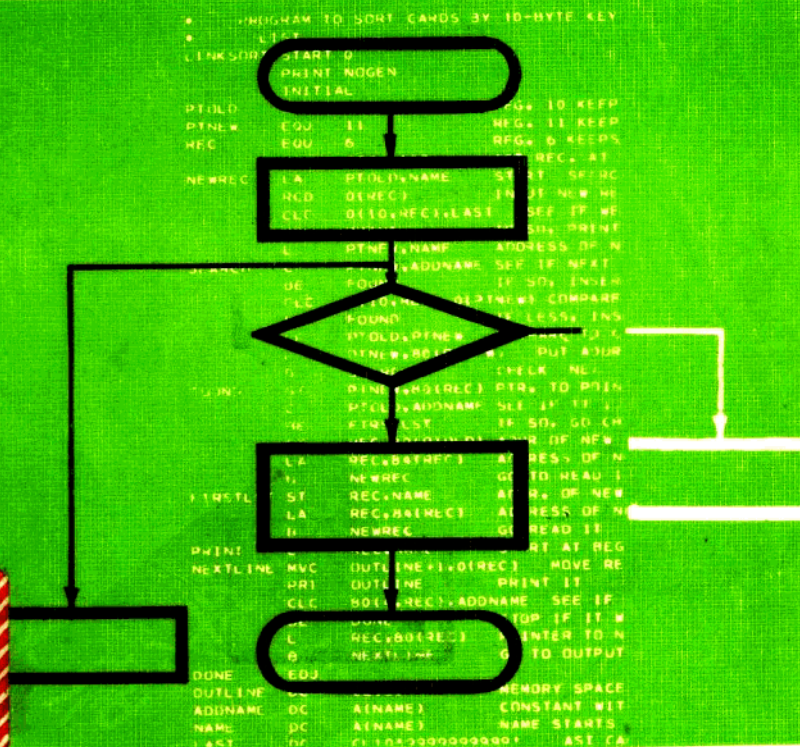


# 系統程式

上册

謝秀芬 編譯



全華科技圖書公司印行

# 序

系統程式是一種原已設計完成的程式，由廠商提供給用戶。計算機用戶不但需了解系統程式，以建立適當的計算機系統，並能藉此改良現有的系統。程式師了解計算機系統之建立及設備後，才能有效的，簡便的使用計算機。

研究系統程式必先了解計算機的原理、結構及計算機語言，因此本書上册主要介紹計算機概論及計算機語言。下册介紹系統程式的原理及範圍。

本書上册適合專科學校一學期的課程。上册重點在第二章機器語言及第三章組合語言。作者簡單的介紹各種指令之使用法，各類指令之後附上多個實例，讀者在研讀實例時，能確切的了解各種指令的用法。

作者才疏學淺，復以倉促付梓，難免有疏漏之處，尚請讀者先進不吝指正。

謝秀芬

中華民國 67 年 5 月

# 圖書之可貴 在其量也在其質

---

量指圖書內容充實、質指資料新穎够水準，我們就是本著這個原則，竭心盡力地為國家科學中文化努力，貢獻給您這一本全是精華的全華圖書。

---

# 目 錄

## 第一章 計算機概論

|                   |    |
|-------------------|----|
| 1-1 計算機結構 .....   | 1  |
| 1-2 資料的代表方式 ..... | 3  |
| 1-3 位址邊界 .....    | 10 |
| 1-4 程式情況暫存器 ..... | 12 |
| 習 題 .....         | 18 |

## 第二章 360或370系統的機器語言

|                       |    |
|-----------------------|----|
| 2-1 指令之構造及運算元位址 ..... | 21 |
| 2-2 基記錄器及指標記錄器 .....  | 22 |
| 2-3 指令之格式 .....       | 25 |
| 習 題 .....             | 29 |

