

第 3 章

定址法

Z 80 微處理器能執行 158 個指令，此 158 個指令之集合，即為 Z 80 微處理器之**指令集**（instruction set）。每一指令即為指使 Z 80 微處理器從事某種運算之一命令。正如語文中之及物動詞必須有受詞一般，幾乎每一指令都需有一被運算之運算元（operand），或稱數據（data）。此一運算元可直接出現於指令上，亦可以其它方式呈現給指令。指令獲得其運算所需之數據的方式，即稱為**定址法**（addressing modes）。

Z 80 微處理器使用十種不同之定址法。此些定址法，外加龐大之暫存器結構與指令集，使 Z 80 微處理器變成一十足能幹之微處理器。然而，不幸的是，這亦同時增加了 Z 80 之複雜性，尤其對初學者而言。所幸，學習上所付出之額外代價，終會得到報償的！第一，存取與運作資料的方式愈多，就愈容易寫出有效率之程式。第二，Zilog 公司已設計出一套查閱 Z 80 助憶符號與其相關十六進碼之極佳方式。使得用者絲毫未感不便。於努力學習之過程中，您將發現，其真正所蘊藏之好處並不止於此。在開始介紹各種定址法之前，必須再叮嚀一句的：初學時，將重點放在各種定址法之真正意義的了解，名稱與符號則由程式設計實務學習。欲對某一微處理器指令集之指令駕馭自如，徹底了解定址法乃是必備的！

Z 80 微處理器一共使用十種不同之定址法：

- 1 隱含定址（implied addressing）。
- 2 立即定址（immediate addressing）。
- 3 擴展立即定址（extended immediate addressing）。

4. 暫存器定址 (register addressing) 。
5. 暫存器間接定址 (register indirect addressing) 。
6. 擴展定址 (extended addressing) 。
7. 修正零頁定址 (modified zero-page addressing) 。
8. 相對定址 (relative addressing) 。
9. 索引定址 (indexed addressing) 。
10. 位元定址 (bit addressing) 。

茲將其分別介紹如下：

3-1 隱含定址

於隱含定址，指令之運算碼隱含某一或某幾個 CPU 暫存器包含運算元。因此，此些指令僅有運算碼欄，而無運算元（或運算元位址）欄。CPU 只要解碼運算碼，即可知運算元之所在。使用此種定址之指令，翻譯成機器碼儲存於記憶器之情形，即如圖 3-1 所示。

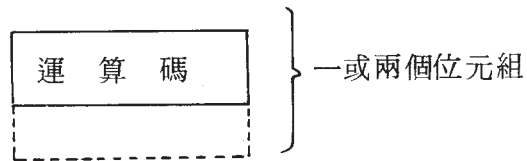


圖 3-1 隱含定址

於 Z 80，隱含定址之指令可能長一位元組或兩位元組。於 Z 80，使用隱含定址的大多為算術邏輯運算指令。每一算術／邏輯運算指令均隱含累加器，為其中一運算元與最後結果之所在。將累加器內含值加一之指令 INC A，即為一使用隱含定址之指令。該指令翻譯成機器碼即為 3C（十六進制），長一個位元組。將索引暫存器 IY 之內含，抄至堆疊指示器之指令 LD SP, IY，亦為一隱含定址指令。該指令之機器碼 FDF 9，長兩位元組。

隱含定址指令之優點為指令簡短，執行迅速。由於 Z 80 進一步區分隱含定址與暫存器定址，因此，隱含定址僅限於那些並未使用特別之“暫存器碼”欄的指令。這種區分法使得 Z 80 又多出了一種定址法，此乃為何定址法無法絕對作為微處理器能力之象徵的原故。

Z 80 使用隱含定址之指令有 LD r, r' ; ADD A, r ; ADC A, s ; SUB S ; SBC A, s ; AND s ; OR s ; XOR s ; CP s ; INC r 。

3-2 立即定址

將運算元（或稱數據）直接給在指令上之定址方式，即稱為立即定址。“立即”之名導自，於此種指令，運算元“立即”緊接於運算碼之後。於 Z 80，立即定址指令翻譯成機器碼後之形式，即如圖 3-2 所示。

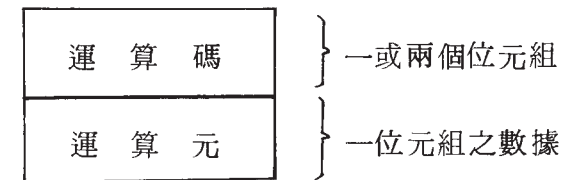


圖 3-2 立即定址指令之格式（機器碼）

運算碼後緊接的即為八位元之數據。

將一定值存入累加器之指令

LD A, 5

即為一立即定址指令之例子。上述指令執行完後，十進數 5 即被存入累加器。注意，於組合語言書寫時，立即運算元直接寫上。於 Z 80，立即定址主要使用於八位元之取入（Load），算術，與邏輯指令。此些指令之組合語言格式為

LD r, n
ADD A, n

式中， r 代表某一 CPU 暫存器， n 代表一八位元數據。

3-3 擴展立即定址

於 Z 80，立即運算元可長十六位元。Zilog 公司特別將此種定址稱為**擴展立即定址**。擴展立即定址指令翻譯成機器碼後，儲存於記憶器的情形即如圖 3-3 所示。

運 算 碼	一或兩位元組
運 算 元	低次八位元
運 算 元	高次八位元

圖 3-3 擴展立即定址之格式

將十六位元之十六進數 124 D 取入索引暫存器 IX 之指令

LD IX, 124 DH

，即為一擴展立即定址指令之例子。該指令翻譯成機器碼儲存於記憶器的情形即如

DD	}	運 算 碼
2 1		
4 D	}	立即運算元
1 2		

注意，由於運算碼長兩個位元組，因此，該指令必須以連續四個記憶

位置加以儲存。

於 Z 80，擴展立即定址僅用於十六位元取入 (LD)，跳越 (JP)，與副程式叫用 (CALL) 等指令。

3-4 暫存器定址

於暫存器定址，指令選取一個或一個以上之 CPU 暫存器。指令之運算碼會有一欄 (幾個位元) 指明，指令之運算元究竟含於那些 CPU 暫存器。將某一暫存器之內含與累加器內含作邏輯且 (AND) 運算之

AND R

指令 (式中， R 代表 A, B, C, D, E, H, 或 L 中任一者)。以及將某一 CPU 暫存器之內含向左移動一位元位置之

RL R

指令 (式中之 R 同前)，皆為暫存器定址指令之典型例子。茲將此兩指令分別說明於下。

RL R 指令翻譯成機器碼之情形如圖 3-4 所示。指令之機器碼共佔兩位元組。第 2 位元組之最後 (低次) 三個位元，即指明了含

第 1 位元組	1 1 0 0 1 0 1 1	運算碼 CBH
第 2 位元組	0 0 0 1 0 R	0 0 0 1 0 ₂ = 運算碼 R 為 CPU 暫存器碼

