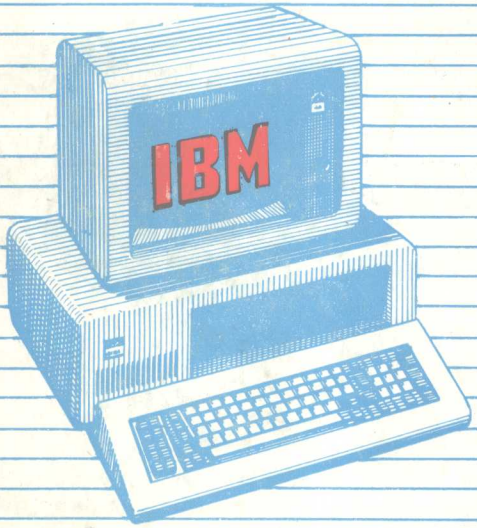
PC



8088 組合語言程式設計

第五册

PC
8088組合語言程式設計

北方电脑公司信息资料部

目　錄

第一章
引言

1981年八月，國際商業機器公司(IBM)正式推出了簡稱爲「IBM PC」的個人電腦，這部電腦本身擁有數種重要的新發明，其中最重要的，便是它採用了Intel公司的8088微處理機作爲其中央處理元件(Central processing element)。

雖然8088在設計上是一個8位元(bits)的晶片(chip)，但是它的結構却跟它的大哥——擁有16位元的8086一樣，這使得PC具有使用大量記憶體的能力。當大部分的微電腦僅能處理64K的記憶容量時，PC却能夠處理高達1024K或1M位元組(byte)的記憶。爲了要利用這項特色，IBM公司便將一個更大而更有威力的培基語言翻譯器(BASIC Language Interpreter)燒製在PC的唯讀記憶體ROM裏面，因此我們不須驚訝大多數的PC使用者，都使用培基語言來編寫他們的程式。

使用培基語言的問題

雖然培基語言易學好用，但它也有某些缺點。培基程式的設計者把PC當作是一個能夠執行培基敍述(statements)和函數(functions)的電腦，然而實際上，PC只能夠執行由其中央處理器——8088所能提供的功能而已，這些機器指令(machine instruction)比起爲我們所熟悉的培基敍述更爲原始；培基的

敍述，例如 PRINT，是由數以百計的 8088 機器指令所構成的。當執行一個培基的程式時，每一個敍述都必須先經過解碼才行，這時一連串適當的機器指令便被用來等效地執行一個培基的敍述，這種過程如圖 1－1 所示，稱爲“邊譯邊執行”（interpretive execution），先天上其執行速度相當慢。

假如我們編寫一個完全由機器指令所組成的程式，那它就可以讓 8088 中央處理器直接地執行，這種程式在執行上會比相同功能的培基程式快上好幾倍？這是因爲我們已經省掉了必須將每個程式敍述解碼的中間過程之故，如圖 1－2 所示。由機器指令所組成的程式就是用組合語言（Assembly Language）所寫的，而用來寫 8088 機器指令程式的語言便叫做「8088 組合語言」。

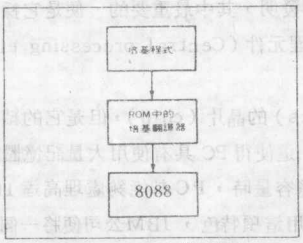


圖 1-1 邊譯邊執行

圖1-2 組合程式之執行

執行速度並非是使用組合語言所能獲得唯一的好處，如前所述，培基程式設計者只把電腦當作是培基電腦而已，結果他們便只能運用由培基語言所提供的功能。可是組合語言程式設計者却能以最低層次的眼光來看電腦，因而能夠利用到電腦的每一項硬體裝置。看見自己所設計的組合語言程式在 PC 上執行是一種相當令人感到滿足的經驗，因爲此時你可以知道你已經完全控制了使機器工作的電路了。

伴隨威力而來的負擔

組合語言的威力是非常強大的，然而在使用它時，連帶的必須付出一些代價

（天下無不勞而獲者）。爲了要高明地利用組合語言來設計程式，你必須十分熟悉電腦內部的組件才行，其中最重要的，便是8088微處理機了。我們將會探討它的內部結構，並且學習它所能執行的指令。

同時，我們也要好好地研究PC系統中的其他組件，例如其中有一個能夠精確地計算事件時間的晶片，它是可以程式規劃的（programmable），我們可以利用所寫的程式來控制它以執行各種不同的計時功能。爲了使它能順利地工作，我們必須瞭解它的內部結構，以及它如何和PC系統的其他組件整合起來。同樣地，還有一些特殊的晶片或電路用來控制揚聲器（speaker），處理中斷(interrupt)，以及維持螢幕顯示器（display）等等。假如我們要控制這些系統的話，我們就必須詳細地了解它們才行。