## 第三章 組合語言

|                      |    |
|----------------------|----|
| 3-1 前言 .....         | 31 |
| 3-2 組合語言格式 .....     | 32 |
| 3-3 訂定常數 .....       | 33 |
| 3-4 文字常數 .....       | 36 |
| 3-5 訂位指令 .....       | 37 |
| 3-6 二進位加法及減法指令 ..... | 37 |

|                           |     |
|---------------------------|-----|
| 3-7 二進位乘法 .....           | 39  |
| 3-8 二進位除法 .....           | 41  |
| 3-9 有關資料傳遞的指令 .....       | 42  |
| 3-10 LA指令 .....           | 46  |
| 3-11 變更執行順序的指令 .....      | 48  |
| 3-12 LM及STM指令 .....       | 54  |
| 3-13 USING及DROP虛擬指令 ..... | 55  |
| 3-14 副程式的連結 .....         | 57  |
| 3-15 有關數值型態轉換的指令 .....    | 64  |
| 3-16 虛擬操作指令 .....         | 68  |
| 3-17 比較指令 .....           | 73  |
| 3-18 位移指令 .....           | 78  |
| 3-19 邏輯運算描述 .....         | 84  |
| 3-20 IC及STC指令 .....       | 87  |
| 3-21 邏輯加法和邏輯減法 .....      | 89  |
| 3-22 邏輯比較指令 .....         | 91  |
| 3-23 LTR指令 .....          | 93  |
| 3-24 TM指令 .....           | 94  |
| 3-25 SPM指令 .....          | 95  |
| 3-26 聚集十進位數的運算指令 .....    | 96  |
| 3-27 MVN, MVZ, MVO .....  | 98  |
| 3-28 TR及TRT .....         | 100 |
| 3-29 ED及EDMK指令 .....      | 103 |
| 3-30 EX指令 .....           | 106 |
| 3-31 堆集 (STACK) .....     | 107 |
| 習題 .....                  | 109 |

## 第四章 組合語言之輸入輸出集體指令

|     |                                                   |     |
|-----|---------------------------------------------------|-----|
| 4-1 | 資料組 ( DATA SET )                                  | 117 |
| 4-2 | 資料控制併組 ( DATA CONTROL BLOCK )                     | 119 |
| 4-3 | 資料組的構造                                            | 120 |
| 4-4 | 緩衝區 ( BUFFER )                                    | 121 |
| 4-5 | 列除順序出入方法 ( QUEUE SEQUENTIAL ACCESS METHOD ) 之集體指令 | 122 |
| 4-6 | OPEN 及 CLOSE 集體指令                                 | 123 |
| 4-7 | GET 及 PUT 集體指令                                    | 124 |
| 4-8 | DCB 集體指令                                          | 125 |
| 4-9 | 例子                                                | 129 |
|     | 習 題                                               | 130 |

## 第五章 程式連接集體指令

|     |     |
|-----|-----|
| 習 題 | 136 |
|-----|-----|

## 第六章 任務控制語言

|     |       |     |
|-----|-------|-----|
| 6-1 | 任務卡   | 138 |
| 6-2 | 執行卡   | 139 |
| 6-3 | DD 陳述 | 139 |
|     | 習 題   | 142 |

## 第七章 高階語言

|     |                             |     |
|-----|-----------------------------|-----|
| 7-1 | 常數及變數                       | 146 |
| 7-2 | 陳述的意義及構造                    | 147 |
| 7-3 | 程式之結構及 PROCEDURE 陳述, END 陳述 | 148 |
| 7-4 | 算術陳述                        | 148 |
| 7-5 | 宣告陳述                        | 149 |
| 7-6 | GO TO 陳述                    | 151 |

|                                          |     |
|------------------------------------------|-----|
| 7-7 IF陳述 .....                           | 151 |
| 7-8 DO陳述 .....                           | 152 |
| 7-9 輸入及輸出陳述 .....                        | 153 |
| 7-10 列陣及結構 .....                         | 154 |
| 7-11 PROCEDURE .....                     | 157 |
| 7-12 併組結構 ( BLOCK STRUCTURE ) .....      | 158 |
| 7-13 程式中斷 ( PROGRAM INTERRUPTION ) ..... | 159 |
| 7-14 字元串 ( CHARACTER STRING ) .....      | 159 |
| 7-15 儲存器之使用 .....                        | 160 |
| 習 題 .....                                | 162 |



