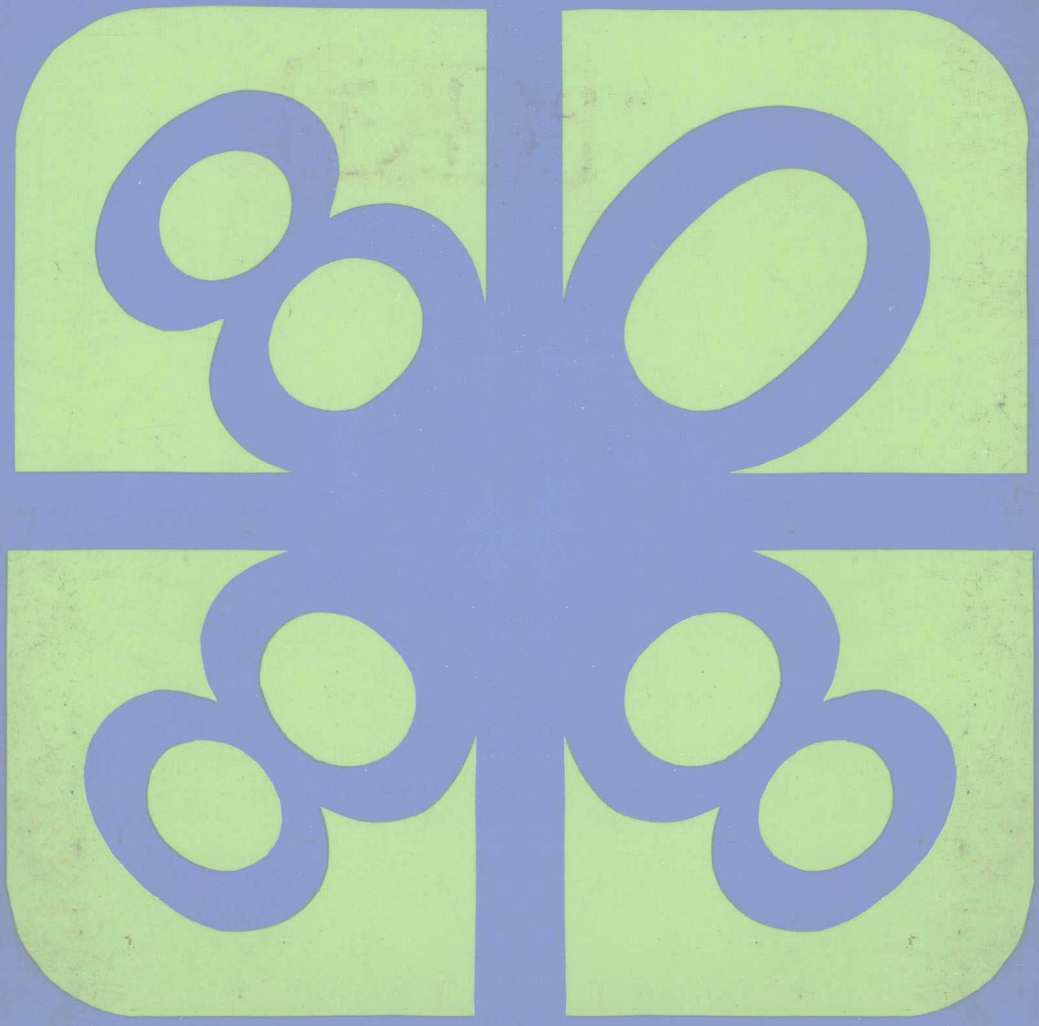
8086組合語言

16位元8086・8088微電腦使用方法

林朝源　譯



8086組合語言

原著發行日期：1983（第一版）
原著書名：THE 8086 Book includes the 8088
原著者：Russell Rector-George Alexy
編譯者：林朝源
發行人：楊鏡秋
出版者：儒林圖書有限公司
地址：台北市重慶南路一段111號
電話：3812302 3110883 3140111
郵政劃撥：106792號
吉豐印刷廠有限公司承印
板橋市三民路二段正隆巷46弄7號

