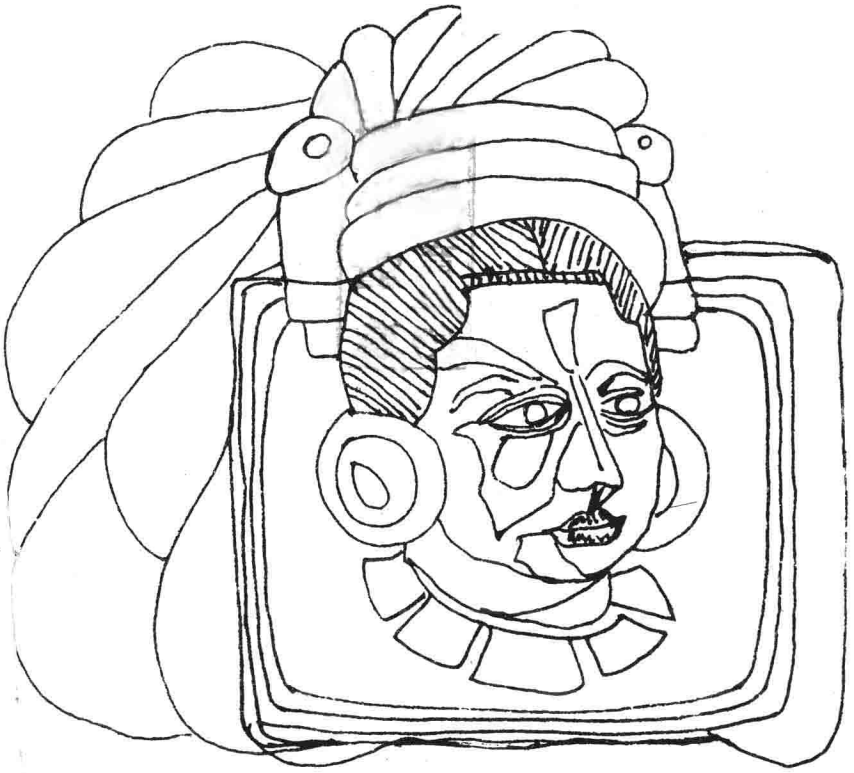
APPLE II

6502组合语言程序

大專用書

6502組合語言程式

ASSEMBLY LANGUAGE PROGRAMMING



北方电脑公司信息资料部

序

自從70年代初期 Intel 公司推出 8008 以來，雖然只有短短的十多年，微處理機的發展，可謂一日千里，各家產品推陳出新，不管是速度、容量、位元長度或功能上都有極大的改進，足以和傳統的小型計算機相比擬，但價格却低很多，而且還在逐年降低中，因此大量地運用在儀表、控制等用途上。尤以最近家用電腦的風行，可以預測的是，最後將如一般家電用品進入每一個家庭。

現在大部分的家用電腦或個人電腦都採用寫起來比較簡單的高階語言如BASIC、FORTRAN 等來寫程式，但對一個程式設計者而言，必須具備有組合語言程式規劃的基礎，對整個設計系統才有整體的概念。尤其對於大部分微處理機的應用並不適於採用高階語言（例如監視一個警告系統或控制一部印出機很難以簡單的代數公式表示）。

微處理機的運用，主要在於軟體程式的巧妙變化。本書所談的就是針對 6502 組合語言程式的規劃（當然只要組合語言變一下，就可應用在它種微處理機上）。本書假設讀者已有微處理機硬體的基本概念。書內第一章是概略介紹，第二章討論組合程式的特性（讀者可先行跳過，等較能適應後再回頭來看）。第三章定義 6502 指令集內的各指令，並且將其與 6800 指令集相比較。第四至第十二章的重點在於用組合語言來寫一些常見的程式，並加以討論。第十三章討論問題定義與程式設計的步驟。第十四章討論程式除錯與測試的方法步驟。第十五章討論加說明書及維護與重新設計的步驟。第十六章把所有步驟集合起來，舉兩個系統性的實例加以說明。

本書的出版，盼能對初學者或從事研究者有所幫助。唯譯者才疏學淺，恐有錯誤疏漏之處，敬請各位先進惠予指教。

6502組合語言程式／目次

第一章　組合語言程式規劃的介紹

INTRODUCTION TO ASSEMBLY LANGUAGE PROGRAMMING

本書所談的是組合語言程式規劃（assembly language programming）。本書假設你已熟悉 An Introduction To Microcomputers：Volume 1—Basic Concepts一書（特別是第6和7章）。本書內不討論計算機、微處理機、定址方法，或指令集的一般特性；你可以參考 An Introduction To Microcomputers：Volume 1 一書中有關的資料。