## 計算機概論

### 1-1 計算機結構

電子計算機 (COMPUTER) 俗稱電腦，爲了方便起見，本書稱之爲計算機。計算機包括三個主要部分：記憶器 (MEMORY)，中央處理機 (CENTRAL PROCESSING UNIT, 簡稱CPU) 及輸入輸出裝備 (I/O DEVICE)。如圖 1-1 所示。記憶器主要功用是儲存計算機

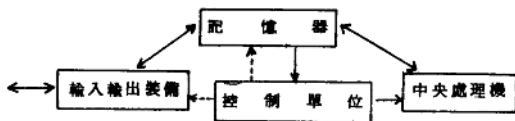


圖 1-1 計算機的基本構造

的指令 (INSTRUCTIONS) 及資料 (DATA)，計算機藉輸入輸出裝備和外界聯絡，假如我們要計算機執行 X 加 Y 之運算，X 和 Y 這兩個資料必須經由輸入裝備以儲存於計算機中，此外還需下達命令以指導計算機完成所要的工作，這種命令就稱之爲指令。指令也必須經輸入裝備以儲存於記憶器中。計算機中真正能進行運算的是中央處理機，中央處理機從記憶

## 2 系統程式

器中取出指令及資料，依指令進行運算，運算結果又儲存於記憶器中，最後經由輸出裝備，把結果送到使用者手中。計算機這一連串的工作必須受計算機控制單位（CONTROL UNIT）的控制。

由以上的敘述可知記憶器是一種儲存單位，無論是指令或資料都必須儲存到記憶器中。記憶器是由許多的磁圈（CORE）構成的，每一個磁圈有正負兩個方向，分別用以表示 1 和 0，稱為位元（BIT），八個位元構成位元組（BYTE）。為了方便起見，記憶器可以取位元組為單位，而加以編號，這些編號就稱為位址（ADDRESS），例如第一個位元組稱為記憶器位址 0，第二個位元組稱為記憶器位址 1，……等。如果記憶器的容量是 128 K 個位元組〔註一〕，那麼記憶器中第一個位元組的位址為 0，第二個位元組的位址為 1，……最後一個位元組的位址是 128K-1。

中央處理機包括指令分析器（INSTRUCTION INTERPRETER）、計位器（LOCATION COUNTER）、指令記錄器（INSTRUCTION REGISTER）、工作記錄器（WORKING REGISTER），以及一般目的記錄器（GENERAL PURPOSE REGISTERS）。指令分析器是由許多的電路（ELECTRICAL CIRCUITS）構成的，它把記憶器中取出的指令加以分析，以確定那一種類的指令。計位器又稱為程式計數器（PROGRAM COUNTER）或指令計數器（INSTRUCTION COUNTER），它指出記憶器中那一條指令正被計算機執行著。記憶器中取出的指令首先被抄到指令記錄器中。工作記錄器有儲存資料的功用，有了指令分析器可以進行快速的資料獲取及儲存，當計算機執行某一個程式時，一般目的記錄器不但可用以儲存資料，同時還可以用它進行運算，360 及 370 系統有十六個一般目的記錄器，分別稱之為記錄器 0, 1, 2, …… 15，每個有三十二位元。另有四個浮點記錄器，分別稱為浮點記錄器 0, 2, 4, 6。

