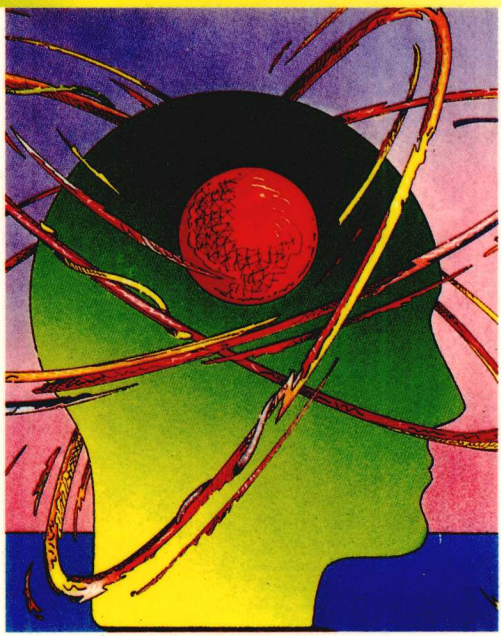
自己設計

SIDEKICK



SoftTip 尖端電腦
瑩圃電腦軟體研究開發部
NO.39360

本書簡介　　書號：39360

開 數： 16開　頁數： 544頁　　譯/著：編譯
適合讀者： 組合語言初學者或有經驗者
所需硬體： IBM PC/ XT或 PC AT
所需軟體： DOS (含 DEBUG、LINK、EXE2BIN), MASM 或 ASM 組譯器
書附磁片： 無

內容概要：

1、 以程式來說明各項功能與指令的用法
2、 凡是第一次出現的指令或功能都加以追蹤或詳解
3、 設計 SIDEKICK 必須用到之程式的技巧 ;如記憶體駐留、鍵盤巨集指令
4、 隨時呼叫之計算器模擬程式之設計

相關書籍：

深入了解 IBM PC
透視 IBM PC XT/AT
IBM PC組合語言應用程式庫
IBM PC DOS & BIOS 實例與學習
IBM PC/MS DOS 技術手冊
IBM PC/MS DOS 系統函數呼叫辭典
MS MASM 5.X 使用手冊
MS MASM 5.X 速查手冊

目 錄

第一章

組合語言的回顧

假如每件事都能符合我們的須要，那本書充其量也不過如坊間給 PC 使用的書一般沒有什麼特色。我們的注意力是集中在發展 PC 上更刺激的能力，並使其為我們所用。我們的主要目的在於能使用電腦裡所有複雜且特殊的功能，以使我們能從泛泛之輩一躍而成為此方面的頂尖高手。

也許你已經 K過有關於 PC 內部的書籍而現正想將這些概念的片斷匯集在一起。也許你很喜歡在滿天雲霧裡探險，以找出 PC 的確實功能在那裡。或許你很懷疑 PC 到底能作些什麼事，或者你可讓它作些什麼事？假如你也想過，如何讓兩個程式在同一時間裡執行；或如何放一個記事本在螢幕上，或如何再取回已刪除的檔案... 等等。那筆者可以肯定的說，本書正是你必須常備在案頭的好書之一。

我們將要與專業的軟體發展者使用相同的工具來建立我們的程式，那就是說，我們將用組合語言。想要讀這本書，你必具備有組合語言的基本知識 -- 就算是不非常熟悉，你也應該有粗略的知識。

本章將針對這個主題作個快速的複習，但是我們的篇幅有限，我們實在沒有辦法說得太過於詳細，所以當你在讀完本章而仍然覺得觀念還

是並不怎麼清楚的時候，筆者以為，你應該多花一點時間與精神澈底的將組合語言弄懂才行。換句話說，如果你已經知道 JGE到底是什麼意思的話，那你可說已經懂得夠多而可提早從此章畢業了。

選擇使用組合語言並不是件奇怪的事情，對於大部份我們將要做的東西，比如能同時讓兩個程式執行，或於某個程式執行時放個計算器於螢幕上，或救回已被刪除的檔案等等，這些只是組合語言所能完成的強大功能之一部份而已。而且我們也會在使用新的東西之前作實際上詳細的討論。總之，學習的較好方式是使用範例，而我們也試著在本書提供些強大而有用的範例供你參考使用。

但是你必須記住一點，本書並不是用來教授組合語言用的。組合語言只是我們拿來作為探測我們所要的東西之工具而已。我們要知道的是在 PC 裡面能使用些什麼有趣的功能之確實知識。也許，能瞭解視訊控制器的原理是件很好的事，但若能實際的應用它不是顯得更有趣味嗎？相同的道理也可以拿來形容磁碟目錄與鍵盤緩衝器。就因為使用 PC 的所有功能是本書的主旨，是故我們不該在未準備週全之前就作深入的討論，因此讓我們花些時間來談談我們所要的工具：

組合語言的心臟

底下是一小群組合成組合語言核心的簡短介紹：

MOV

MOV 是組合語言最基本的指令。假如你已經知道 MOV的確定意義，

那麼可以說你已經上路了。底下是些 MOV的用法：

```
MOV  BX,5CH
MOV  AX, [BX]
MOV  AX,CX
```

第一個命令是將 5CH搬入暫存器 BX 中。第二個命令使用了 BX 的一個厲害功能，它被當作成一個註標 (index)。這個用法是 8088 或相類似的微處理器與上一代微處理器不同的功能之一，而這點對我們來說相當重要。若你想從記憶體裡位址 1234:5678開始的地方取得一個 16 位元的內含值時，那麼 DS 必須設成 1234H而 BX 必須放入 5678H。當BX被放於括弧內時，MOV AX, [BX]會將此字組 (word) 內的值搬入AX之中。因為這種定址的方式只使用在程式裡的資料而不是命令，所以PC將會假定你正在使用資料節段。也就是說，MOV AX, [BX]是 MOV AX,DS:[BX]的縮寫。此種方式稱為間接定址 (indirect addressing)。形成這種定址方式會如此有彈性的原因是，假如持續的增加或減少 BX 的值，你可以向上或向下掃描整個資料節段的資料。如上所述。此種定址的方式非常的有效率，我們將會很快的應用到實例上。

MOV AX, CX這個指令也許你已經知道是作什麼用的了，它只是很簡單的將 CX 裡的數值移到 AX 裡而已，而遺留在 CX 裡的數值並沒有被更改。

INTEL 的 8088 在記憶體裡儲存字 (word) 時，是以一種令人感到無奈的方式完成的。那就是低位元組先存，然後再存高位元組 (也就是說，較高的位元組放於記憶體的較高位址) 。例如，欲存 1234H時，將