相同。然而，一旦將來的 DOS 版發展到保護模式等高階特性之後，80xxx 系列與 80xx 系列的差距就顯而易見了。

組合語言通常稱為低階語言，然而低階並非低級。只有學過組合語言之後，才能明瞭如何控制整部電腦。在高階語言階段，程式設計師只能依靠編譯器所提供的特色來設計程式，一旦編譯器功能不全，那就無計可施了。譬如，編譯器中若不提供電玩調整器資訊的取樣函數或程序，那麼就無法用高階語言來設計精美的電動玩具程式了。因此，程式設計師只好利用組合語言來設計一些必須的程序，以彌補高階語言不足的功能。

本書中都採用完整的程式，示範組合語言程式的用法，絕不節省篇幅，敷衍了事。此外，程式之註解亦力求詳盡，俾使讀者能透徹了解，舉一反三。

目 錄

本書程式一覽表

譯 序	I
簡 介	III
第 1 章 組合語言簡介	1
1.1 組合語言的主要優點：速度與控制	2
1.2 80286 / 80386 的族譜	3
1.3 學習本書後的預期效果	5
1.4 學習本書的必備基礎	5
1.5 數字系統	6
1.5.1 二進位數	6
1.5.1.1 二進位數的加法與減法	8
1.5.1.2 位元組	9
1.5.1.3 字元	10
1.5.1.4 有號數	10
1.5.1.5 2' 補數	11
1.6 符號延伸	13

1.6.1	十六進位數	13
1.7	位元組以上的位元組織	16
1.7.1	字組	16
1.7.2	雙字組	17
1.7.3	四字組	18
1.7.4	十字組	19
1.7.5	80386 的資料型態	19
1.7.6	非標準位元欄位	19
1.8	邏輯運算	20
1.9	定址方式	22
1.9.1	立即式定址	23
1.9.2	暫存器定址	24
1.9.3	直接式定址	24
1.9.4	暫存器間接式定址	25
1.9.5	基底式定址	26
1.9.6	直接註標式定址	27
1.9.7	基底註標式定址	28
1.9.8	80386 之延伸	29
1.10	程式設計格式	30
1.10.1	名稱欄	30
1.10.1.1	變數名稱	31
1.10.1.2	標記名稱	31
1.10.1.3	常數名稱	32
1.10.2	運算欄	33
1.10.3	運算元欄	34
1.10.4	註釋欄	35
1.11	組合語言的興起	35
1.12	組合語言範例	35

第 2 章 組譯器簡介	37
2.1 組合語言與機器碼	38
2.2 典型的組合過程	39
2.2.1 第一步：建立原始程式	40
2.2.1.1 程式贅餘	42
2.2.2 第二步：產生目的程式	44
2.2.3 第三步：鏈結	45
第 3 章 80286／80386微處理器的結構：暫存器、旗幟與指令	47
3.1 80286 微處理器	47
3.1.1 基本結構	47
3.1.1.1 區段暫存器	49
3.1.1.2 註標、指標及基底等暫存器	50
3.1.1.3 狀況暨控制暫存器	50
3.1.1.4 指令指標	52
3.1.1.5 機器狀況字組	52
3.2 80386 微處理器	52
3.2.1 資料型態	53
3.2.2 運算元定址方式	53
3.2.2.1 有效位址之計算	53
3.2.3 8086 程式執行	54
3.2.4 基本結構	54
3.2.4.1 多用途暫存器	55
3.2.4.2 區段暫存器	56
3.2.4.3 指令指標與 EFLAGS	57
3.2.4.4 控制暫存器	58
3.2.4.5 系統位址暫存器	59

3.2.4.6	偵錯與測試暫存器	60
3.3	80286 / 80386 指令集	61
3.4	80386 指令集	124
第 4 章	80287 / 80387 數值型共同處理器	133
4.1	80287 / 80387 的運作	133
4.1.1	浮點堆疊	134
4.1.2	狀況字組	135
4.1.3	控制字組	136
4.1.3.1	例外類中斷遮蓋	136
4.1.4	標籤字組	137
4.1.5	例外類中斷指標	138
4.1.6	資料型態	139
4.1.6.1	二進位整數	139
4.1.6.2	聚集十進位數	140
4.1.6.3	短實數、長實數及暫時實數	140
4.1.6.4	特殊數值	141
4.2	80287 / 80387 指令集	141
第 5 章	程式設計基本技巧	181
5.1	算術程式	182
5.1.1	立即式定址的十六進位數加法	183
5.1.2	直接式定址的十六進位數減法	185
5.1.3	直接式定址的多重準度加法	188
5.1.4	註標式定址的多重準度加法	192
5.1.5	暫存器間接式定址的十六位加法	195
5.1.6	用連加法來模擬乘法運算	198
5.1.7	用乘法指令做乘法、平方及立方運算	201

