大學用書

微電腦高層語言

杜德煒編著

三民書局印行

微電腦高層語言

杜德煒編著

學歷：國立台灣大學電機系畢業
美國史蒂文生理工學院電機碩士
經歷：電子計算機程式規劃師
美國矽技術公司研究發展部門
微電子計算機系統設計工程師
美國電機電子工程師學會會員

三民書局印行

行政院新聞局局版臺業字第〇二〇〇號

中華民國七十三年三月初版

微電腦高層語言

基本定價肆元肆角肆分

編著者 杜 德 煒

發行人 劉 振 強

出版者 三民書局股份有限公司

印刷所 三民書局股份有限公司

臺北市重慶南路一段六十一號

郵撥：〇〇〇九九八一五號

序

我衷心感謝三民書局劉振強先生的建議，使我們有機會共同努力，為讀者貢獻一份心力。這些心力灌注出的成果就是包括顯現在你面前這本書在內的微電腦叢書。

我深深地感謝國立臺灣大學(57)電機系留美加同學的支持和贊助。他們提供的寶貴意見和豐富資料，使得這套叢書的內容更充實。

我要感謝所有對這套叢書有幫助的人們，特別是馬利蘭大學中國留學生對這套叢書的貢獻；同時，我衷心感謝妻子的體諒和關懷，使我能付出一份專注的心力。

我們把這套叢書分為四部份。第一部份致力於討論微電腦的基本原理和介紹八數元微處理機，單晶片微電腦，十六數元微處理機和三十二數元微處理機。

在美國，八數元微處理機於七十年代使電子工業起了翻天覆地的變化。它們在我國於八十年代中期應開出更好的花，結出更甜的果。單晶片微電腦和十六數元微處理機在八十年代工業發展中將成為主流。而三十二數元微處理機將為切片上安裝完整的大型電腦開闢新的境界。

第二部份主要討論軟體，包括程式語言，程式規劃，軟體設計和操作系統。

第三部份介紹半導體記憶器，介面切片，控制單位和週邊裝置。

第四部份討論介面技巧，系統設計，資料通訊和網路以及各種實際應用等。

這套叢書具有很強的系統性。編著之初，各册的內容和相互關係都已有週詳的考慮和適切的安排。同時，也考慮到讀者閱讀的方便，各册之依賴關係已盡可能地避免。不過，編著過程，錯誤難免，敬請讀者批評指正。

這套叢書各册依完成之先後順序出版。

美國的時代雜誌每年都要選出一位當年的「風雲人物」(man of the year)，五十幾年來從未間斷。可是，它去年選出的却不是「人」，而是「物」。這個中選的「風雲機器」(machine of the year) 就是聲名赫赫的電腦。

以一向密切關注世界政情的時代雜誌之聲譽，破例推選電腦為「風雲人物」，足證電腦對人類和社會影響之一斑。這對我們來說，是一項莫大的鼓舞，鞭策着我們更加地努力。

我們誠懇地希望，這套叢書能夠提供給讀者有用的知識和資料。同時，也希望「風雲機器」電腦能更激勵起大家奮發圖强的雄心壯志。

杜德煒 謹識

中華民國七十二年

微電腦高層語言

目　次

第四章 COBOL 程式規劃語言

第五章 BASIC 程式規劃語言

第六章 PASCAL 程式規劃語言

第七章　PL/M 程式規劃語言

第八章　C 程式規劃語言

第九章　Ada 程式規劃語言

第十章　未來展望

主要參考資料

第一章　緒　　論

第一節　電腦語言的種類

就同我們日常生活中存在著種種不同的語言一樣，電腦也有種種不同的語言。據不完全的統計，存在的電腦語言有兩百多種。不過，也跟我們日常生活中遇到的語言一樣，兩百多種電腦語言中，只有少數幾種是流行的，而絕大多數都是極少使用的。

就跟人類語言可以劃分成三大類：土語、國語和外國語一樣，電腦語言也可以劃分成三大類：機器語言（machine language），組合語言（assembly language）和高層語言（high-level language）。一般說來，本地人都會說本地話，但不一定會說國語和外國語。外地人想同這類本地人交談的話，就需要有人居間翻譯。電腦就跟只會說本地話的人一樣，只懂得機器語言，如果我們想以組合語言或高層語言和電腦通訊，就得請"人"翻譯，這個人通常被稱爲組譯程式（assemblers），直譯程式（interpreters）或編譯程式（compilers），視此程式翻譯的是何種語言和方法如何而定。