圖 3-4 暫存器定址之 RL R 指令

運算元之某一 CPU 暫存器，此一指明情形為

R	所代表之暫存器
0 0 0	B
0 0 1	C
0 1 0	D
0 1 1	E
1 0 0	H
1 0 1	L
1 1 1	A

注意到，除了 110_2 外，其它之位元組合情形皆用上了。此乃因為 $R = 110_2$ 特別留給了暫存器間接定址使用。嚴格說來，暫存器定址之每一指令可分開來看，而變成一種隱含定址指令。

AND R 指令之機器碼格式則如圖 3-5 所示。該指令為單位元組指令。指令運算碼之最低次三位元，用以指明 AND 運算所涉及之暫存器（除了隱含之累加器 A 外）。此地暫存器的寫碼情形，完全與 RL R 指令相同。

1 0 1 0 0	R
-----------	---

10100_2 為 AND 之運算碼
R 為 CPU 暫存器碼

圖 3-5 暫存器定址之 AND R 指令

於 Z80，使用暫存器定址之指令包括八位元之算術與邏輯指令，十六位元之算術指令，旋轉移位指令，位元置定、清除、與測試指令等。

3-5 暫存器間接定址

所謂暫存器間接定址，即指令之運算元的位址，在 CPU 之某一暫存器內。換言之，指令之運算元仍存在記憶體內。指令必須先參照

某一 CPU 暫存器之內含，將之當成運算元位址，然後再根據此一記憶位址，前去記憶體中拿取運算所需之運算元。暫存器間接定址之情形如圖 3-6 所示。

於 Z80，暫存器間接定址之指令指明一十六位元之 CPU 暫存器，以作為運算元所在之記憶位置的指示器（pointer）。（亦即，該暫存器含運算元之記憶位址），此一型態之指令甚為能幹，並且具有相當廣泛之用途。其指令格式為



將十六位元暫存器對 HL 所指之記憶位置的內含，取入累加器 A 之指令

L D A, (HL)

即為一暫存器間接定址之例子。該指令之十六進機器碼為 7E，長一個位元組。特別注意，在組合語言符號上，暫存器間接定址以將指定暫存器加一括弧表示。

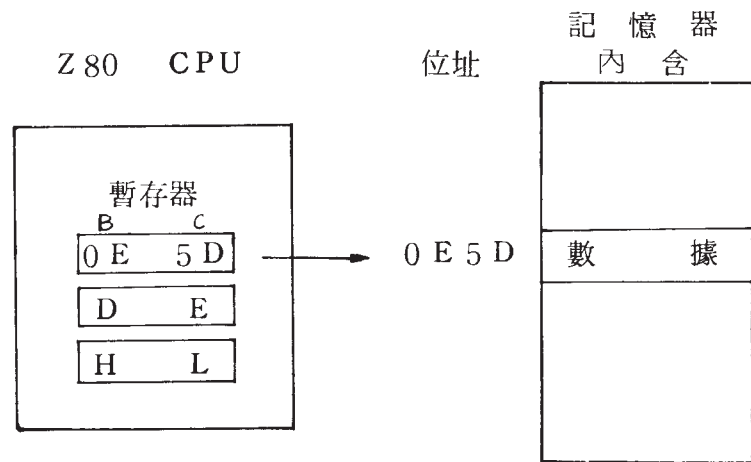


圖 3-6 暫存器間接定址

對許多指令言，暫存器間接定址用以指明一十六位元之運算元。在此一情況下，暫存器內含指至運算元之低次位元組，然後，暫存器之內含自動加 1，以指至運算元之高次位元組。

POP BC

指令即為此種情況之一例子。該指令將 SP 所指之記憶位置的內含取入 C 暫存器，然後，再將 SP 之值加 1 所指之記憶位置的內含，取入 Z80 CPU 內之 B 暫存器。

於表列 (list) 處理應用，暫存器間接定址甚為好用。藉著每次增加暫存器之內含值，暫存器間接定址指令可用以存取一連串的資料項目 (不論連續儲存者或不連續儲存者)。暫存器間接定址與索引定址很類似。事實上，索引定址即為暫存器間接定址的一種。唯一不同的是，索引定址先將暫存器內含加上一位移 (displacement)，所得結果再當為運算元位址。Z80 指令集中，功能強大之區段搬運與搜尋指令，即為以暫存器間接定址，加上自動暫存器內含加一，減一，與比較作業所構成的。

3-6 擴展定址

於擴展定址指令，運算元之十六位元位址直接給在指令上。若為程式跳越 (jump) 指令，則該位址所指即為控制欲轉移之目的位置。擴展定址指令之格式為

運 算 碼	} 一或兩個位元組
低次運算元位址	
高次運算元位址	

圖 3-7 擴展定址指令

若擴展定址用以指明運算元 (數據) 之來源 (source) 或目的 (destination) 位址，則在組合語言符號上以 (nn) 之符號表示。式中，nn 為一十六位元位址，每一個 n 代表八位元位址。將累加器 A 之內含存至位址 2504 H 之記憶位置的指令

LD 2504 H, A

即為一例。指令中，運算元之目的位址 2504 以括弧括起。該指令翻譯成機器碼，儲存於記憶器之情形為

3 2	}	運算碼
0 4		低次與高次
2 5		運算元位址

於跳越指令，控制最後欲轉移之目的位置的位址，就不需加以括弧，例如，擴展定址之跳越指令

JP 2504 H

即將程式控制轉移至位址 2504 H 之記憶位置上所儲存之指令。

由於運算元之真正 (最終) 位址直接寫在指令上，不因組合翻譯程序而改變，因此，擴展定址又稱絕對定址 (absolute addressing) 或直接定址 (direct addressing)。

3-7 修正零頁定址

Z80 有八個指令使用修正零頁定址。此些指令，稱為重始 (restart) 指令 (事實上即為副程式叫用指令)，使程式控制轉移至八個零頁記憶位置之一——位址 0000 H, 0008 H, 0010 H, 0018 H, 0020 H, 0028 H, 0030 H, 0038 H 等八個位置。八個重始指令皆僅長一個位元組。如圖 3-8 所示，指令之運算碼指明了產生

那一位址：

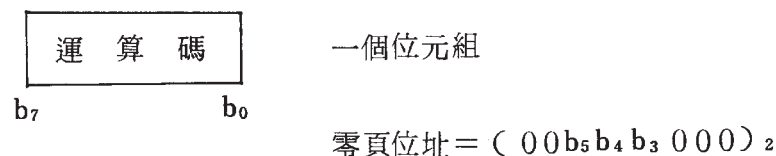


圖 3-8 修正零頁定址

例如，若運算碼之 $b_5b_4b_3 = 111$ ，則有效位址即為 $0038H$ 。

重始指令之主要目的乃在叫用 (call) 一常用之副程式 (subroutine)。其優點為節省空間 (指令短) 與時間。若用副程式叫用指令 (CALL)，則就需三個位元組加以儲存。

