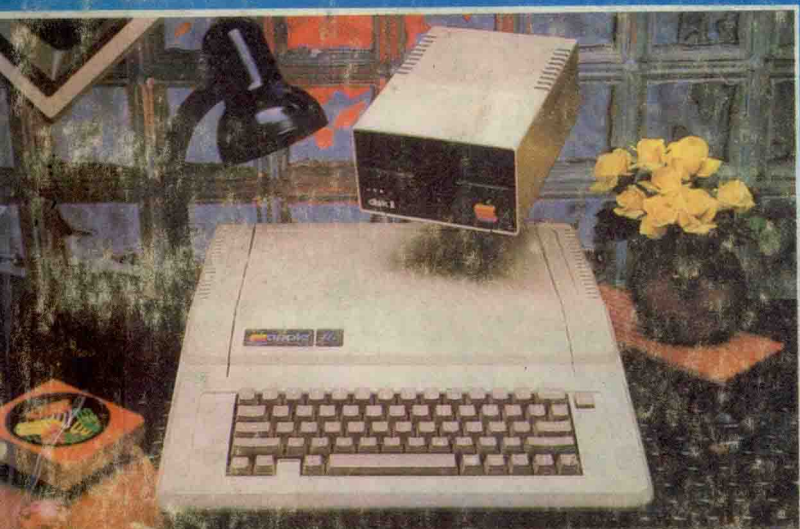


APPLE II

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 組合程式規劃的介紹

(Introduction to Assembly Language Programming) 1

指令的意義	1
計算機程式	1
程式規劃的問題	2
使用八進位或十六進位	3
指令碼的縮寫名稱	4
組合程式	5
組合程式的其他特性	6
組合語言的缺點	7
高階語言	8
高階語言的優點	9
高階語言的缺點	9
微處理機的高階語言	11
你應使用那一種階層的語言 ?	13
未來將會怎樣 ?	14
為何要寫這本書 ?	14

第二章 組合程式 (Assemblers) 15

組合程式的特性	15
組合程式的指令	15
標記	17
組合程式的運算碼 (縮寫名稱)	18
假運算	19
DATA 假運算	20

EQUATE (或 DEFINE) 假運算	21
ORIGIN 假運算	22
RESERVE 假運算	23
LINKING 假運算	24
HOUSEKEEPING 假運算	25
標記與假運算	25
位址與運算元欄	26
條件性的組合	28
巨指令	29
註解	30
組合程式的類型	31
錯誤的訊息	32
載入程式	33

第三章 6502的組合語言指令集

(The 6502 Assembly Language Instruction Set) 34

CPU 的暫存器與狀態旗號	35
6502 的記憶體定址模式	38
記憶體一立即	39
記憶體一直接	39
隱含或固有的定址	40
累積器定址	41
先指標間接定址	41
後指標間接定址	41
指標定址	42
間接定址	44
相對定址	45
6502 的指令集	45
簡稱	46
指令的縮寫名稱	48
指令的目的碼	48
指令的執行時間	48

狀態.....	62
ADC—將記憶體與進位加入累積器中	68
AND—將記憶體與累積器作 AND 運算	69
ASL—將累積器或記憶體位元組向左移位.....	71
BCC—如果進位清除 ($C = 0$)，執行分支	73
BCS—如果進位置定 ($C = 1$)，執行分支.....	73
BEQ—如果等於 0 ($Z = 1$)，執行分支	74
BIT—一位元的測試.....	74
BMI—如果負 ($S = 1$)，執行分支	76
BNE—如果不等於零 ($Z = 0$)，執行分支.....	76
BPL—如果正 ($S = 0$)，執行分支	77
BRK—強制切斷	77
BVC—如果溢位清除 ($V = 0$)，執行分支	79
BVS—如果溢位置定 ($V = 1$)，執行分支	79
CLC—清除進位	80
CLD—清除十進位模式	80
CLI—清除中斷遮罩 (啟動中斷要求)	81
CLV—清除溢位.....	81
CMP—比較記憶體與累積器	82
CPX—比較指標暫存器 X 與記憶體.....	83
CPY—比較指標暫存器 Y 與記憶體	84
DEC—減量記憶體 (減 1).....	85
DEX—減量指標暫存器 X (減 1).....	86
DEY—減量指標暫存器 Y (減 1).....	87
EOR—將累積器與記憶體作不相容或閘運算	88
INC—增量記憶體 (增 1).....	89
INX—增量指標暫存器 X (增 1).....	90
INY—增量指標暫存器 Y (增 1).....	91
JMP—藉絕對或間接定址，執行跳越	92
JSR—跳到副常式.....	93
LDA—將記憶體載入累積器中	94
LDX—將記憶體載入指標暫存器 X 中.....	95
LDY—將記憶體載入指標暫存器 Y 中	96