內容前瞻

以下的章節將會逐次地討論所有上面所提過的課題，使用本書的讀者並不需要具備任何組合語言的經驗，然而他必須熟悉編寫電腦程式的一些觀念，並且最少要了解一種電腦語言，例如培基語言。

第二章將討論二進位及十六進位數字系統，以及組合語言的結構。假如你已經熟悉了其他機器的組合語言，那麼你可以跳過這章（請注意：許多常見的組合語言慣例並不適用於8088）。

有關8088的內部結構將在第3章中加以詳細地說明，其中將討論到暫存器（register），記憶定址方式（memory addressing mode），和此甚具威力的微處理機之指令集（instruction set）。在第4章裏頭我們將學習如何使用IBM的集體指令組合器（Macro Assembler），並且以一個組合語言程式設計者的觀點，仔細地探討IBM的磁碟作業系統（Disk Operating System DOS）。此章也包含了衆多的範例程式，這些範例程式均可以實際地鍵入你的PC而執行，爲此你最少必須具備64K的記憶體，一個磁碟機（disk drive）、單色適應器（monochrome adapter）以及顯示裝置（display）、IBM磁碟作業系統

(DOS)、以及 IBM集體指令組合器程式(Macro Assembler Progrom)。讀者必須要熟悉各種磁碟作業系統的操作，例如磁碟檔案的建立及儲存等，並且假定你已了解EDLIN程式，該程式是用來鍵入及改變程式內容的。

第 5 章說明了PC的系統主機板(system board)，即此個人電腦的心臟。在這章中我們將會學習如何控制這機器的中斷結構，以及鍵盤與計時器的工作原理。第 6 章將討論單色及彩色顯示器接應器(monochrome and color display adapter)，以及印表機接應器(printer adapter)。若要執行彩色顯示的範例程式，當然需具備彩色接應器與彩色顯示器才行。

第 7 章說明了串列式通信(Serial Communication)的觀念，並說明了如何使用非同步通信接應器(Asynchronous Communication Adapter)。能夠和遠方的電子計算機通信，使開啓整個世界之門成爲可能。本章並且提出了一個將PC轉換成數據(資料)終端機(data terminal)的程式，要執行這個程式必須要有一個非同步的通信接應器。

最後，第 8 章談到磁碟的輸入/輸出(Input/Output)，並說明了兩種不同層次的磁碟操作；高層次的磁碟操作使我們能夠處理標準的磁碟檔案(disk file)，而低層次的則使我們能夠直接地存取(access)到磁碟上的磁軌(track)或磁扇(sector)裏頭之資料。

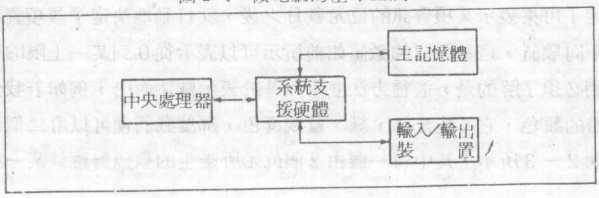
第二章

基本概念

如前所述，一個組合語言程式設計者必須對他的電腦內部組成元件有清晰地了解才行。在這章裏頭，我們將探討所有現代微電腦的基本組成方塊。並對這些機器的工作原理作一深入的了解。

現代的微電腦在邏輯上可以分成 4 個基本部份，如圖 2－1 所示。整個系統的心臟是「中央處理單元」(Central processing unit)，它能夠讀取和執行構成電腦程式的指令；「系統支援硬體」(System support hardware)，則提供了資料流進或流出中央處理器的通路，它並且能執行多種的功能，例如中斷的記錄以及提供用以維持電腦運行的“心跳”等；「主記憶體」(main memory)，用來儲存資訊以備中央處理器隨時讀取；「輸入／輸出裝置」(i/o devices)，則提供了系統與外界通信的界面，鍵盤及螢幕顯示器是最常見的輸入／輸出裝置，而大量資料儲存裝置諸如磁碟機與磁帶機也是屬於這一大類。