行政院新聞局局版台業字第1492號
中華民國七十三年 月初版
定價新台幣 300 元正

目　錄

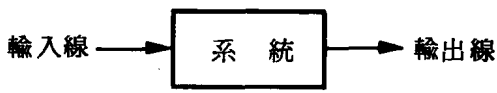
1

程式設計

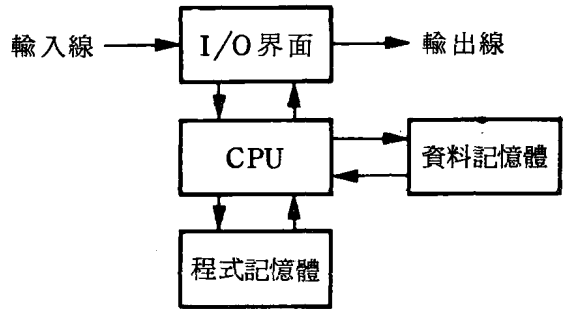
組合語言

組合語言在微電腦系統內有何功能呢？又它與機器語言或較高階語言的程式設計有何不同呢？本章藉著評估組合語言所扮演的各種不同角色來答覆這些問題。

一般而言，微電腦系統均採用下列形式：



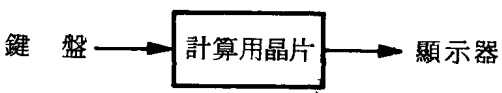
輸入線（input lines）在此是提供訊息給系統，而輸出線（output lines）是用來自系統送出訊息。一般而言，系統包括下列各部份：



中央處理單元（central processing unit；CPU）自輸入線再經過I/O界面（I/O interface）取得資料；CPU藉著執行取自它的程式記憶體（program memory）內之指令來處理此一資料，其處理的結果再經由輸出線輸出；CPU 將暫時性的資料存放在資料記憶體（data memory）內。

CPU、I/O界面和實體記憶體均爲系統的硬體（hardware）部份，而存放在程式記憶體的資料則爲系統或程式的軟體（software）部份。8086 的組合語言程式是由 8086 組合語言的各種元素組合而成的，此程式是經過處理而存放在程式記憶體中；由此可見，組合語言是用來指定位於程式記憶體的程式。

若要瞭解程式的概念，茲以具有下列元件的收銀機（point-of-sale terminal）爲例：



當敲入字鍵時，計算用晶片（calculating chip）將每個鍵擊換成機器可接受的碼（machine-acceptable code），此機器碼可代表一個數或一個計算。計算用晶片藉著詮釋這些機器碼而執行所需的運算並將結果顯示在顯示器（display）上。

計算用之晶片藉著執行一序列的工作來完成這些運算；例如

：計算用晶片將會作下列的工作來決定是否有按鍵發生。

1. 讀取鍵盤狀態位元組。
2. 自狀態位元組取出位元3。
   （若位元3爲0，無按鍵；若位元3爲1，有一按鍵。）
3. 測試位元3。
   （若位元3爲0，則返回步驟1；若位元3爲1，則進行步驟4）。
4. 作下個工作，此工作可能是清除按鍵位元的命令或制止鍵盤的命令。

計算用晶片所作的整組工作（包括所有的移轉和計算運算）即是所謂的演算法（algorithm）。演算法是由有次序且定義完備而具有起點和判定停止的標準的一序列工作所組成的；演算法通常是以上例之形式來表示，即描述工作性質的英文句。可惜CPU不能回應如“讀取鍵盤狀態位元組”之類的英文句，而必須將此英文句所組成的演算法翻譯成 CPU 能夠詮釋的形式；這類形式爲一序列的二進位 CPU 指令。實行演算法的一組 CPU 指令即是所謂的目的程式（object program）。

CPU藉著分析由二進位數元（不爲0即爲1）所組成的訊息單位來執行指令。簡單的 CPU 有兩種週期：指令擷取週期和指令執行週期。在指令擷取週期中，CPU 產生包含下個所要執行指令（訊息單位）的位址，CPU 會向記憶體請求提供該位置的訊息，記憶體即會產生正確的訊息。在接下來的指令執行週期中，CPU對該訊息加以分析而採取適當的行動。

例如：假設在 Intel 8086 系統中有下列的資料以便執行上例所述的工作（位址與指令均爲二進位）。

位 址	指 令
0000	11100100
0001	00001010
0010	00100100
0011	00001000
0100	01110101
0101	11111010

若 8086 從位置 0000 開始執行，它會讀取位於位置 0000 的第一個指令並加以分析。 CPU 斷定此指令爲一輸入指令而且下個位置 0001 包含著所要讀取資料之裝置的位址；由此可知，該裝置碼爲 00001010 。若位於裝置碼 00001010 的裝置產生鍵盤狀態位元組，則執行下個 8086 指令時會將鍵盤狀態位元組讀入 8086 的AL暫存器內。位於 0000 的指令執行過後，下個要執行的指令就是位於 0010 的指令。位於 0010 的指令是利用位於 0011 的訊息與AL暫存器作AND運算。位於 0100 和 0101 的指令判斷位元 3 是 1 或是 0 ，然後採取適當的行動。