LSR—將累積器或記憶體的內容向右邏輯移位	98
NOP—不動作	100
ORA—將記憶體與累積器作邏輯的OR運算	100
PHA—放累積器在堆疊上	102
PHP—放狀態暫存器(P)在堆疊上	102
PLA—從堆疊中取出累積器的內容	103
PLP—從堆疊中取出狀態暫存器(P)的內容	104
ROL—將累積器或記憶體經由進位向左旋轉移位	105
ROR—將累積器或記憶體經由進位向右旋轉移位	107
RTI—從中斷回復	108
RTS—從副常式回復	109
SBC—從累積器中減去記憶體與借位	110
SEC—置定進位	112
SED—置定十進位模式	113
SEI—置定中斷遮罩(制止中斷)	113
STA—將累積器儲存在記憶體中	114
STX—將指標暫存器X儲存在記憶體中	115
STY—將指標暫存器Y儲存在記憶體中	116
TAX—將累積器的內容移至指標暫存器X中	117
TAY—將累積器的內容移到指標暫存器Y中	118
TSX—將堆疊指引器的內容移到指標暫存器X中	119
TXA—將指標暫存器X的內容移到累積器中	119
TXS—將指標暫存器X的內容移到堆疊指引器中	120
TYA—將指標暫存器Y的內容移到累積器中	121
6800 / 6502 的通用性	122
MOS TECHNOLOGY 6502 組合程式的規例	126
組合程式欄的結構	126
標記	126
假運算	126
標記與假運算	128
位址	129
其他的組合程式特性	130
第四章 簡單的程式(Simple Programs)	131

例題的一般格式.....	131
習題的要點.....	132
程式例題.....	133
8 位元數據的轉移.....	133
8 位元加法.....	134
向左移位一位元.....	135
遮除最大有效 4 個位元.....	136
清除一個記憶體位置.....	137
字組的分解.....	137
找出兩數之較大者.....	139
16 位元的加法	140
平方值表	141
1s 補數	144
習題	145

第五章 簡單的程式迴路(Simple Program Loops).....149

例題	151
數據的和.....	151
數據的 16 位元和	155
負元素的數目	158
找出最大者.....	160
調整一個二進位分數.....	163
後指標(間接)定址法.....	166
先指標(間接)定址法.....	167
習題.....	168

第六章 字元碼數據(Character-Coded Data).....171

例題.....	172
一串字元的長度.....	172
找出第一個非空白的字元	175
用空白代替前導零.....	178
加偶同位至 ASCII 字元	180
圖型配合.....	183

習題.....	185
第七章 碼的轉換(Code Conversion).....	189
例題.....	189
十六進位轉換為ASCII.....	189
十進位轉換為七段碼.....	192
ASCII轉換為十進位.....	194
BCD轉換為二進位.....	197
二進位數轉換為ASCII串列.....	198
習題.....	200
第八章 算術問題(Arithmetic Problems).....	203
例題.....	203
複準度加法.....	203
十進位的加法.....	206
8位元二進位的乘法.....	209
8位元二進位的除法.....	213
自身核對數—利用 Double Add Double Mod 10 技術 ...	219
習題.....	224
第九章 表與序列(Tables and Lists).....	228
例題.....	228
把單元加入序列中.....	228
檢查一個按次序排列的序列.....	231
從一排列中移動元素.....	233
8位元的分類.....	236
使用一個按次序排列的跳越表.....	240
習題.....	242
第十章 副常式(Subroutines).....	246
副常式說明書.....	248
例題.....	248
16進位轉換為ASCII.....	249

一串字元的長度.....	252
最大值.....	256
圖型匹配.....	260
複準度加法.....	264
習題.....	267
第十一章 輸入／輸出(Input / Output).....	271
定時的間隔(延遲).....	278
延遲常式.....	279
延遲程式.....	280
時間估算.....	281
6502 的輸入／輸出晶片.....	283
6520 週邊裝置介面轉接器.....	283
PIA 的控制暫存器.....	286
PIA 的配置.....	289
PIA 配置的例題.....	290
使用 PIA 來轉移數據.....	293
6522 多用途的介面轉接器(VIA).....	294
VIA 的配置.....	298
CA2 輸入.....	300
CA2 輸出.....	301
VIA 配置的例題.....	302
使用 VIA 來轉移數據.....	305
VIA 的中斷旗號暫存器.....	305
VIA 的定時器.....	306
6522 VIA 定時器 2 的操作.....	307
6522 VIA 定時器 1 的操作.....	308
6530 與 6532 多功能支援裝置.....	310
例題.....	313
一個按鈕開關.....	313
一個雙位置開關.....	318
一個多位置(旋轉式、選擇式、或姆指撥輪式) 開關.....	323
單一個 LED.....	329