一、電腦機器語言

機器語言是電腦的"土"語，是邏輯"1"和邏輯"0"的組合。每一部電腦都由無數條電線組成。這些電線可以承受電壓，接通電流。在正常表示法中，承受高電壓或者接通電流的電線，其邏輯狀態就被定義爲1；而承受低電壓或者沒有電流流通的電線，其邏輯狀態就被定義爲0。此故，電腦懂得由1和0組成的機器語言。

雖然機器語言都由1和0組成，但是每一種電腦都是只懂得自己的機器語言，這就跟不同地方的人都講自己的土語一樣。比如，下面是應用於8080/8085微電腦中的機器語言片斷：

機器語言	代表的意義
00100001	將00裝載入暫存器
00000000	L中，而將02裝載
00000010	入暫存器H中。

這個機器語言片斷如果應用到別的電腦中，就可能有不同的解釋了。機器語言在不同的電腦中有其獨特性，因此機器語言與電腦有極其密切的關係，對程式規劃者來說，如果想用機器語言來規劃程式，他（她）必須對該部電腦有深入的了解。

爲了方便，機器語言除了利用1和0的二進位數碼來表示外，也經常利用八進位數碼或十六進位數碼來表示。

二、電腦組合語言

組合語言也叫做符號語言（symbolic language）。在這種語言程式中，所有操作運算以及指令（instruction）和資料所據記憶器位置的位址都可以用容易記憶的符號來代表。比如前面提到的8080/8085機器語

言片斷

```
0 0 1 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
```

如果以組合語言來表示的話，就變成

```
LXI H, 0 2 0 0H
```

組合語言充分利用了電腦指令集（instruction set）的助憶符號，簡單、易記。以組合語言來規劃的程式必須經由組譯程式翻譯成機器語言，才能爲電腦接受。

儘管我們不把組合語言劃分成不同的種類，但是不同電腦的組合語言還是有很大區別的，就跟不同地方的人講國語一樣，有的甚至差別到完全聽不懂的地步。

組合語言跟電腦也有十分密切的關係。對程式規劃者來說，如果想用組合語言來規劃程式，就必須對該部電腦有深入的了解。

三、高層語言

雖然土語和國語十分重要，但是人類語言多姿多彩的地方仍是因爲存在著很多種不同的外國語。

電腦語言的多姿多彩也在於它的高層語言。不過，電腦高層語言和外國語有一重要的區別。一般說來，外國語不容易學，而電腦高層語言却很容易學，因爲高層語言就是爲著便於使用而設計的。

高層語言雖然有兩百多種，但是流行的却只有幾種，包括 FORTRAN, COBOL, BASIC, PASCAL 等。所有這些語言大致上都可以歸納成

- 一般用途還是特殊用途語言
- 程序規劃還是解題語言

- 直譯還是編譯語言
- 結構還是非結構型語言
- 互作用還是非互作用型語言

下面我們略爲討論這些語言的差異

(一) 一般用途或特殊用途語言

絕大多數的電腦語言都是爲著某種目的而設計的，其中有些後來隨著應用範圍的擴大和語言本身的修改，才成爲一般用途（general purpose）的語言。比如，FORTRAN 語言最初是爲科學工程計算而設計的，COBOL 是爲商業資料處理而設計的，BASIC 是爲了教初學者如何做程式規劃而設計的；它們都已成爲十分流行的一般用途電腦語言。其它的還包括 ALGOL，APL，PL/I，PASCAL 等。

在特殊用途（special purpose）語言中，有的專用來教程式規劃觀念，有些專門用於工業控制或者專門用於繪圖等。這些語言便於使用者用來解決一些具有特殊意義的問題或非普通的應用。比如C，FORTH，MODULA-2 和 SMALLTALK 等語言主要用於規劃系統程式，而 PILOT 用於電腦輔助教學，PROLOG 用於邏輯規劃程式等。