指令的意義（THE MEANING OF INSTRUCTIONS）

微處理機的指令集是二進位輸入的集合，這些二進位輸入在一個指令週期能產生所定義的動作。微處理機的指令集就像一個邏輯裝置（如一個閘（gate）、加法器或移位暫存器）的功能表；當然，處理機對指令輸入所得的反應遠比組合邏輯裝置對它們的輸入所顯示的反應更加複雜。

BINARY INSTRUCTIONS	二進位指令

指令只是一種二進位元的圖型，爲了利於翻譯成一個指令，指令所代表的二進位數據必須在適當的時間輸入微處理機中。例如6502 微處理機在指令提取作業期間接受8位元的二進位圖型（pattern）11101000當作輸入時，該圖型的意義爲：

"增量（加1至）暫存器X的內容"

同樣地，圖型 10101001的意義爲：

"把程式記憶體的下一個字組（word）的內容載入累積器（Accumulator）中"

微處理機（和其他的計算機一樣）只認識二進位圖型的指令或數據；它不認識字組或8進位，十進位，或十六進位數目。

計算機程式（A COMPUTER PROGRAM）

COMRUTER PROGRAM	計算機程式

事實上；計算機程式不只包括指令；它也包含數據和記憶體位址，微處理機必須使用這些來完成指令所定義的工作。顯然，如果微處理機執行加法時，它必須有兩個相加的數和一個結果的目的地。計算機程式除了執行所特定的運算之外還要決定數據的來源和結果的目的地。

除了遇到一個改變執行順序或停止計算機的指令之外，所有微處理機都能順序地執行指令（即，除了現有的指令特別命令它作其他的指令之外，處理機從下一個連續的記憶體位址中取得下一個指令）。

每一個程式最後翻譯成一個二進位數的集合。例如，下列的 6502 程式是把記憶體位置 0060_{16} 和 0061_{16} 的內容相加，並且把結果存於記憶體位置 0062_{16} 中：

```
10100101
01100000
01100101
01100001
10000101
01100010
```

這是一種機器語言或稱目的程式。如果這個程式放入 6502 爲基礎計算機的記憶體中，微計算機就可以直接執行它。

OBJECT PROGRAM	目的程式
MACHINE LANGUAGE PROGRAM	機器語言程式

程式規劃的問題 （THE PROGRAMMING PROBLEM）

直接寫目的程式或二進位的機器語言程式，會遇到許多困難。可能遇到的難題如下：

(1)這種程式難以瞭解或除錯（尤其在你連續看幾個小時之後，二進位數看來好像相同。）

(2)程式的輸入很慢；由於你必須分別輸入每一個位元。

(3)程式無法說明計算機所執行的工作，因爲程式中的格式（文字）並非人類能力所能讀出。

(4)程式太長，寫出來麻煩。

(5)程式師常因粗心而造成錯誤，而且這種錯誤很難找出。

例如，下面加法目的程式中包含一個錯誤的位元，請試找看看：

```
10100101
01100000
01110101
01100001
10000101
01100010
```

雖然計算機處理起二進位數很簡單，但如由人們來作可沒那麼容易了。有人認爲二進位程式很長、枯燥無味、容易混淆、而且沒有什麼意義。也許，時間久了以後，程式師可以記下一些二進位碼，但爲何不把這些努力花在更有建設性的地方呢？

使用八進位或十六進位
（USING OCTAL OR HEXADECIMAL）

OCTAL OR HEXADECIMAL	八進位或十六進位

如果我們使用八進位或十六進位來寫指令，不要使用二進位數，便可改善以上的問題。在這本書中，我們使用十六進位數，因爲十六進位數較短，而且它們對微處理機工業而言是一種標準表示法。表 1－1 中定義十六進位數字和它們所對應的二進位數。現在，可以把兩數相加的 6502 程式變爲：

```
A5
60
65
61
85
62
```

至少，十六進位寫法寫起來較短，而且比較容易檢查。

在一系列的十六進位數中，很容易就可找出錯誤。上面所例出的錯誤的加法程式，如以十六進位數字表示時，應爲：

```
A5
60
75
61
85
62
```

在這程式中，錯誤很容易看出來。

由於微處理機只認識二進位的指令碼。對這種十六進位的程式我們該怎樣做呢？答案是我們必須把十六進位數轉換爲二進位數。這種轉換是一種重覆、乏味的工作。這種工作由人類來作，會有許多因疏忽所造成的錯誤；例如，看錯行、漏了一個位元、或顛倒了一個位元或一個數字。但是