RST 10H

即為一重始指令例子。該重始指令之十六進機器碼為 $D7$ ，因此，程式控制最後轉至位址 $0010H$ 之記憶位置上的副程式。注意，重始指令之用法為，RST 後緊接之兩位十六進數字，即等於副程式所在位置之位址的低次兩位十六進數字。換言之，RST 18H 即使控制轉移至 $0018H$ 之記憶位置等。

3-8 相對定址

於 Z80，相對定址僅用於相對跳越 (relative jump，以 JR 代表) 指令。相對定址之真意為，跳越運算之目的位址 (或運算數據之真正位址)，是基於目前指令所在之記憶位置，以相對之方式指明。如圖 3-9 所示，於相對定址，緊接指令運算碼後即為一八位元，表示成有號 2 補數形式之位移 (displacement)。微處理器於執行相對跳越指令時，會將此一位移加上當時程式計數器之內含 (即相對跳越指令之次一緊接指令所在之記憶位置的位址)，作為跳越指令之目的位址。

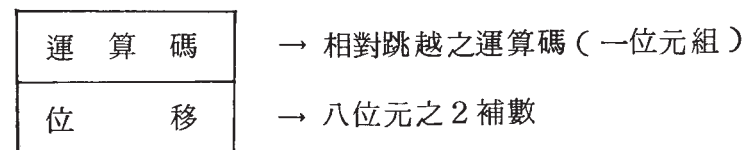


圖 3-9 相對定址指令

由於八位元有號 2 補數所能表示之最大數值為 -128_{10} 至 $+127_{10}$ (10000000_2 至 01111111_2)，因此，如圖 3-10 所示，自相對跳越運算碼算起，相對跳越指令最多僅能往回 (位址小的方向) 跳 126 個記憶位置，往前 (位址更大之方向) 跳 129 個記憶位置。(因為，相對跳越指令本身佔兩個記憶位置，指令取出後，程式計數器已指至下一緊接指令。) 換言之，相對跳越僅能及指令前後之鄰近位置。

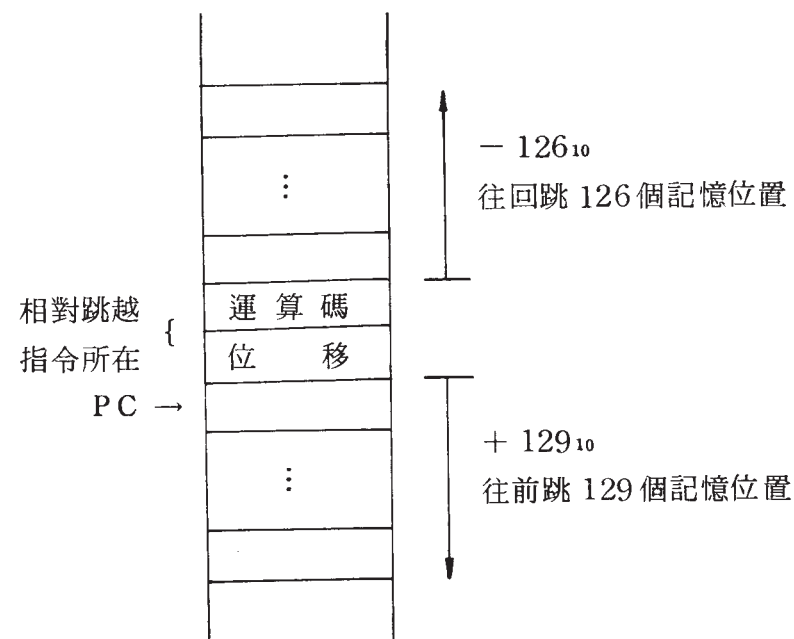


圖 3-10 相對跳越所及之範圍

下面，我們舉兩個例子說明相對跳越指令之執行情形。圖3-11 A爲

J R 0 9 H

之執行情形。該指令翻成機器碼爲 18 09。由於運算碼後之位移爲正值(09)，因此，程式計數器之內含值加 9，控制往前跳至相對越指令以下第十個記憶位置上之指令。

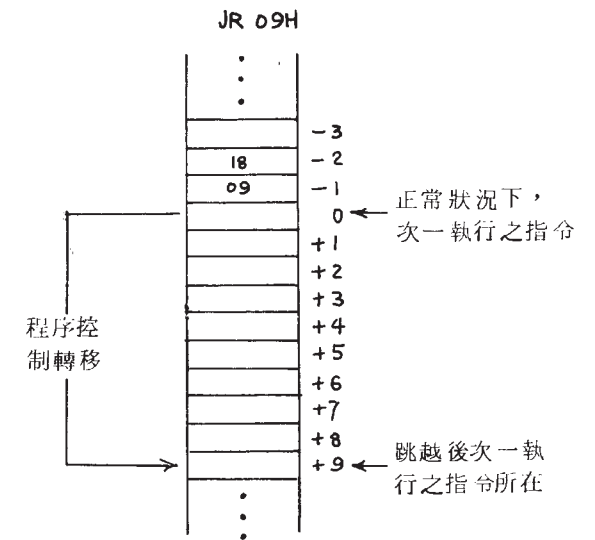
圖3-11 B則爲相對跳越指令

J R F C H

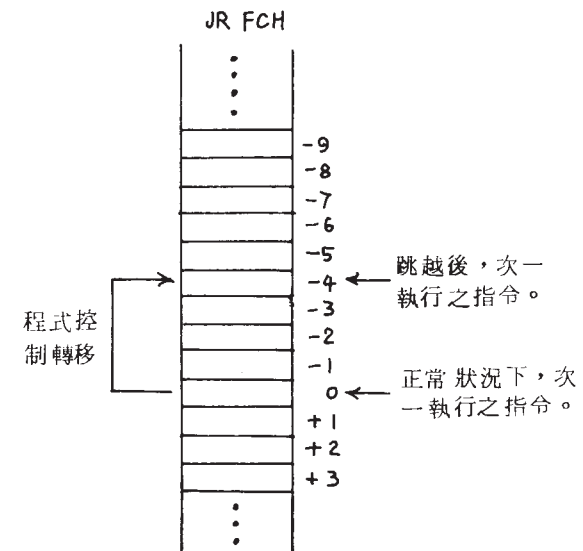
之執行情形。該指令之機器碼爲 18 FC。由於運算碼後之位移FC爲負數(-4)，因此，控制往回跳四個位置，跳至相對跳越指令前第2個位置上之指令。微處理器在執行此一指令時，即將指令上之位移，加至程式計數器之低次位元組，捨棄進位，結果再存回程式計數器之低次位元組。由以上說明，聰明之讀者應可領悟，相對跳越指令之位移應如何求得——只要將目的位址減去相對跳越指令次一位置之位址即可。

相對定址主要用於迷你電腦與微電腦，以減短指令並且縮小程序所佔之記憶空間。雖然使用直接(即擴展)定址能選取記憶之任一位置，但此種指令却須佔三個位元組之記憶空間。零頁與相對定址可將指令減短至兩位元組，這不僅可節省程式所佔之記憶空間，同時亦可減短指令之執行時間。當然，零頁與相對定址亦有其缺點存在；零頁定址僅能選取第零頁之256個記憶位置，而相對定址亦僅能選取指令附近之256個記憶位置。