5.1.8	除法運算	205
5.1.9	平方根運算	207
5.2	邏輯運算	210
5.2.1	模擬硬體邏輯閘	210
5.3	查表法	213
5.3.1	查對數表	215
5.3.2	數碼轉換	217
5.3.3	ASCII 轉數換成十六進位數	220
5.4	80386 的 32 位元運算	223
5.5	使用 BIOS 及 DOS 中斷	228
5.5.1	使用 BIOS 中斷清除螢幕	229
5.5.2	使用 BIOS 中斷印標語	235
5.5.3	使用 BIOS 中斷程式資料	237
5.5.4	使用 BIOS 中斷從鍵盤讀入字元	243
5.5.5	使用 BIOS 中斷從鍵盤讀入字串	244
5.5.6	使用 BIOS 中斷查詢時間及日期	247
5.5.7	使用 BIOS 中斷查詢 IBM AT 記憶體大小	250
5.5.8	使用 BIOS 中斷查詢系統裝備	251
5.5.9	使用 BIOS 中斷從列表機印出字串	253
5.5.10	使用 BIOS 中斷在中解像度螢幕描點	255
5.5.11	使用 BIOS 中斷在高解像度螢幕畫線	258
5.6	高級字串指令	259
5.6.1	字串掃描	260
5.6.2	字串搬運	262

第 6 章 組譯器的輔助命令 265

6.1	IBM、MICROSOFT 和 SPEEDWARE 等組譯器之輔助命令	266
-----	-------------------------------------	-----

第 7 章 巨集、程序和庫集	297
7.1 巨集	297
7.1.1 巨集使用法	299
7.1.2 巨集庫	303
7.2 程序	307
7.2.1 程序使用法	308
7.2.2 程序庫	313
7.3 庫集	318
7.4 巨集、程序庫及庫集之比較	319
第 8 章 高等程式設計技巧	323
8.1 彩色螢幕繪圖	324
8.2 讀秒	329
8.3 簡易“菜單”型程式	334
8.4 比較複雜的“菜單”型程式	337
8.5 使用高級字串指令	344
8.6 檔案處理	348
8.7 實際模式與虛擬保護模式程式設計：範例	361
第 9 章 80287/80387 共同處理器程式設計	369
9.1 晶片規格	370
9.2 整數運算	372
9.3 整數列印巨集	377
9.4 整數相乘	379
9.5 開根號	381
9.6 實數運算	385
9.6.1 IEEE 實數格式	386

9.6.2	簡易的實數運算.....	389
9.6.3	IBM MACRO ASSEMBLER 資料轉換常式.....	391
9.6.4	IBM 常式庫使用範例	394
9.6.5	實數角度的正切函數值.....	402
9.6.6	求正弦函數值.....	406
9.6.7	正弦函數值表.....	411
9.6.8	描繪正弦波形.....	415
9.7	利用傅立業 (Fourier) 級數繪圖.....	419
第10章	聯結高階語言.....	435
10.1	STSC 的 APL	436
10.2	BORLAND 的 TURBO PASCAL	441
10.3	MICROSOFT 的 BASIC 編譯器.....	445
10.4	MICROSOFT 的 C 編譯器.....	448
10.5	IBM 的 FORTRAN 編譯器.....	452
10.6	IBM 的 PASCAL 編譯器	456
附錄 A	IBM Macro Assembler	461
附錄 B	MICROSOFT Macro Assembler	477
附錄 C	TURBO EDITASM	493
附錄 D	ASCII 字元表	507
附錄 E	庫集管理器	511
中英名詞對照		519

1

組合語言簡介

對軟體工程師而言，研究組合語言程式設計是最有價值，也是最富有挑戰性的一項工作。只要能隨心所欲地組合各種高階語言及五花八門的硬體設備，你就能駕輕就熟地應用你的機器。但是組合語言程式設計師所面臨的問題也相當的多，舉凡週邊設備的命令與控制、記憶體的管理、速度、程式的效率、資料保密等等，樣樣都要考慮。總之，想要精通組合語言程式設計，就要有耐心和細心。