圖 2-1　微電腦的基本組成



電子式的資訊表示法

圖 2－1 所描寫的微電腦系統能夠非常快速而又精確地儲存和處理資訊(information)。爲了能在這系統上設計程式，我們必須先了解在這系統中資訊的表示方法。

大家都很習慣於用十進位的數字系統來表示數字資訊，在這系統中資訊的基本單位爲十個十進位數元(digit)，當然它們就是 0，1，2，3，4，5，6，7，8 和 9。在電腦裡頭，資訊的儲存乃是利用電的信號之有和無，因此便只能有二個而非十個數元，我們稱之爲 0 與 1，所有儲存在電腦裡頭的資訊，即由這兩個不同的數字組合而成。

因爲只有 2 個數元，所以我們便稱呼電腦的數字系統爲二進位系統，而以位元(bit，即 binary digit 之縮減字)來表示其資訊儲存的基本單位，當然一個位元只能表示 0 或 1 兩種數字。

現在假如我們要表示超出 0 與 1 範圍的數字時，我們便需要用多於一個的資訊儲存單元或位元才行。如果我們使用二個位元的話，那麼便會有 4 種不同的組合：二個位元均爲 0、二個位元均爲 1、第一個位元爲 0 第二個位元爲 1、以及第一個位元爲 1，第二個位元爲 0。我們可以利用這 4 種組合來表示 0 到 3，假如我們再加上第三個位元，那麼便有 8 種不同的組合而可以表示從 0 到 7 的數字。很明顯地，每多加一個位元，便能加倍所能表示的數字範圍，因此用四個位元便可以表示 16 個不同的數值(圖 2－2)。

決定了用來表示某項資訊的位元數目之後，就自動地決定了該項資訊所可能等於的不同數值，這些不同的數值如前所示可以表示從 0 到某一上限的數字，有一點我們必須了解的是，這種方法並不只限於表示數字而已，例如若我們要儲存一個物體的顏色，它可能是紅、綠、藍或黃色，那麼我們便可以用二個位元來表示，如圖 2－3 所示，其中每一個由 2 個位元所產生的數值對應到某一種顏色。

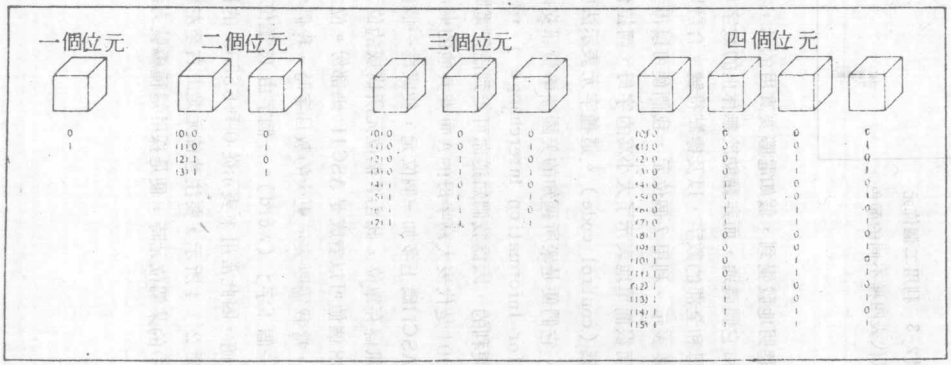


圖 2-2　利用位元以表示數字資料

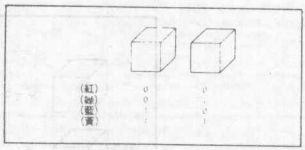


圖 2-3 利用二個位元
以代表四種不同的顏色

以前，有人很聰明地發覺到，假如電腦要實用的話，我們就必須能夠藉助傳統的文字及符號來和它們溝通，而非藉助於無休止的數字串。這便需要有一套方法來表示每一個字母和全部的數字，以及標點符號。合計將近有一百個不同的字元（character）需要表示。利用 7 個位元，我們便可以得到 128 個不同的數值，如表 2－1 所示，而很簡單地表示了大小寫的字母，阿拉伯數字和標點符號，以及各種不同的控制碼（control code）。這種字元表示法已經經過標準化而且廣爲電腦工業所採用。它們便是衆所週知的美國標準資訊交換碼（American Standard Code for Information Interchange），簡稱爲 ASCII 碼，有了這種標準碼是非常便利的，因爲我們知道你可走到任何電腦面前而確信它能夠認出二進位碼 10000001 是代表大寫字母的 A 而非其他的東西。