相對定址之另一優點爲，此種定址之指令所構成的程式，**可重新定位(relocatable)**。所謂可重新定位即指，將程式搬動至另一段記憶區域，並不影響程式之執行結果，其同樣能正確執行。欲測知一程式是否可重新定位，你只要將其原封不動地整體搬至另一段記憶區域，然後再看其是否仍能正確執行便知。很明顯地，含有擴展定址之一般跳越指令的程式，即爲不能重新定位。因爲，一般跳越指令所指



(A)



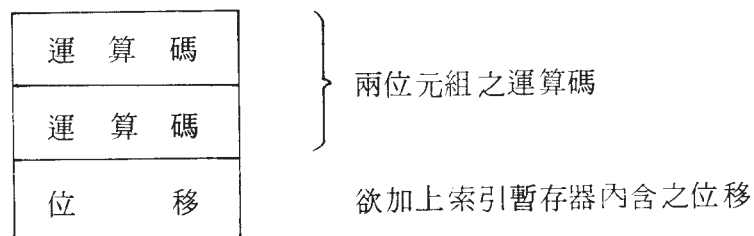
(B)

圖3-11 相對跳越指令之執行情形

明的是絕對位址，程式一移動後，該絕對位址位置上所儲存的，已非原先預定之指令。在變換程式位置時，對有關之絕對位址作適當調整，以便程式能保持正常動作之程序，即稱**將程式重新定位**（relocate）。

3-9 索引定址

Z 80 微處理器具有兩個稱為**索引暫存器**（index register）之十六位元特殊用途暫存器，分別稱為 IX 與 IY。此兩暫存器主要用於索引定址。索引定址非常類似於暫存器間接定址，因為，兩者皆以一十六位元之暫存器，指至儲存於記憶器中之數據。唯一不同的是，於索引定址，指令運算碼後必須有一準備加至索引暫存器內含，以便產生運算元之真正位址（又稱**有效位址**）之位移。該位移為八位元，並表示成 2 補數形式。注意，**索引作業之加法運算並不改變索引暫存器之原有值**。就最終位址之獲得方式而言，索引定址亦極類似相對位址。所不同的是，索引定址牽涉索引暫存器，且目的在於存取數據；而相對跳越涉及程式計數器，且目的在於控制轉移。圖 3-12 所示即為索引定址之機器指令的格式。



運算元之有效位址 = 索引暫存器內含 + 指令上之位移

圖 3-12 索引定址之指令格式

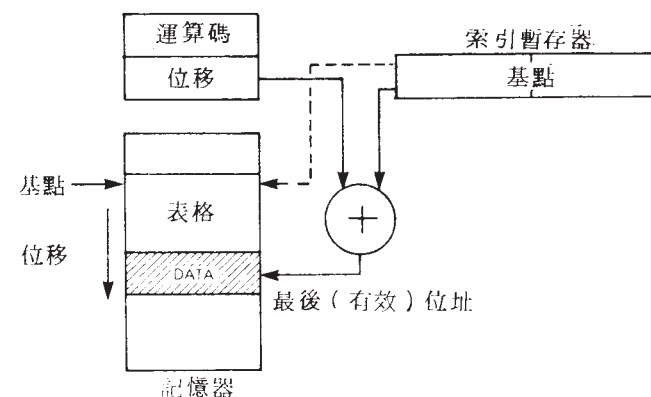


圖 3-13 索引定址

於 Z 80 之組合語言符號，索引定址記成

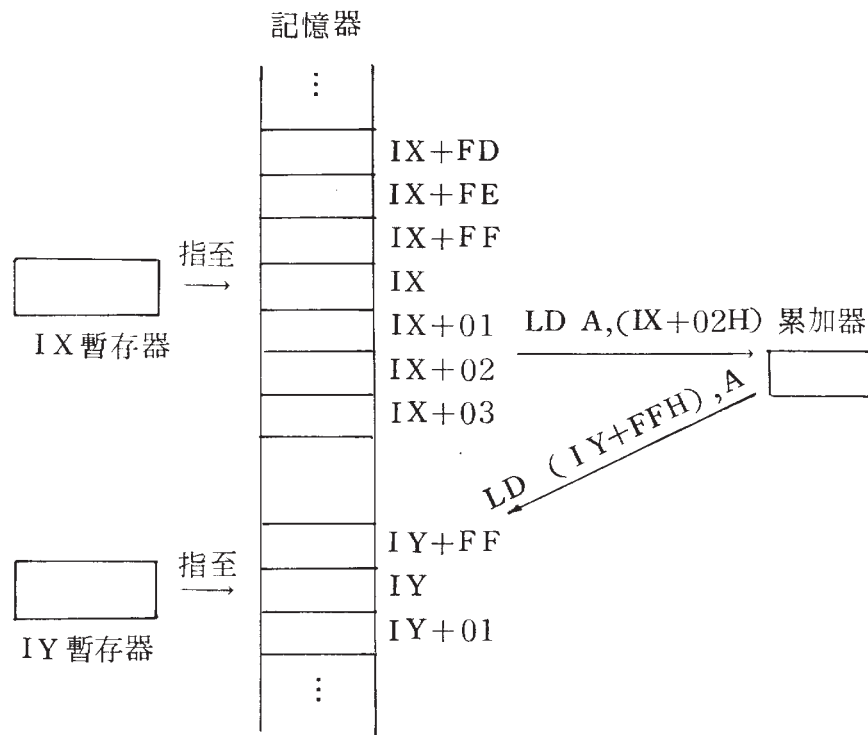
$(IX + d)$ 或 $(IY + d)$

，式中， d 即為緊接運算碼後之位移。下面是兩個索引定址指令之例子

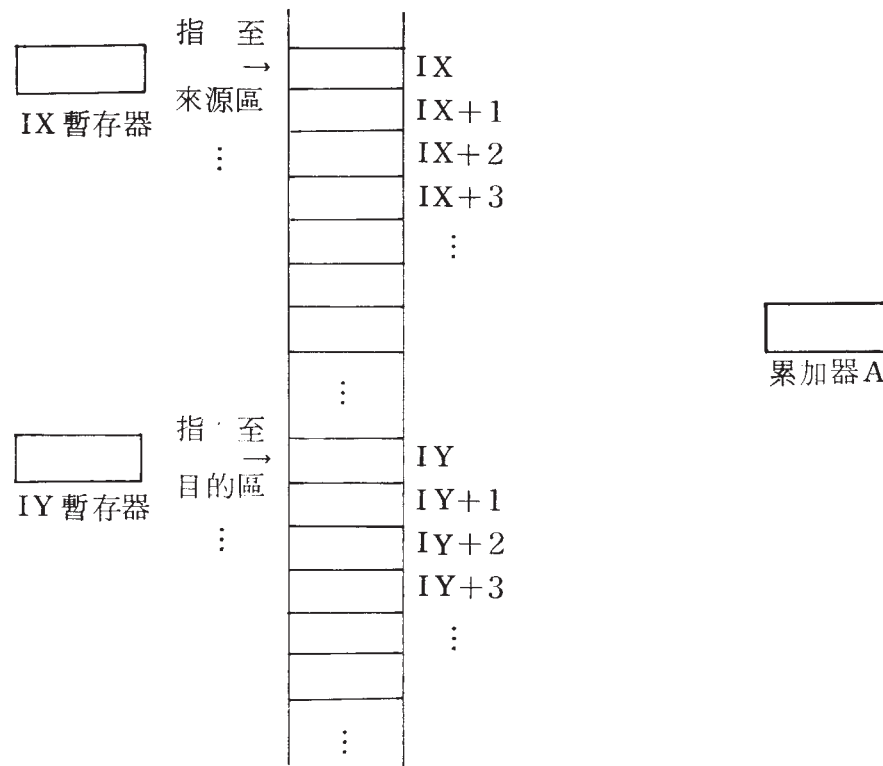
LD A, (IX + 02H)

