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

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

譯　者：王　兆　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

每本定價 270 元整
書號：2101245
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

譯序

本書說理清晰、內容廣博，且不殫煩複、力求詳盡，實爲介紹性質的好書。筆者有幸，得覽是書，且爲之譯，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。是爲序。

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)作爲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。

組合語言通常稱爲低階語言，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

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），否則組合語言會成爲你最強的第二語言。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），因爲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，因爲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。也就因爲這個緣故，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），因爲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。）因爲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。因爲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，因爲編譯器要

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，使用者勢必爲「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操作系統直接溝通，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。一個純鈔的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（upword 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的設計較爲簡單，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爲基礎所設計出來的。

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。若是作爲即時（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的外部資料滙流排有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），因爲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。