在 7 個位元的 ASCII 碼上多加一個位元，便成爲一個位元組（byte），這是在電腦中最常用的資訊儲存單位，此由 8 個位元所構成的位元組，可以有 256 個不同數值，前面的 128 個值可以看成是 ASCII 碼部份。位元組也可以用來表示從 0 到 255 的數目字，當我需要表示更大的數目字時，我們常常使用 2 個位元組配對來表示，這便是一個「字」（word）。一個字由 16 個位元所組成，它可以有 65536 個不同的數值，因此常用來表示從 0 到 65535 的數目字。這些不同長短的資訊儲存單位如圖 2－4 所示，要注意位元是由右至左編號的，當我們要參考到一個位元組或字中的某個位元時，便是採用這種編號方法。

表 2 － 1

美國標準資訊交換碼（ASCII）

Binary	Hex	Char	Binary	Hex	Char	Binary	Hex	Char
0000000	00H		0110000	30H	0	1100000	60H	`
0000001	01H		0110001	31H	1	1100001	61H	a
0000010	02H		0110010	32H	2	1100010	62H	b
0000011	03H		0110011	33H	3	1100011	63H	c
0000100	04H		0110100	34H	4	1100100	64H	d
0000101	05H		0110101	35H	5	1100101	65H	e
0000110	06H		0110110	36H	6	1100110	66H	f
0000111	07H	<BELL>	0110111	37H	7	1100111	67H	g
0001000	08H	<BKSP>	0111000	38H	8	1101000	68H	h
0001001	09H	<TAB>	0111001	39H	9	1101001	69H	i
0001010	0AH	<LF>	0111010	3AH	:	1101010	6AH	j
0001011	0BH		0111011	3BH	;	1101011	6BH	k
0001100	0CH		0111100	3CH	<	1101100	6CH	l
0001101	0DH	<CR>	0111101	3DH	=	1101101	6DH	m
0001110	0EH		0111110	3EH	>	1101110	6EH	n
0001111	0FH		0111111	3FH	?	1101111	6FH	o
0010000	10H		1000000	40H	@	1110000	70H	p
0010001	11H		1000001	41H	A	1110001	71H	q
0010010	12H		1000010	42H	B	1110010	72H	r
0010011	13H		1000011	43H	C	1110011	73H	s
0010100	14H		1000100	44H	D	1110100	74H	t
0010101	15H		1000101	45H	E	1110101	75H	u
0010110	16H		1000110	46H	F	1110110	76H	v
0010111	17H		1000111	47H	G	1110111	77H	w
0011000	18H		1001000	48H	H	1111000	78H	x
0011001	19H		1001001	49H	I	1111001	79H	y
0011010	1AH		1001010	4AH	J	1111010	7AH	z
0011011	1BH		1001011	4BH	K	1111011	7BH	{
0011100	1CH		1001100	4CH	L	1111100	7CH	|
0011101	1DH		1001101	4DH	M	1111101	7DH	}
0011110	1EH		1001110	4EH	N	1111110	7EH	~
0011111	1FH		1001111	4FH	O	1111111	7FH	Δ
0100000	20H	space	1010000	50H	P			
0100001	21H	!	1010001	51H	Q			
0100010	22H	"	1010010	52H	R			
0100011	23H	#	1010011	53H	S			
0100100	24H	$	1010100	54H	T			
0100101	25H	%	1010101	55H	U			
0100110	26H	&	1010110	56H	V			
0100111	27H	'	1010111	57H	W			
0101000	28H	(	1011000	58H	X			
0101001	29H	)	1011001	59H	Y			
0101010	2AH	*	1011010	5AH	Z			
0101011	2BH	+	1011011	5BH	[			
0101100	2CH	,	1011100	5CH	\			
0101101	2DH	-	1011101	5DH	]			
0101110	2EH	.	1011110	5EH	^			
0101111	2FH	/	1011111	5FH	_			