LD (IY + FFH), A

前一指令將比 IX 所指之記憶位置再高兩位置之內含，取入累加器 A。該指令之機器碼為 DD 7E 02。後一指令將累加器之內含，存至索引暫存器 IY 所指之記憶位置的前一位置，因為，位移 FF 即為 -1。該指令之機器碼為 FD 77 FF。圖 3-14 說明了索引定址指令之位移 d 的意義。



索引定址亦是一非常能幹 (powerful) 之定址技巧，尤以處理表列 (list) 或表格 (table) 資料時，更能見效。於此等應用，索引暫存器可指至表列或表格之起點，然後，以指令上之位移存取表列中所要之資料項目。必要時，索引暫存器之內含亦可加減，以移動指示器所指之位置。圖 3-15 所示即為以兩個索引暫存器，從事區段搬運之情形。開始時，IX 指至來源區 (source block) 之第一項資料，而 IY 則指至目的區 (destination block) 之第一位置。LD A, (IX + 00H) 指令將欲搬家之資料取入累加器，而 LD (IY + 00H), A 則將之存至目的區之位置。第一項資料搬完後，IX 與 IY 兩暫存器之內含可分別加 1 (以 INC IX 與 INC IY)，



然後，微處理器再執行上述兩個指令，以搬動第 2 項資料。如此類推，直至搬完所有資料為止。

注意，於此一例子，若來源區之起點與目的區之起點相距在 128 個記憶位置以內，則程式亦可僅使用一個索引暫存器。倘若來源區與目的區相距 100₁₀ 個記憶位置，且程式使用 IX 暫存器，指至來源區之起點，則程式所必須具備之指令為

```
LD      A, ( IX + 00H )
LD      ( IX + 64H ), A
INC     IX
```

此外，若目的區與來源區有部份互相重疊，則資料必須由表列之最後一項開始搬起，方不致發生破壞某些原有資料之錯誤。

於 Z 80，八位元取入，八位元算術、邏輯、移位、旋轉、位元置定、清除，與測試等指令，皆使用索引定址法。

3-10 位元定址

Z 80 有許多指令，能將某一記憶位置或暫存器所儲存之位元組的某一位元，置定為 1、清除為 0，或測試其內含值。此些位元運作指令（分別為 SET, RES, 與 BIT），使用暫存器、暫存器間接、或索引等定址法，以指明有關之記憶位置或暫存器；而以運算碼之其中三個位元，指明指令所欲運算之位元。

圖 3-16 所示即為一使用索引定址，將某一記憶位置所存資料之某一位元置定為 1，之 SET B, (IX + D) 指令的情形。該指令長四個位元組，第 1 與第 2 位元組為運算碼，第 3 位元組為索引定址之位移，而第 4 位元組之第 3, 4, 5 等位元，即為用以指明欲運算位元之位元碼。該位元碼與其所代表之位元的關係為

B 欄	所指明之位元
0 0 0	b ₀ (第 0 位元)
0 0 1	b ₁
0 1 0	b ₂
0 1 1	b ₃
1 0 0	b ₄
1 0 1	b ₅
1 1 0	b ₆
1 1 1	b ₇ (最高次位元)

第 1 位元組	1 1 0 1 1 0 1	} 運算碼 DD CB
第 2 位元組	1 1 0 0 1 0 1 1	
第 3 位元組	D	← 索引之位移值
第 4 位元組	1 1 B 1 1 0	← B 為位元碼

圖 3-16 位元 / 索引定址之 SET B, (IX + D) 指令

圖 3-17 所示則為另外位元定址之例子。例中之指令乃將 Z 80 CPU 之 C 暫存器的某一位元置定為 1 之

SET B, r

指令。為了方便比較起見，特將運算前後暫存器之內含同時列出。

暫存器 C 之內含		指 令
運 算 前	運 算 後	
0 1 0 1 0 0 0 0	0 1 0 1 0 0 0 1	SET 0, C
	*	
0 1 0 0 0 0 0 1	0 1 0 0 1 0 0 1	SET 3, C
	*	
1 1 1 0 0 0 0 0	1 1 1 1 0 0 0 0	SET 4, C
	*	
0 0 0 1 1 1 1 0	1 0 0 1 1 1 1 0	SET 7, C
	*	

圖 3-17 位元定址之例子

表 3-1 摘要了 Z 80 之所有定址法。該表使用之符號同於前面所描述者。爾後各章之討論，我們亦將使用此一套符號。爲了便於讀者比較，8080 所使用之指令與定址法，於點下特別加一橫線。而 8080 與 8080 皆有者，則加雙線。