【註】 1024 個位元組稱為 1 K，2048 個位元組稱為 2 K。

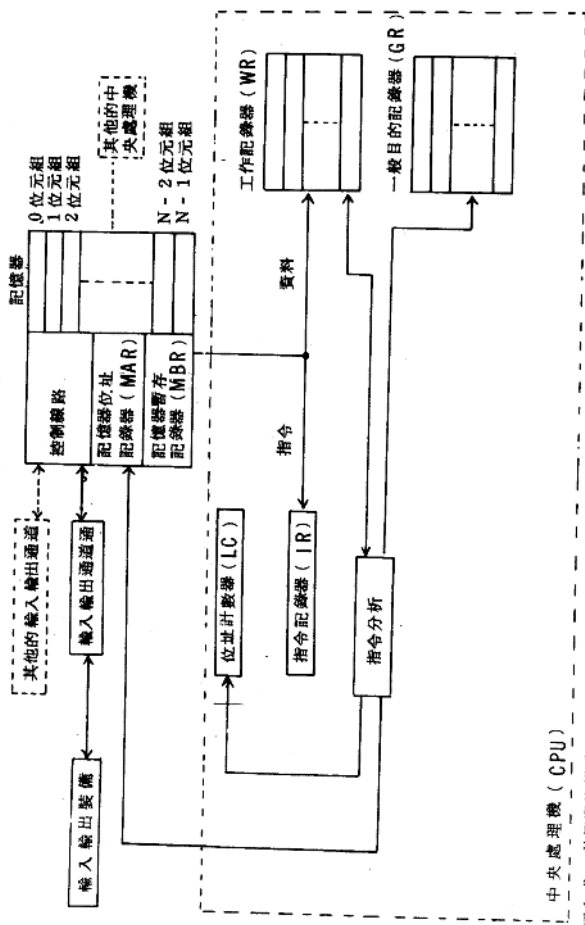
中央處理機和記憶器的主要橋樑是記憶器位址記錄器 (MEMORY ADDRESS REGISTER) 及記憶器暫存記錄器 (MEMORY BUFFER REGISTER)。記憶器位址記錄器中包含記憶器位址, 它指出資料應儲存到那一個位址, 或者從那一個位址取出資料, 不論是把新資料存入記憶器, 或把由資料從記憶器中取出, 這些資料必須先經過記憶器暫存記錄器。而記憶控制則為控制線路, 它控制資料在記憶器及記憶器暫存記錄器之間的傳遞。(見圖 1-2)

輸入輸出通道 (I/O CHANNEL) 可視為一個小型計算機, 它把記憶器中取出的輸入輸出指令加以翻譯後, 並指定某些輸入輸出裝備, 以進行輸入或輸出, 輸入輸出裝備中最常用的是讀卡機 (CARD READER) 及印字機 (PRINTER), 讀卡機主要為輸入用, 常用的卡片長 7.5 英寸, 寬 3.25 英寸, 卡片上有 80 行, 12 列, 因此卡片上形成 960 個位置, 每個位置上都可以打孔, 見圖 1-3。把卡片送入讀卡機之前, 必須先在卡片上打孔, 打孔的方法依照 HOLLERITH 卡片數碼 (HOLLERITH CARD CODE), 該數碼用卡片上孔的組合的表示 26 英文字母, 十個阿拉伯數字及一些特殊的符號像 +, -, \*, /, =, ; ..... 等等。每一行代表一個字元 (CHARACTER), 每張卡片上可打八十個字元。例如 L 是在列 11 及列 3 上各打孔, 數字 3 是在列 3 上打孔, A 在別 12 及列 1 上打孔, \* 是在列 11, 列 4 及列 8 上打孔。

印字機可以把記憶器內儲存的資料印到紙上, 印字機可分兩類: 一為行印字機 (THE LINE PRINTER), 一為字印字機 (THE CHARACTER PRINTER)。前者以行為單位, 一行通常有 132 個字, 132 個字同時印出。後者以字為單位, 每次只印出一個字。

## 1-2 資料的代表方式

計算機記憶器內儲藏的不外乎程式及資料, 所謂程式是許多指令的集合, 這些指令有一定的執行順序, 可以讓計算機依照執行, 資料中有些是



中央處理機 (CPU)

圖 1-2 計算機的構造

|      | L | 3 | A | * |
|------|---|---|---|---|
| 列 12 |   |   | ■ |   |
| 列 11 | ■ |   |   | ■ |
| 列 0  |   |   |   |   |
| 列 1  |   |   | ■ |   |
| 列 2  |   |   |   |   |
| 列 3  | ■ | ■ |   |   |
| 列 4  |   |   |   | ■ |
| 列 5  |   |   |   |   |
| 列 6  |   |   |   |   |
| 列 7  |   |   |   |   |
| 列 8  |   |   |   | ■ |
| 列 9  |   |   |   |   |

圖 1-3 卡片之形式及文字在卡片中的表達形式

數字，有些是字元 (CHARACTER)。數字可以做為運算，而字元多半是用來聯絡計算機及使用者的，使用者的某些信息 (MESSAGE) 必須由字元告訴計算機，計算機也可以藉字元告訴使用者某些消息。