註：控制碼以＜＞表示（BKSP＝backspace，TAB＝tabulate，
LF＝line feed，CR＝Carriage return）

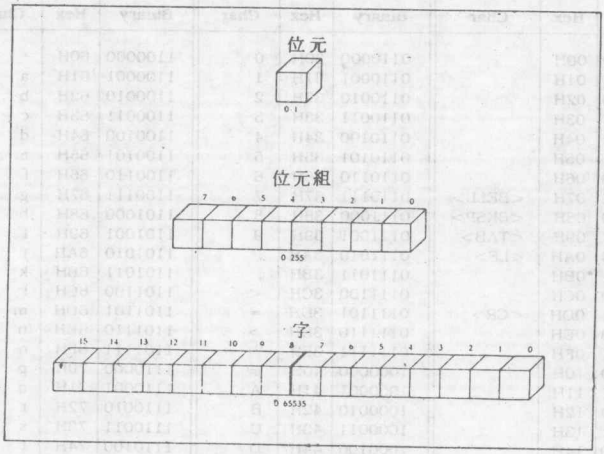


圖 2-4 資訊儲存單位

二進位數字系統是用來定義圖 2－4 中每個位元組合的數值，可能你已經很熟悉我們常用的十進位數字系統，在這系統中總共有 10 個不同的阿拉伯數字，而數目的每個位置代表十的不同乘冪。因此十進位數字 916 表示一個 9 在百位（十的二次方），加上一個 1 在十位（十的一次方），加上一個 6 在個位（十的零次方）。在二進位系統中，只有 2 個不同的數字 0 與 1，所以每一個位置代表二的不同乘冪，所以二進位數字 101 表示一個 1 在四位（二的二次方），加上一個 0 在二位（二的一次方），加上一個 1 在個位（二的零次方） 二進位數 101 在十進位中等於 5，如圖 2－5 所示。

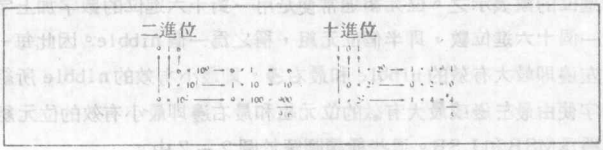


圖 2-5 十進位對二進位數字系統

我們已經看到，假如數目愈大，則用來表示此數目的位元數便愈多，爲了避免陷入使用一長串的 0 與 1 之窘境，我們便利用十六進位的數字系統予以簡化。十六進位表示以十六爲底，因此便有十六個基本數字如圖 2－6所示。我們以熟悉的 0 到 9 十個數字來表示前面的十個十六進位數字，後面還需要六個數字便用最先的六個英文字母代表，因此，十六進位的A表示 10 ，B表示 11 ，依此類推，最後一個 F 表示 15。十六進位數目中每一個位置表示十六的不同乘冪，這些位置由右至左分別表示個位（十六的零次方），十六位（十六的一次方），二百五十六位（十六的平方）等等。我們以附上H字母來表示這是個十六進位的數字，因此 2 F H表示 2 乘上 16 加上 F（15）乘上 1 ，即十進位的 47 。假如一個十六進位數字是以 A 到 F 字母爲開頭的話，我們通常在其前面再冠上一個 0（例如 0A2H，0FFH等等）以避免和以字母爲開頭的標題（label）相混淆。

圖 2-6 十六進位數字系統

現在我們就可以將一個位元組分成前面 4 個位元以及後面 4 個位元，而分別用

一個十六進位的數表示之。位元組通常便是用一對十六進位的數字加上字尾H來書寫的。一個十六進位數，即半個位元組，稱之爲一個 nibble。因此每一個位元組便由最左邊即最大有效的 nibble 和最右邊，即最小有效的 nibble 所組成。同理，一個字便由最左邊或最大有效的位元組和最右邊即最小有效的位元組所組成，分別簡稱爲MSB和LSB。這些稱謂圖解於圖 2－7 中。

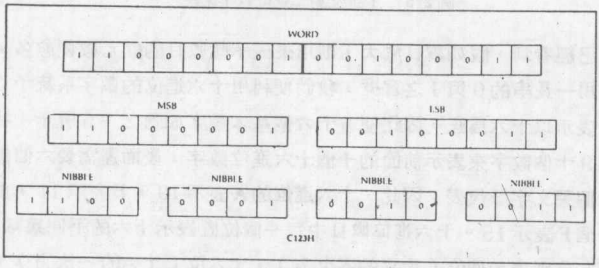


圖 2-7　最大有效和最小有效組成

二進位與十六進位算術

因爲我們將使用二進位及十六進位的數字系統，所以我們必須先了解它們的算術是如何執行的。它們的計算過程和我們所熟悉的十進位系統非常類似。當我們在十進位系統作兩個數的加法時，都是從右至左逐行將同行的數字相加起來，若相加結果超過十進位中最大的基本數字 9，便往前進一位而把相加結果減去10之餘數寫在原位中。在二進位系統裡，如果相加結果大於最大的基本數字 1 便要往前進一位，且把相加結果減去 2 之餘數擺在原來位置。同理在十六進位系統，當和超過了 F H（即 15），便往前進一位，且將和減去 16 之餘數放在原位中。圖 2－8 展示一些計算的例題。

到目前爲止我們只處理了正數而已，要以二進位數來表示負數則需使用二補法（Twos Complement）。在一個二補數中最左（最大有效）的位元用來表示