除非你是在開發軟體的特殊部門工作（像是處理器設計部門之類的），否則組合語言會成為你最強的第二語言。使用組合語言可以直接接觸到暫存器、記憶體，還有處理位元的指令。在高階語言觸摸不到的領域裏，只好用組合語言來解決了。

組合語言程式能產生最快的「執行碼」（executable code），因為組合語言不需要經過「解譯器」（interpreter）的過程（像APL和BASIC），也不需要經過「編譯器」（compiler）過程（像Pascal、PL/I、C和Modula-2）。但是，這也得花相當的代價來換取。

組合語言程式設計師必須要相當細心，因為程式本身控制著微處理器的運作，而且又沒有編譯程式來幫你檢查程式，一不小心，後果難料！組合語言只有一些相當基本的指令，諸如資料搬運、資料比較以及邏輯運算等等。因此同樣是一頁程式的組合語言，如果拿來和高階語言相比，那簡直就是「小巫見大巫」了。也就因為這個緣故，所以一般的大程式都不會用組合語言來設計。（除非有大公司願

意提供金錢與時間來發展一套超高速與超效率的應用程式。)

使用組合語言比高階語言必須具備更多的知識。組合語言程式設計師必須要考慮記憶體的分段問題 (segmenting)，因為組合語言直接控制記憶體的存取，所以有很多細節部分必須注意。例如：程式與資料該擺在記憶體的什麼地方？是否需要「堆疊」(stack)？如果需要堆疊，又該把記憶體的哪個部分劃給堆疊使用？另外，每項資料的大小，如位元組 (byte)、字組 (word)、雙字組 (doubleword)、四字組 (quadword)、十位元組 (tenbyte) 等等，以及資料的型態，如字串、BCD、靜態、動態、有號/無號 (指正負號)、實數、浮點等等，都必須要設計師自己決定。另外，系統裏是否有 80287 / 80387 數值處理器可供使用？記憶體變數是否需轉換成 80287 / 80387 所需的格式？要使用那些暫存器？再者，什麼部分可以用到彩色螢幕顯示器？哪個部分可用到黑白螢幕顯示器？又是哪個部分用到了列表機？諸如此類的問題，只有在設計組合語言程式時，才會遇到。

1.1 組合語言的主要優點

速度與控制

組合語言的一大特色就是速度驚人。一般而言，一個組合語言的「助憶碼」(mnemonic) 對應一個「機器碼」(machine code)，而執行時間只需要幾微秒而已。

BASIC 和 APL 之類的解譯器語言，執行時必須把原始程式逐一地解譯成機器碼，或者是調用很多內建常式 (built-in subroutines) 來完成運算。(這些常式是早已寫好並事先定義以供微處理器使用。) 因為程式每次執行的時候，運算式都必須重新計值並測試是否有錯誤狀況發生，所以速度會變得很慢。

另外，像 Pascal，PL / I 和 FORTRAN 之類的編譯器，會先把原始程式編譯成微處理器可以執行的機器碼。因為編譯的工作只要做一次即可，所以編譯的程式執行起來會比解譯的程式來得快。但是編譯的程式也有缺點，因為編譯器要

應付各式各樣的命令，所以編譯出來的機器碼會有很多累贅的地方。例如，只要在螢幕上印幾個字的 BASIC 程式，其編譯出來的機器碼可能要佔 33K 位元組。如果以組合語言來表示，大概只要三分之一就足夠了，也就是 10K 位元組左右。

組合語言的第二個優點，就是其控制能力。如果沒有組合語言的話，使用者勢必為「罐頭程式」(conned program) 的限制所束縛。例如，一些「罐頭程式」的設計師，可能會禁止使用者使用 CTRL-ALT-DEL 來輸入資料。但是，只要我們用組合語言來寫程式的話，就可以免除這項限制。

使用組合語言的軟體工程師，可以與作業系統溝通，直接控制顯示器、列表機和軟／硬磁碟機的輸出與輸入。而應用程式的設計人員通常也需要和作業系統直接溝通，這些常式一般也都要用組合語言來設計。

1.2 80286/80386的族譜

很多族譜的樹根都有好幾個世紀的歷史。在微處理器設計的範疇裏，計算機族譜樹的成長速率相當可觀。微處理器族譜的樹狀圖，只有十年的歷史，但是以現在的成長速率來看，大約每隔二年就得在樹狀圖中，加上新枝葉了。