計算機以二進位數 (BINARY NUMBER) 為主，一個位元可用以表示 0 或 1 兩種情況，二進位數在計算機中儲存的方式有兩種，一種稱為定點二進位 (FIXED-POINT BINARY)，一種為浮點二進位 (FLOATING-POINT BINARY)。一個字 (WORD) 可以有十六個位元，或三十二個位元，或六十四個位元，分別稱為半字 (HALF WORD)、全字 (FULL WORD) 及雙字 (DOUBLE WORD)。每一個字中，最左邊的位元稱為位元 0，左邊第二個位元稱為位元 1，……最右邊的位元稱為位元  $N-1$ ，半字時  $N$  等於 16，全字及倍字則分別為 32 及 64，定點二進位數的正負符號出現在位元 0 上，0 表示正號，1 表示負號，位元 1 至位元  $N-1$  則表示該數的大小，見圖 1-4。

尚未詳細介紹定點及浮點二進位之前，我們還要介紹另一種數字系統稱為十六進位數字。包括的數字有 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F 十六個數字， $2^4 = 16$ ，所以必須用四個位元才能表示一個十六進位數字，十六進位數字與二進位數字的關係見圖 1-5。圖中顯示十六進位數與二進位數有密切的關係，(00111100)<sub>2</sub>，與 (3C)<sub>16</sub>。

之意義相同， $(1001011011110001)_2$  與  $(96F1)_{16}$  相同。二進位表示法太長，且容易產生錯誤，十六進位的數字短且明瞭，所以我們常用十六進位數字來表示二進位數字。把二進位數字從最右邊開始，四個位元一取，化成相當的十六進位數，就成為十六進位的表示法了。

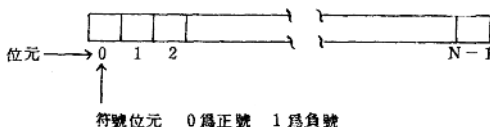


圖 1-4 符號二進位 (SIGN-BINARY) 數表示法

| 十六進位數字 | 二進位數字 | 十六進位數字 | 二進位數字 |
|--------|-------|--------|-------|
| 0      | 0000  | 8      | 1000  |
| 1      | 0001  | 9      | 1001  |
| 2      | 0010  | A      | 1010  |
| 3      | 0011  | B      | 1011  |
| 4      | 0100  | C      | 1100  |
| 5      | 0101  | D      | 1101  |
| 6      | 0110  | E      | 1110  |
| 7      | 0111  | F      | 1111  |

圖 1-5 十六進位數字用四位數之表示法

定點二進位數為正數時非常簡單，位元 0 之內容是 0，表示正數，剩下的位元表示該數之值，例如  $(0000000000000001)_2$  或  $(0001)_{16}$  表示 +1， $(0000000000000101)_2$  或  $(0005)_{16}$  表示 +5。但 -1 及 -5 的表示法却不是  $(1001)_{16}$  及  $(1005)_{16}$ 。360 及 370 系統中，為了計算機運算方便起見，負數是採用 2 的補數型式 (2'S COMPLIMENT)。假設數值用五個位元來表示，那麼位元 0 為符號位元，其他四位元表數值之大小，例如  $(01001)_2$  為 +9 之表示法，如何表示 -9 呢？-9 用 2 的補數表示是  $(10111)_2$ 。其方法如下：

- (1) +9 以二進位表示為  $(01001)_2$ 。
- (2) 把上列二進位數中，0 變成 1，1 變成 0。這稱為 1 的補數 (1'S

COMPLEMENT)，即  $(10110)_2$ 。

(3) 把求得的二進位數中，最右位元加1，得  $(10111)_2$ 。

所以半字-1及-5在計算機內的表示法為  $(1111111111111111D)_2$  及  $(1111111111111011)_2$ 。

計算機中二進位小數點的位置不出現在假數 (MANTISSA) 上，而是隨著指數部分 (EXPONENT) 移動的稱為浮點二進位。見圖 1-6

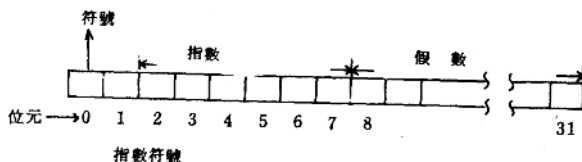


圖 1-6 浮點二進位之表示法

假數本身的小數點設在假數的最左邊，指數部分永遠為整數，浮點二進位數以三十二個位元表示時，位元 0 為該數的正負號，位元 1 為指數的正負號，位元 2 至位元 7 屬於指數部分，位元 8 至位元 31 為假數部分。