CPU 是以 0 與 1 來工作，不過人類對使用 0 與 1 並不習慣；於是在 CPU 和人類之間就必須提供中間的媒介。此一媒介即爲組合語言。人們以組合語言來撰寫程式而不用直接輸入 0 與 1 到電腦。組合語言程式會被所謂的組合程式（assembler）轉譯成相對應之 0 與 1 。以組合語言所寫成的使用者程式就是所謂的原始程式（source program）。

例如：上述以 0 和 1 寫成的程式可用 8086 的組合碼（原始碼）來取代並輸入到組合程式：

```
TOP:    IN      AL,0AH
        AND     AL,08H
        JNZ     TOP
```

組合程式會將此組合碼譯成上例中的 0 與 1 （目的碼）。

組合語言包括一組可被組合程式轉譯成所有可讓系統執行的 0 與 1 組合。

例如：8086 組合語言指令：

```
AND    AL,08H
```

可被組合程式轉譯成兩位元組的目的碼：

```
00100100
00001000
```

8086 則將此目的碼詮釋成：將字組 00001000 的內含與 AL 暫存器內含作 AND 運算的指令。

從許多角度來看，以組合語言來撰寫程式均較以二進碼來撰寫程式來得有效率。其一是：用組合語言指令（如 AND 、ADD 或 XOR ）來寫組合碼顯而易見地較撰寫二進位指令（如 01001000 、10100010 或 01110000 ）來得容易。其二是：用 CPU 語言指令所發生錯誤的機率非常地高；而用組合語言撰寫時，組合程式通常可以找出所犯的任何錯誤。

撰寫程式之工作事項

在此將探討程式師與微電腦系統之間的關係。要讓微電腦系統工作，程式師所作的工作不外乎是：

1. 系統規格之制定：系統規格包括系統所提供之所有功能的一般性討論，以及系統所處理有關輸入和輸出字元的說明。
2. 實行一既定系統之電腦程式設計：這需要將系統規格轉化成一序列步驟以便讓研擬中的系統能應付特殊的應用。
3. 以特殊電腦語言來使程式設計付諸實行：此階段分成三種工作項目：寫碼（coding）、除錯（debugging） 和整合（integration）。
4. 整個系統之測試：將多組測試資料輸入系統以測試程式邏輯和硬體元件。

5. 系統文件之撰寫：合乎要求的文件要將整個系統的工作細節、系統的操作指引、和程式完整的文件加以詳盡地描述。
6. 系統之維護：程式師必須預擬更新系統的計劃以適應新的要求或新的裝備。

在規劃複雜的系統時，上述所列的工作事項往往是不可或缺的。不過有些情形並不需要一份長達250頁的系統規格（包括系統擴展的三個小節）、一份 50 頁的操作指引和嚴格的測試程序。這些程式將會發生在下列情況：

1. 串列通道失靈而硬體設計師和軟體設計師相互推諉責任。幸好，解決此一爭端的方法很簡單，用個簡短的程式來啓動此通道，然後在每次串列 I / O 通道指示有資料時，隨即予以讀取並加以顯示；如此要確定硬體部份是否能正常工作乃是輕而易舉的事；若硬體能正常工作，則此責任的歸屬不言而喻。
2. 系統需作少許的計算而又有 FORTRAN 系統可資運用時，其解決方法爲一個有 20 個敍述的程式即能得到所要的結果。

在以上的兩個情況下，系統規格和程式設計的紙上作業很少而僅僅是記在程式師的腦子裡；於是文件是可有可無，至於是否有維護的需要，更是見仁見智。不過，這些只是少數的例外。

若從事同一個系統的程式師有好幾個，則劃分上列的工作事項可增進效率。有些專門執行步驟 1 和步驟 2，有些只作實行步驟，有些專門作程式系統的測試，另外有一些只撰寫文件或是作系統維護，其餘的人統籌整個工作事項。在如此的安排下，程式師各職所司必能提高生產力。不過，在撰寫組合語言時，全部的工作事項都是由一個組合語言設計師來獨立作業。本書是着重於8086的組合語言，茲在此逐一加以討論。