第一個 IC 是在 1960 年代初期製造出來的。一個純鈹的 IC 薄片，取代了很多電子原件，並且節省了大量的空間。十年之後，Intel 公司開發完成了第一個 8 位元微處理器晶片，也就是 8008 處理器。在 1974 年，第二代微處理器 8080 也開發完成，8080 是多用途 (general-purpose) 的微處理器。此時，Zilog 的 Z-80 等產品也開始加入競爭日益擴張的微處理器市場。

四年後，第三代的微處理器誕生了。Intel 公司又開發出 8086 微處理器。雖然 8086 向上相容 (upward compatible) 於 8080，但在設計上則比 8080 進步很多，且具有很多特色。Intel 也開發出與 8086 類似的處理器 8088，8088 的設計較為簡單，而且與目前的 I/O 設備相容。在當時的市場上，8088 也是最進步的產品之一。IBM 的第一代個人電腦，就是以 8088 為基礎所設計出來的。

幾乎在同一段時間內，8087 數值型共同處理器 (coprocessor) 也宣告完

成，這個數值處理器是爲了因應高速率、高準度的計算而設計的。

1984 年，Intel 公司在結構設計上又跨出了一大步，再度開發完成了 80286 微處理器。這個更新一代的高性能微處理器，向上相容於 8086 / 8088，而且具有最現代化的特點，包括：記憶體管理、保護機制、任務管理，和虛擬記憶體支援。在一小片 80286 的 VLSI 上，已擁有迷你電腦上的計算與結構的特徵。

由於 80286 的「定址模式」(addressing modes)和基本指令與 8086 / 8088 相同，所以 80286 向上相容於 8086 / 8088，而其暫存器集(register set)的基本結構，也相當適合高階語言程式的編譯。

在資料型態上，80286 提供了幾項很強的型態，諸如字串、BCD 以及浮點等格式。而 80286 完善的定址模式，更能夠很有效率的處理複雜的資料結構，諸如靜／動態陣列(static / dynamic arrays)、資料錄(records)，以及資料錄內的陣列等等。

80286 的記憶體結構更提供了「模組」(modular)程式設計技巧，軟體工程師可以很容易地把記憶體劃分成不同的「區段」(segment)。由於在同一區段中的存取或分岔跳躍等動作的距離較短，所以機器碼也可以縮短。另外，有了「分段」(segmentation)的型式以後，要設計複雜的記憶體管理就易如反掌了。

80286 的實際記憶體空間高達 16 Mega 位元組(共 2^{24} 個位元組)，是由 ROM 及 RAM 所組成。因此 80286 可以同時存放幾個大程式及其相關資料結構，達到高速存取的要求。基於多使用者(multiuser)系統對於記憶體的動態要求，80286 提供 2^{30} 個位元組(1 Giga 位元組)的虛擬位址空間給每位使用者使用。

以前的微處理器大概僅能由四、五個使用者共用，現在的 80286 已經能夠同時應付 3 倍以上的使用者了。若是作為即時(real time)系統使用的話，反應時間可達到從前要求的六分之一之內。

目前 Intel 公司家族樹裏的最新成員是 80386 微處理器。80386 的外部資料匯流排有 32 個位元的寬度，是 80286 的 2 倍，因此能直接定址的範圍就有 4 Giga 個位元組。

1.3 學習本書後的預期效果

1. 對於 80286 / 80386 和 80287 / 80387 的內部結構有深入而紮實的認識。
2. 熟悉指令集與內部暫存器的結構及應用。
3. 明瞭組合語言的優劣點，及 IBM、Microsoft、和 Speedware 等「組譯器」(assembler) 的用途。
4. 掌握「輔助命令」(pseudo-op) 的用法，及「巨集指令」(macros) 和「程序」(procedures) 的設計。
5. 了解高階語言及組合語言的聯結方法。
6. 學會使用機器的繪圖及檔案處理等特性。
7. 了解並使用 80287 和 80387 數值型共同處理器的高速度高準度運算。
8. 明瞭組合語言與 DOS 及 BIOS 常式的介面聯結（指軟體方面的引數傳輸等方式）。
9. 學會使用 IBM 的 DEBUG 和 Microsoft 的 SYMDEB 等偵錯器 (debugger)。

1.4 學習本書的必備基礎

1. 至少要會一項高階語言，像 BASIC 或 PASCAL 都可以。本書中不講授此類基礎。
2. 最好熟悉 16 進位系統 (hexadecimal)，因為了解此系統將有助於了解邏輯運算，像 AND、NAND、OR、NOR、XOR、SHIFT LEFT / RIGHT、ROTATE、和 COMPLEMENT 等等。