(二) 程序規劃或解題語言

程序規劃語言（procedural programming languages）要求使用者指定一組操作以一定之順序來執行。不過，這種語言不像機器語言或組合語言那樣要求所用的指令必須與某種電腦有密切關係。程式規劃語言的設計是與特定電腦無關係的，也就是說具備一定條件的所有電腦都可以利用這種語言。目前流行的大多數一般用途語言都是程序規劃語言。

程序規劃語言的重點在如何去完成某一工作，而解題語言(problem-oriented languages) 的主要目的是要完成什麼工作。解題語言重視解答問題，而不強調解題的步驟。因此，也就不注重程序。

（三）直譯或編譯語言

所有高層語言程式都必須經過轉譯成機器語言，才能爲電腦所接受。存在著兩種不同的轉譯語言程式，一種叫做直譯程式 (interpreters)，另一種叫做編譯程式 (compilers)。

每當程式運轉時，直譯程式每次轉譯原始程式的一個敍述句，然後執行；執行完後，再去轉譯下一敍述句。編譯程式則一次將整個原始程式完全轉譯成機器語言程式 (卽目的程式)，然後可能將有關聯的程式鏈結在一起，裝存入主記憶器裡去執行；其轉譯成的目的程式可以一再地重新執行，而不必再行轉譯。

一般說來，FORTRAN，COBOL 語言都屬於編譯語言，而 APL 語言則爲直譯語言。除外，BASIC 語言介在兩者之間。

（四）結構或非結構語言

幾乎所有的電腦高層語言都具有一定的結構性。因此，這裡所謂的非結構語言 (unstructured languages) 只是相對於結構語言 (structured languages) 而言。

高層語言的結構就跟外語的文法一樣，如果不了解外語的文法，就很難寫出清楚明白的句子或文章。學習高層語言程式規劃，必須留意該種語言的結構性。

以良好結構電腦語言規劃的程式具有模組性 (modular)。所謂模組，是可以獨立存在，各別編譯和執行，含有完整的邏輯推理的分程式。

以良好結構語言來規劃程式可以花用較少的時間，程式規劃過程較不容易出錯，卽使有了錯也容易改正；而且結構語言程式也較容易維護。

PASCAL 是著名的結構語言，包括 COBOL 在內的多種流行高層語言也正向結構語言方面改進中。

（五）互作用或非互作用語言

在流行的一般用途高層語言中，只有 APL 和 BASIC 語言是爲執行互作用（interactive）操作而設計的。在新語言中，SMALLTALK 和 PILOT 及 LOGO 可以算是眞正互作用語言。

互作用語言使使用者在輸入程式或者運轉程式時如遇有錯誤，可以馬上利用編輯程式來改正錯誤，接著再運轉。反之，以非互作用語言（non interactive languages）來規劃的程式，在運轉時如果有錯誤，就必須放棄運轉；等利用編輯程式來改正後再行運轉。在非互作用語言中，編輯程式和轉譯程式（translator）是各自分開的；而在互作用語言中，上述兩個程式是結合在一起的。

第二節　高層語言的一般功能

雖然高層語言有不同的句型和結構法，但是它們都有共通之處。從純理論的觀點來看，所有高層語言都是一樣的。但是，在實際運用上則不然。因爲一種語言的適用性是由多個因素來決定的，這些因素中有些已超越該語言所提供的邏輯推理的理論可能性之外。

儘管高層語言之間有著差異，但是所有語言都能提供以下的一般功能。

- 資訊表示法
- 修改數值
- 控制結構
- 抽象符號
- 輸入/輸出
- 句型結構

一、資訊表示法

電腦語言的第一個功能是提供資訊表示法 (representative of information)。除了一些微不足道的程式外，所有電腦程式都需要出入一些數值。這些數值可能是有固定值的常數，但在一般情形中，變數 (variables) 是需要的。變數值在程式執行期間是可以改變的。不管是常數也好，變數也好，電腦必須能够參考程式中的這些數值，並安排儲存位置給它們。

變數可以經由宣告明確地應用到程式中，一如 PASCAL 語言程式中那樣；或者經由程式規劃師的使用內在地應用到程式中，就如在 APL 語言程式中一樣；變數也可以經由上述兩種方法的組合應用到程式中，就如在 FORTRAN 語言和 PL/I 語言程式那樣。