系統規格之制定

制定微電腦系統初步的構想是來自以下的兩種分析之一：

1. 所面臨的問題是爲解決特殊的問題。例如：太空製造廠商在製造飛彈導向系統時，其計算系統的電路板大小和系統速度必須合乎所求。
2. 爲了替新的微電腦系統開拓特定的市場。例如：以微電腦設計商用系統，其價格降到某種合理價格以下時，原先無力購置電腦來使會計作業電腦化的小型企業都有能力添購。

在以上任何一種情況下，將所構想之系統的眞正功能予以制定是當務之急。在第一種情況下，其問題的本質可能會將系統限制在某些特定的功能。在第二種情況下，則只要依照指定的規格去設計即可。就以小型商用系統而言，我們必須將眞正要做的會計功能及系統所能允許的記錄種類加以定義；要不然微電腦可能被指派過多的工作以致於它的硬體無法勝任。

在此回顧本章先前所提之微電腦系統的簡單模型，其規格定義如下：

- 輸入由此系統接收
- 計算由此系統處置
- 輸出由此系統產生

輸　入

微電腦系統的輸入規格與程式設計的等級息息相關。撰寫BASIC的應用程式設計者並不在乎磁碟控制器之命令型式；他倒很在乎磁碟上的資料型式、磁碟檔案中紀錄的分佈情形以及作

業系統對處理磁碟檔案有何限制等等。由於本書是着重於組合語言的程式設計，因此關於資料庫的處理技巧則超出本書討論的範疇。我們在此所強調的輸入和輸出規格僅限於硬體這個層次。

在硬體層次中，定義輸入通道特性的參數有三種：

1. 資料通路的寬度：控制處理器的誤差系統可能一次只輸入一個位元，而並列或串列的 I/O 通道一次輸入八個位元，而軟性磁碟控制器在收到請求時一次可能傳送 1024 個位元的訊息。
2. 資料傳送速度和型式（同步或非同步），資料可能每隔 200 微秒傳送一次。串列的 I/O 通道可能以非同步方式每隔 10 微秒輸入一次，而在控制系統內的 A/D 轉換器則可能以不定的速率傳送資料，但其速度每次總是較 500 微秒還慢。
3. 附帶之控制訊息。軟性磁碟在有資料時會產生中斷信號，鍵盤系統在有資料時會設定一個狀態位元，而 A/D 轉換器會要求系統去讀取輸入資料並將它與以前的資料比較以決定是否有新資料。

在考慮過這些參數之後，指定輸入通道的製作方式也頗爲重要。

輸入通道通常有三種埠：

1. 資料埠：資料埠包含着傳給系統處理區的資料。
2. 狀態埠：狀態埠所包含的訊息指示何時會有資料進出、指示此通道是否有發生錯誤以及與外界有關的其它訊息。
3. 控制埠：控制埠一般是用來設定通道的操作模式及用來控制通道對外界的表達方式。

這三種埠往往並非一應俱全。有些情況只有資料埠，而有些情況，在電源輸入時，通道會自動地被設定，以致於控制埠並不

需要。

計　算

談到微電腦系統的計算部份時，與其相關的主要有三方面：

1. 處理來自輸入區的原始資料：它可將輸入資料轉換成便於系統使用的碼（如將 ASCII 轉換成二進位），可以將一段資料分解成各個分量（如將來自磁碟的一段資料分解成檔案標題（file header）、標題檢查和（header checksum）、資料和資料檢查和。
2. 系統實際執行的演算法（algorithm）：通常在程式設計內對實際演算法有完整的說明。因此，此一部份的規格應該列出系統所要作的主要功能。
3. 處理資料給輸出區：它將資料轉換成輸出裝置所能使用的形式（如將二進位轉換成 EBCDIC）。

輸　出

微電腦系統輸出規格所需的分析與前述輸入部份所作的分析頗爲相似。每個輸出通道有三種主要的參數：

1. 通道傳送的位元數。
2. 輸出通道的資料傳送速率。
3. 附帶之控制訊息，在發信機還要資料時或者在發信號可以用來處理更多資料時，此控制訊息就會轉告系統。

在考慮過這些參數之後，必須指明如何去控制通道。如同輸入埠一般，輸出通道通常有三種主要埠：

1. 資料埠：資料埠收到要送往外界的資料。
2. 狀態埠：狀態埠所包含的訊息指示何時可以送資料到資料埠、通道是否發生錯誤、以及與外界有關的訊息。