表 3-1 Z80 定址法

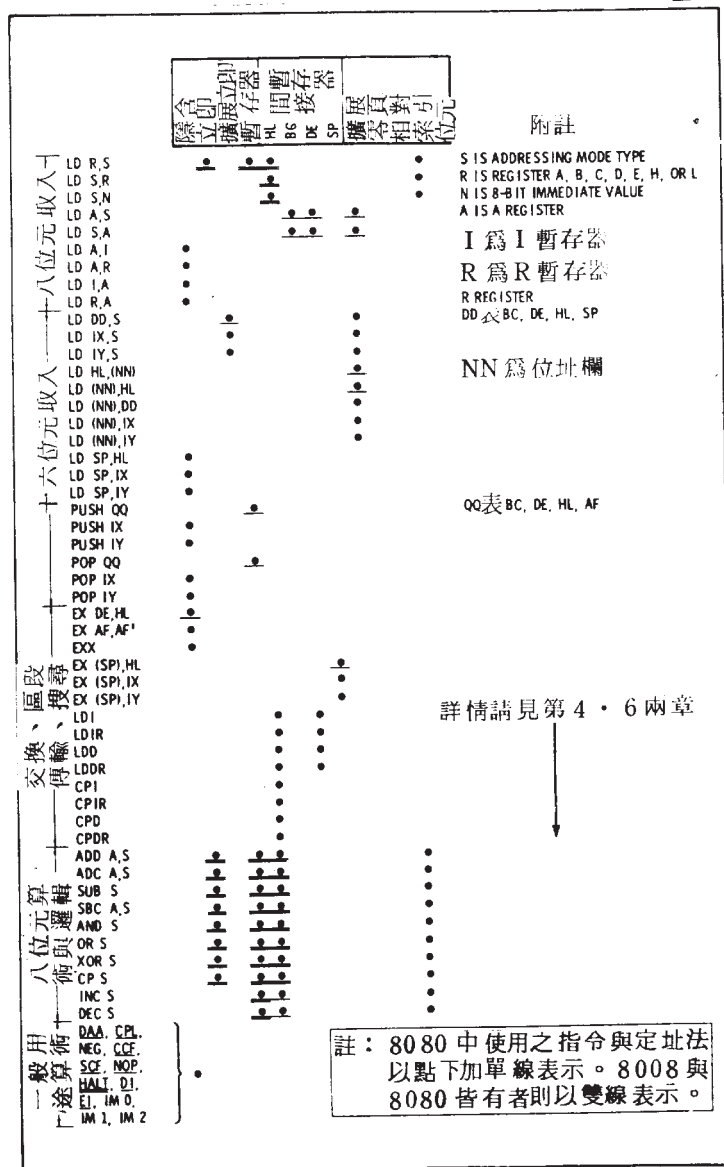
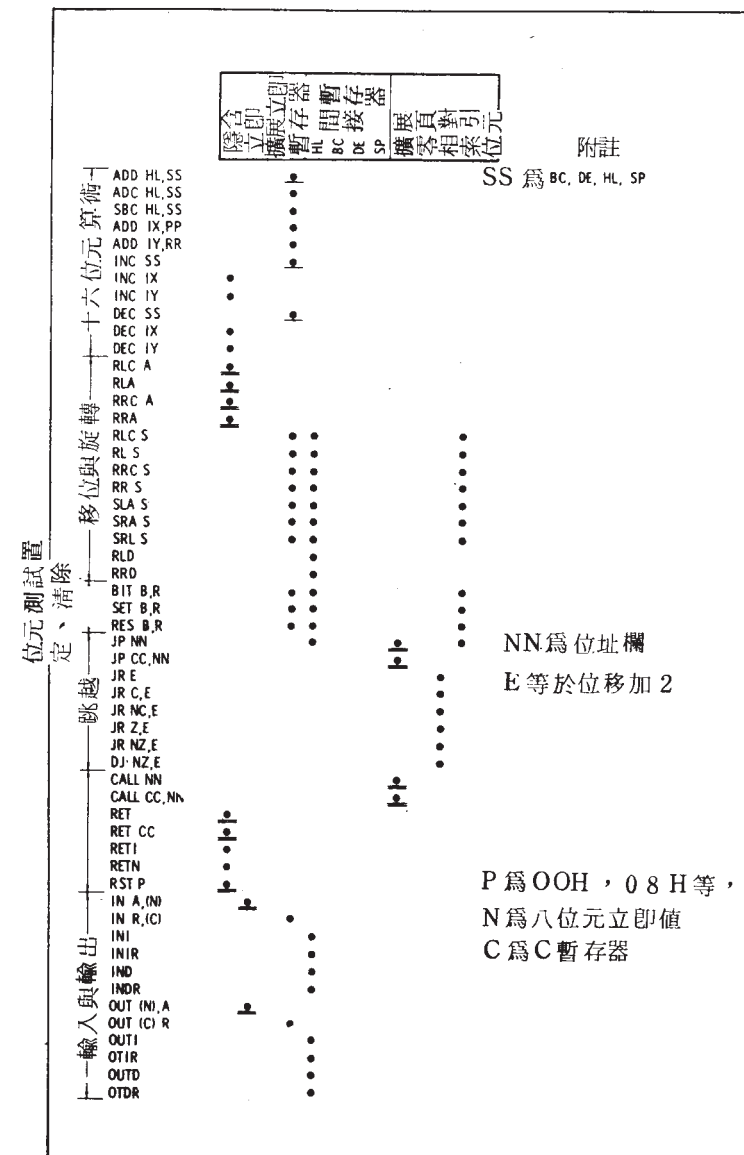


表 3 ~ 1 Z80 定址法 (續)



第 4 章

Z80 指令集

於介紹過 Z80 微處理器之內部結構與各種定址法後。本章，我們開始介紹 Z 80 微處理器之指令集，以便讀者能儘快寫程式。首先，我們將先分析一般計算機所具有之各類指令，以及其功能。然後，分別介紹 Z 80 指令集之各類指令。最後，再一一將 Z 80 指令集之各個指令，分別舉例詳盡說明於最後，以便讀者參考。由於 Z 80 指令集複雜異常（至少對初學者而言），因此，作者認為此種分層，然後再個別詳盡之介紹方式，對於初學者或一般之程式設計者，較有助益。

4-1 計算機指令之種類

計算機之指令有多種不同的分類方式，並且無一定之標準。在此，我們將之分成五類：

1. 資料傳送（或稱資料傳輸）。
2. 資料處理。
3. 測試與控制轉移。
4. 輸入 / 輸出。
5. 控制。

下面，我們一一介紹此各類指令。

資料傳送指令

資料傳送指令將資料傳遞於暫存器之間，暫存器與記憶器之間，

或暫存器與輸入／輸出設備之間。

有些資料傳送指令扮演特殊之角色。譬如，堆疊作業指令 PUSH（推入）與 POP（取出）即是一例。此些指令將資料傳遞於累加器與堆疊器之間，而一方面又更新堆疊指示器之內含。

某些微處理器將資料傳送指令分成三類。將資料自記憶體或輸入／輸出電路傳遞至 CPU 暫存器之指令，稱為**取入**（load）指令，組合語言以 **LD** 表示。取入指令之特徵為，資料傳遞之目的地必為 CPU 內之某一暫存器，而來源地則位於 CPU 之外。將資料自 CPU 暫存器傳遞至記憶體或輸入／輸出電路之指令，稱為**存出**（store）指令，組合語言助憶符號寫成 **ST**。很明顯地，就資料運動之方向而言，取入指令與存出指令正巧相反。最後，將資料自某一 CPU 暫存器傳遞至另一 CPU 暫存器之指令，稱為**暫存器傳輸**（register transfer）指令。組合語言以 **T** 表示。圖 4-1 所示即為取入與存出指令之含意。