指數如為正整數，則假數部分的二進位小數點應往右移，以便得到該數之正確值，如為負整數，則小數點往左移。360 及 370 系統中，指數部分以超 64 數碼 (EXCESS 64) 表示，如果  $N$  為某個十進位數值，以超 64 數碼表示時為  $N+64$ ， $N$  之範圍是  $-2^{7-1} \leq N \leq 2^{7-1}-1$ ，即  $N$  必須大於或等於 -64 且小於或等於 63。 $N$  最小為 -64，加 64 後得到 0，因此不可能有負數的出現。圖 1-6 顯示指數部分如包括符號位元共佔七個位元，從位元 1 至位元 7。這七個位元如為  $(1000000)_2$  表示  $N$  原為 0， $(1000001)_2$  表  $N$  原為 1， $(1111111)_2$  表  $N$  為最大數 63， $(0111111)_2$  表  $N$  原為 -1， $(0111110)_2$  表  $N$  原為 -2，……， $(0000000)_2$  表  $N$  原為最小數 -64。由此可見從  $(0000000)_2$ ，……至  $(1111111)_2$  分別表示 -64，-63，-62，……，-1，0，1，……至 63。符號位元也用來表示數值之大小，七個位元表示 0 至 127 之正整數。

假數部分除了符號位元外佔二十四個位元，這二十四個位元表示假數值之大小。這稱為符號大小表示法（SIGNED MAGNITUDE），和我們日常數字的表示法相同，符號和大小各自獨立，符號部分只表正負，而大小部分只表該數之絕對值，浮點二進位的小數點在假數之最左邊，也就是假數永遠以分數之形式出現，爲了使計算機表示的精確度最大，通常小數點之後第一個位元絕不能是0，這種表示法稱為標準化表示法（NORMALIZED REPRESENTATION）。360及370系統以十六爲底，每個十六進位數字以四個位元表示，所以對360及370系統而言，所謂標準化是指的小數點之後第一個十六進位數不能是0，但是第一個位元可以是0。例如小數點之後四個位元是（0001）<sub>2</sub>，（0001）<sub>2</sub>是十六進位數1，非0，雖然第一個位元是0，仍然合乎標準化的規則。但有例外，如果用浮點二進位表示0時，指數部分爲0，假數部分也爲0，當然小數點之後所有的位元全爲0，這稱為純淨零（CLEAN ZERO）。

計算機中浮點二進位數一定採用標準化的形式，計算機可以使假數部分往左或往右移動（LEFT/RIGHT SHIFT），往左每移動4個位元，指數部分減1，往右每移動四個位元，指數部分加1，例如（43 030000）<sub>16</sub>，假設小數點在3與0之間，經過標準化之後得（42 300000）<sub>16</sub>，前者指數部分爲（1000011）<sub>2</sub>，爲+3之意思，表示真正的小數點位置是往右邊移動三個十六進位數字，即（30.）<sub>16</sub>，爲十進位數+48。後者指數部分爲（1000010）<sub>2</sub>，即+2，指出真正的小數點應往右邊移動兩位，成（30.）<sub>16</sub>，亦+48，兩者表示法不同，其值相同。同理（BE 00E800）<sub>16</sub>與（BCE80000）<sub>16</sub>比較，前者未標準化，而後者已標準化，前者指數爲-2，後者指數爲-4，負指數表示小數點位置必須往左移，所以前者往左移兩個，後者往左移四個十六進位數，都得（.0000E8）<sub>16</sub>。

雖然用二進位較合乎計算機的構造，但是有時爲了特殊需要，仍需要十進位的表示法。例如在商業上使用的計算機，時常會牽涉到金錢，如果用二進位數字系統時，有些十進位數字無法用二進位完全表示，如1/3，只能以近似值表示，如果計算機使用的金錢單位是元，誤差尚小，如果