除了個別的變數外，電腦經常把多個變數結合起來成爲群 (groups) 或結構 (structures)，而以一個共用的名字來參考這些組合的變數。不同的語言以多種不同的方法來處理變數的結合，比如 FORTRAN 語言利用行列 (arrays)，PASCAL 語言利用組合 (sets)，COBOL 語言利用記錄 (records) 等。

不同的語言是設計來處理不同種類的變數。比如 FORTRAN 主要處理科學工程數字計算。因此，它在代表和處理數字量方面提供相當龐大的功能；又如 SNOBOL 語言主要處理字元字串 (character strings)，故具有代表和處理字串值的良好功能。當然，FORTRAN 語言也可以代表和處理字串值，而 SNOBOL 也可以執行數字計算，但是在效率上就有很大的差別。

二、修改數值

因爲變數在程式執行期間可能改變它的值，所以電腦語言能否指定數值給變數及由舊的數值產生新的數值是相當重要的。在大多數流行電腦語言中，這個能力通常以指定敍述句（assignment statement）的形式來執行。一個表示式依著規定的法則來求值，並決定指定給變數的值。一般說來，流行高層語言都存在著一組依變數來執行的運算。這些運算可與變數和常數一起使用來產生複雜的表示式。

比如有一種電腦語言可以假定變數爲整數值，那麼加、減、乘、除運算就可以整數變數和整數常數來組成含有整數值的表示式。與此相類似，相關運算子如邏輯［及］可用來產生邏輯［眞］或［假］值。藉這種分法，將已存在的數值和變數組合成表示式來計算新的數值是可能的。

三、控制結構

電腦語言也必須提供決定程式中敍述句執行次序的基本架構，這通常叫做控制結構（control structure）。在大多數高層語言中，敍述句可被劃分爲兩大類：宣告性的敍述句和可執行的敍述句。

宣告敍述句主要提供有關變數的資訊，以便轉譯程式利用來分配儲存空間，宣告句也定義輸入/輸出的格式和提供其他類似的功能。可執行敍述句主要指導電腦的計算步驟和邏輯運算。

在高層語言中存在著很多不同種類的控制結構。最通用的結構是決策和順序型，其中敍述句的執行順序都已有良好的定義。這些敍述句依著次序執行，除非遇到如 GOTO 等改變執行順序的特別敍述句，我們可以把單一程式當做代表含有單一開始和單一終止的單一計算。實際

上，所有高層語言都擁有如此的結構。最通用的控制結構允許敍述句的順序和條件執行以及一組敍述句的重覆執行。

不過，控制結構仍有極廣泛的範圍。有些實際語言採用非決策型控制結構，其中，一組敍述句的執行次序是由雜亂處理來決定的。

在 Ada 和很多系統程式規劃語言中，一個程式可以起動另一個程式平行執行。讓兩個程式互相通訊，並且同流執行（execute concurrently）是可能的。平行處理（parallel processing），共常式（co-routines）和內處理通訊（interprocessing communication）的這些特性在發展操作系統和其他控制程式中是十分有價值的。

四、抽象符號

對程式規劃者來說，高層語言比機器語言或組合語言更好使用的理由之一是高層語言提供更自然的符號或記號（notations），也就是說高層語言所提供的符號更能支持程式規劃者思考解決問題的方法。

因爲很多計算程序是重覆地執行的，所以收集經常使用的敍述句並給予一個名稱以在需要時就可使用，是十分方便的。若能如此，只要當程序的名字被涉及的時候，該收集成組的敍述句就可被執行。如果我們設計一個程序來求一組數字的最大值，那麼我們可以命名這個求最大值的順序步驟（程序）爲 MAX。這樣 MAX(S) 的使用就可以在一組數字 S 中找出最大值來。

事實上，所有流行的高層語言都具有這樣的功能。不過，由於語言之不同，這些抽象符號的叫法也就不一樣。比如，有的叫做程序，有的叫做分程式，有的叫做函數，而有的甚至叫做巨碼（macro）。

除了命名計算程序功能外，有些新語言，如 Ada, CLU, Alphard, Euclid 和 PLAIN 等，還引進了抽象資料的觀念（notion of data