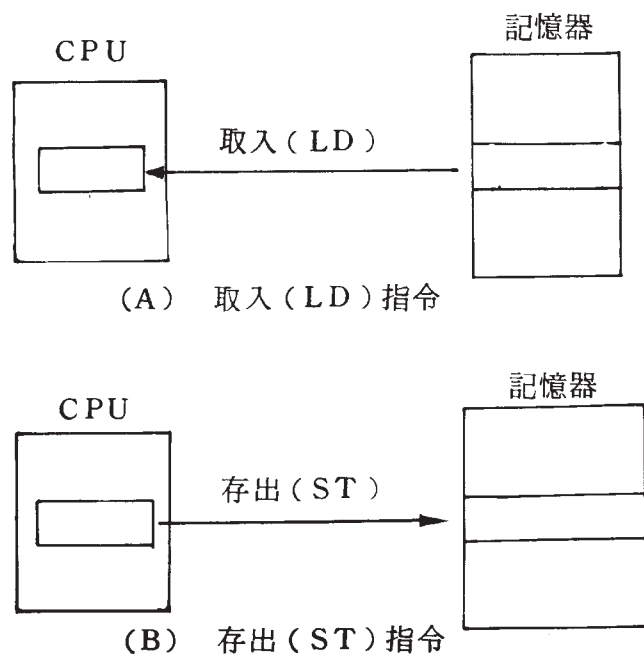


圖 4-1 取入與存出指令

於組合語言符號，取入指令之一般格式為

LD **r, S**

其中，**r** 為資料傳遞之目的暫存器的名稱，而 **S** 為資料來源。**S** 欄可為一立即運算元，亦可為運算元所在之記憶位置的位址，或索引定址等。

存出指令之一般格式為

ST **r, D**

式中，**r** 為資料之來源暫存器的名稱，而 **D** 為資料傳遞之目的地。

暫存器傳輸指令之一般格式為

T s d

其中，**s** 為資料來源暫存器之名稱，**d** 為資料目的暫存器之名稱。例如，將 **A** 暫存器之內含傳遞至 **B** 暫存器之指令，即記為

T A B

於著名之 Motorola 6800 以及其類似之 6502 等微處理器，資料傳送指令即如上述地分成三類。不過，於 **Z 80** 微處理器，所有資料傳送指令皆稱作取入指令，並以 **LD** 表示。

資料處理指令

資料處理指令又可分為五類：

1. 算術運算指令（如加、減等）。
2. 邏輯運算指令（如 AND, OR, XOR 等）。
3. 加一與減一指令。
4. 位元運算指令（置定與清除）。
5. 移位與旋轉指令。

注意，若計算機欲作有效之資料處理，則其必須具備能力較強之算術指令，如乘與除指令等。很遺憾的，大多數微處理器都無此些指令。此外，計算機尚須具有能力較強之移位與扭曲（skew）指令，諸如移動 **n** 個位元位置，或將兩半位元組資料對調等。此些指令在大多

數微處理器亦都找不到。

關於算術與邏輯運算指令，有一點必須說明的是，於大多數微處理器，這些指令均**隱含累加器為其中之一運算元的來源，以及運算結果之目的暫存器**。更明確地說，加法指令將運算元加上累加器之內含，結果存回累加器。減去指令將運算元自累加器內含減去，結果（差）存回累加器。邏輯 AND 指令將指令運算元與累加器內含作邏輯“且”（AND）運算，結果存於累加器，等等。

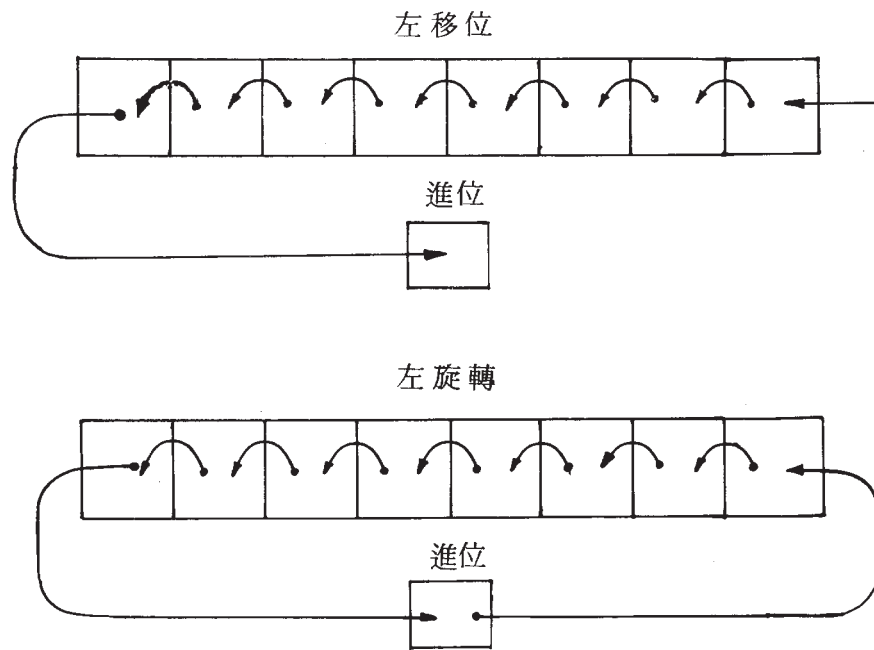


圖 4-2 移位與旋轉

在此，於討論 Z 80 指令之前，我們先說明一下移位 (shift) 與旋轉 (rotate) 之差別。**移位**即將某一暫存器或某一記憶位置之內含，向左（左移）或向右（右移）移動一個位元位置。被移出暫存器或記憶位置之位元，進入進位旗號位元。而另一端空出之位元位置則補 0。“算術右移”之情況例外，於此一情況，最高次位元一直向右複製。

於**旋轉**，整個資料字組與進位形成一串，順次移動。被移出資料字組之位元進入進位位元，而進位位元原有值則移入資料字組之另一端。這相當於一九位元之旋轉。雖然大多數微處理器皆不能，但 Z 80 却可作八位元之旋轉運算——移出資料字組之位元直接進入資料字組之另一端，不再經由進位。圖 4-2 所示即為移位與旋轉運算的情形。

最後，將資料字組右移時，最好能多一種所謂的符號展延 (sign extension) 或**算術右移** (arithmetic shift right)。於 2 補數運算時，特別是在作浮點數 (floating number) 運算時，通常必須將一負數右移。在將 2 補數負數右移時，由最左邊進入之位元必須恒為“1”（亦即，符號位元必須一直連續重複）。此即為算術右移。

測試與控制轉移指令

測試指令測試某一特定暫存器之某些位元值，看其究為 0 或 1。最低程度，其必須能測試旗號暫存器。因此，旗號暫存器應儘可能包含最多之旗號。此外，為了方便起見，最好我們亦能以單一指令測試幾個位元狀態之組合。最後，我們最好能測試任意暫存器之任意位元位置，以及某一暫存器值與另一暫存器值之比（大於，等於，小於）。微處理器之測試指令一般皆僅限於測試旗號暫存器之某一位元。Z 80 則比一般微處理器強一點。

程式使用測試指令主要在於**測知先前指令之運算結果**（或微處理器之當前狀態），以便決定 (make decision) 進一步應採取何種行動。因此，測試指令後經常緊接的就是**跳越** (jump) 指令。跳越指令使程式控制轉移至程式之其它段落，使緊接被執行之指令為其它指令，而非緊接目前指令之後的指令。