單位是萬、十萬，甚或千萬時，這些誤差就絕不允許存在，爲了避免此類誤差，只有用十進位數字系統，所以許多大型計算機不但有二進位的運算系統，也有十進位的運算系統，此外十進位又是人類所習慣的，使用者通常以十進位數字輸入計算機，計算機在運算時可以轉換成二進位數字，運算之後結果又被轉換成十進位，再交到使用者手中。

計算機裏十進位數字的表示法有區域十進位 (ZONED DECIMAL) 及聚集十進位 (PACKED DECIMAL) 兩種。區域十進位由 0, 1, 2, …… 9, +, - 共十二個字元構成，每一個數字佔八個位元，把這八個位元分成兩邊，左邊的四個位元稱爲區域位元 (ZONE BITS)，右邊四個位元稱爲數字位元 (NUMERIC BITS)。區域位元除了最右邊位元組表示符號之外，都以  $(1111)_2$  表示，符號位元之內容  $(1100)_2 = (C)_{16}$  或  $(1111)_2 = (F)_{16}$  代表正值， $(1101)_2 = (D)_{16}$  代表負值。例如  $(F1F2F3C4)_{16}$  爲 +1234， $(F7F9D5)_{16}$  爲 -795。

聚集十進位也包含 0, 1, 2, …… 9, +, - 共十二個字元，聚集十進位的符號位元爲最右邊的四個位元，仍以十六進位數 C, F 表正數，以 D 表負數，除了個位數與符號位元共佔一個位元組外，其他每兩個數字佔用一個位元組。例如  $(09876C)_{16}$  代表 +9876， $(231D)_{16}$  代表 -231。區域十進位不能拿來運算，聚集十進位可用以運算，區域十進位一個數字佔用一個位元組，浪費記憶體，聚集十進位每兩個數字佔用一個位元組，所以計算機通常以聚集十進位方法來表示十進位數。

計算機使用的字元必須是計算機能懂的字元，也就是說必須是字元表 (CHARACTER SET) 內的字元，包括

- (1) 英文字母或特殊符號：A, B, C, …… Z, 及 \$, #, @。
- (2) 數字：0, 1, …… 9。
- (3) 特殊符號：+, -, \*, /, =, (, ), &, ^, ;, …… 及空白 (BLANK) 等。

用程式處理一筆商業用的資料，或處理一個微分方程式時，時常需要用自然語言 (NATURAL LANGUAGE) 如英語。因此這一類的字元必須

能儲藏到計算機裏，我們可以用不同的位元樣式 (BIT PATTERN) 以代表不同的字元。要用多少個位元來表示這些字元呢？通常  $N$  個位元最多可表示  $2^N$  個字元，如 6 個位元，最多可以表示 64 個字元，7 個位元，最多表示 128 個字元，8 個位元最多表示 256 個字元。圖 1-7 是各種表示字元的數碼。

### 1-3 位址邊界 (ADDRESS BOUNDARY)

有些指令中運算元所代表的內含位址 (IMPLIED ADDRESS) 必須受某種邊界的限制，因此讀者必須了解什麼叫做位址邊界。位址邊界分成下列三種：

#### 1. 半字組邊界 (HALF-WORD BOUNDARY)：

是兩個位元組的意思，也就是說該內含位址是 2 的倍數。360 及 370 系統中，記憶器的最大範圍是  $2^{24}-1$ ，也就是說內含位址最大可至六個十六進位數字，半字組邊界的意思是這六個十六進位數字的最右邊數字必須是 0, 2, 4, 6, 8, A, C, E 八個數字中的一個。

#### 2. 全字組邊界 (FULL-WORD BOUNDARY)：

四個位元組的意思。內含位址的最右邊十六進位數字必須是 0, 4, 8, C 中的一個。如果內含位址屬全字組邊界的話，也一定屬於半字組邊界。

#### 3. 雙字組邊界 (DOUBLE-WORD BOUNDARY)：

八個位元的意思，即運算元的內含位址必須是 8 的倍數，即六個十六進位數字所表示的位址中，最右邊的數字必須是 0 或 8，凡屬於雙字組邊界的也一定屬於半字組及全字組邊界。見圖 1-8。