七段 LED 顯示器	331
習題	340
更複雜的輸入 / 輸出裝置	342
例題	345
一個未編碼的鍵盤	345
一個編了碼的鍵盤	353
數位至類比轉換器	357
類比至數位轉換器	360
電傳打字機	364
6850 非同步的通訊介面轉接器	372
6551 非同步通訊介面轉接器 (ACIA)	377
邏輯與物理裝置	381
標準的介面電路	383
習題	383

第十二章 中斷處理 (Interrupts).....386

6502 的中斷系統	388
6520 PIA 的中斷處理	390
6522 VIA 的中斷處理	391
6530 與 6532 多功能裝置的中斷處理	394
ACIA 的中斷處理	396
6502 的詢訊中斷系統	396
6502 導向中斷系統	397
例題	398
一個起動中斷	398
一個鍵盤的中斷	401
一部印字機的中斷	405
一個實時時脈的中斷	408
電傳打字機的中斷	416
更通用的服務常式	420
習題	421

第十三章 問題的定義和程式的設計

(Problem Definition and Program Design)

	423
軟體發展的工作	423
各步驟的定義	425
問題的定義	426
輸入的定義	426
輸出的定義	426
處理的部分	427
錯誤的處理	427
人爲因素	428
例題	429
對一個開關的反應	429
開關爲基礎的記憶體載入器	431
核驗終端機	434
問題定義的回顧	438
程式的設計	439
作流程圖	440
例題	442
對一個開關的反應	442
開關爲基礎的記憶體載入器	443
信用卡核驗終端機	444
模組式程式規劃	448
例題	449
對一個開關的反應	449
開關爲基礎的記憶體載入器	450
核驗終端機	450
模組式程式規劃的回顧	452
結構式程式規劃	453
例題	459
對一個開關的反應	459
開關爲基礎的記憶體載入器	459
信用卡核驗終端機	461
結構式程式規劃的回顧	467

由上至下設計	468
例題	470
對一個開關的反應	470
開關為基礎的記憶體載入器	471
核驗終端機	472
由上至下設計的回顧	474
問題定義和問題設計的回顧	475

第十四章 除錯和測試 (Debugging and Testing).....477

簡單的除錯工具	477
較進步的除錯工具	483
用核對表除錯	486
尋找錯誤	487
除錯例題 1：十進位至七段的轉換	491
除錯例題 2：按照漸減的次序分類	495
測試觀念的介紹	503
選擇測試數據	504
測試例題 1：分類程式	505
測試例題 2：自身核對數 (參看第 8 章)	506
測試的預防辦法	506
結論	507

第十五章 列說明書和重新設計

(Documentation and Redesign).....508

自身說明的程式	508
註解	509
加註解例題 1：複準度的加法	511
加註解例題 2：電傳打字機的輸出	514
流程圖當作說明書	516
結構式程式規劃當作說明書	516
記憶體圖	517
參數與定義表	517
庫存常式	519

程式庫例題	520
程式庫例題 1：數據的和	520
程式庫例題 2：十進位至七段的轉換	521
程式庫存例題 3：十進位數之和	522
總說明書	523
重新設計	524
重新組織以使用較少的記憶體	525
大部分的重新組織	527
第十六章 範例設計(Sample Projects)	529
計劃 # 1：數位式計時錶	529
程式解說	531
計劃 # 2：數位式溫度計	544
索引	559

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。

指令只是一種二進位元的圖型，

BINARY	二進位
INSTRUCTIONS	指令

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)

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

COMRUTER PROGRAM

計算機 程式

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 為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，因為程式中的格式（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
01103001
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。有人認為二進位程式很長、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，但為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)

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，

OCTAL OR HEXADECIMAL	八進位或 十六進位
-------------------------	--------------

便可改善以上的問題。在這本書中，我們使用十六進位數，因為十六進位數較短，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 程式變為：

```

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，應為：

```

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為二進位數。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