若依名稱分，控制轉移指令可分為三類：

1. 跳越。此種指令指明一十六位元之全位址，可跳至記憶器之任一記憶位置。
2. 相對跳越。相對跳越指令即為使用相對定址之跳越指令。此種

指令之位址欄指明一八位元之位移，跳越僅能及指令所在附近之記憶位置。有些微處理器稱相對跳越指令為分支（branch）指令。

3. 副程式叫用（call）與回返（return）指令。（所謂副程式即為儲存於另一段記憶區域之一常用程式。此種程式唯有當叫用指令將控制傳給它時才會被執行。）

依跳越動作發生之條件分，跳越或相對跳越指令可分為**無條件跳越**（unconditional jump）與**條件跳越**（conditional jump）指令兩種。無條件跳越指令恒使跳越動作發生。而條件跳越指令是當某一陳述條件滿足時，跳越動作才發生；否則，若條件不滿足，微處理器就像沒執行過條件跳越指令一般，繼續往下執行次一緊接指令。事實上，在一般微處理器，條件跳越指令可看成就是將旗號暫存器之測試指令，與其緊接之跳越指令合成之指令。此種指令構成了計算機程式之**決策**（decision-making）能力。

有些機器甚至還有雙途或三途之跳越指令（如依比較結果是大於、等於、或小於而跳越之指令），以及向前或向後略過幾個指令之“略過”（skip）指令。略過指令事實上就是跳越指令。最後，在迴路時，迴路末了通常有加一或減一運算，然後再測試且跳越。Z 80 微處理器就具有一包含此些所有作業之指令，因而大大地簡化了程式設計之工作，並且增進效率。

輸入 / 輸出指令

輸入 / 輸出指令乃管理輸入 / 輸出設備之特殊指令。事實上，大多數八位元之微處理器均使用記憶映像輸入 / 輸出（memory-mapped I/O）結構：輸入 / 輸出設備如同記憶晶片般地連接至位址巴士，且以同樣方式選取。對程式設計者而言，輸入 / 輸出設備就宛如記憶位置。由於所有記憶器型之作業通常都需要三個位元組之指令，以致速度緩慢。因此，為了達成有效的輸入 / 輸出作業，最好能有較

短之定址法，以使作業速度具有關鍵性影響之設備，能駐於第零頁之記憶位置內。可是，零頁定址即使是有，通常也都用於讀寫記憶器，因此，無法有效應用於輸入 / 輸出設備。職是之故，與 8080 一樣，Z80 另外具備了特殊之輸入 / 輸出指令。用者可以選擇將輸入 / 輸出設備視同記憶位置之方式，亦可使用輸入 / 輸出指令，將輸入 / 輸出設備個別看待。後者之方式即稱為**隔離式輸入 / 輸出**（isolated input/output）。

控制指令

控制指令供應同步信號，並且可能延緩或插斷某一程式之執行。它們亦可作為中斷（break）或模擬插斷。插斷將於後面討論輸入 / 輸出技巧時，再予介紹。

4-2 Z80指令集

Z 80 微處理器主要設計以取代 Intel 8080 微處理器，並提供額外之能力。因此，Z80 之指令集包含了 Intel 8080 之所有指令，並且還增加一些新的指令。看八位元運算碼之有限位元數，您可能極想知道 Z80 之設計者究竟如何達成上述目的。其實，這乃用盡 8080 所餘之空白運算碼，以及索引指令時再增加一位元組之運算碼的結果。後者使有些 Z80 指令長達五個位元組。

記得，任何程式都有好幾種不同之寫法，徹底地認識與了解整個指令集，乃是寫出有效率之程式所不可或缺。初學程式設計者並不一定須寫最精簡、有效率之程式。因此，初讀本章時，詳細記得各個指令並不重要。重要的是要記得指令之種類並且研習典型之例子。然後，於寫程式時，再參考 Z80 指令集之說明，選擇最適合自己所需之指令。本章，我們先以簡化的方式，歸納介紹 Z80 之各種指令型態。然後，再將指令集之每一指令一一詳細說明於後，讀者在設計程式時，即可參閱此一明細表。

下面，我們先依前一節之指令分類方式，將 Z80 指令集所含之指令種類分別描述於後。

4-3 資料傳送指令

於 Z80，資料傳送指令可分為四類：八位元傳送，十六位元傳送，堆疊作業，與區段傳送。茲將其分別介紹於後。

4-3-1 八位元傳送

於Z80，所有八位元之資訊傳送皆以取入（load）指令達成。取入的動作是將資料自某一來源位置傳至目的位置，而不改變來源位置之原有內含。取入指令之格式為

LD dst , src

其中，dst 表資料之目的地，其可為某一 CPU 暫存器或某一記憶位置。src 為資料來源，其亦可為某一 CPU 暫存器或某一記憶位置。

將資料取入累加器A 可使用暫存器，暫存器間接，索引，擴展，與立即等多種定址法。其指令格式分別為

LD	A, r	(暫存器)
	(HL), (BC), (DE)	(暫存器間接)
	(IX+d), (IY+d)	(索引)
	(nn)	(擴展)
	n	(立即)

將資料取入B, C, D, E, H, 或L等暫存器，則可使用暫存器，暫存器間接，索引，與立即等定址法。

$$\begin{aligned} & \text{LD} \quad \text{B}, \text{r} \\ & \quad (\text{HL}), \\ & \quad (\text{IX}+\text{d}), (\text{IY}+\text{d}) \\ & \quad \text{n} \end{aligned}$$

其中，r 代表A,B,C,D,E,H,或L 等任一暫存器，而 n 爲一八位

元立即值。爲了簡明起見，目的暫存器欄僅以B代表，其亦可爲C，D；E，H，或L。

將CPU暫存器內含“存出”至記憶器之指令，則只需將上述指令之目的欄與來源欄對調即可。當然，此時目的欄是不能放立即運算元的，其必須恒為一記憶位址。

附帶一提的，特殊暫存器 I 與 R 之內含皆可取入累加器（利用隱含定址）。同時，反之，累加器之內含亦可傳至 I 與 R。

下面，我們舉幾個實際例子。

1. LD B, A

爲一暫存器對暫存器傳輸指令。該指令將累加器A之內含抄入B暫存器。指令執行完後，累加器A之內含變成同於B暫存器之內含。於Z80，任何兩八位元之一般用途暫存器間，資料皆可直接傳送。

2 LD A, (1234H)

該指令將位址 1234（十六進制）之記憶位置的內含，取入累加器 A。注意該擴展定址指令之記憶位址的寫法。“1234H”表“十六進數 1234”。(1234H)表位址 1234 之記憶位置的“內含”。小括弧即表內含。該指令譯成機器碼儲存在記憶器中之情形為

位址	A	3A	運算碼
	A + 1	34	低次位址
	A + 2	12	高次位址

特別留意到，任何記憶位置之內含欲取入A以外之CPU 暫存器，都必須利用HL之暫存器間接定址，而不能如取入累加器A般地使用擴展定址。例如，若欲將位址25CD之記憶位置的內含取入B暫存器，則就必須先將25CD存入HL暫存器對（以十六位元取入指令），然後再執行

LD B, (HL)

指令。

3 LD C, 15H

該指令將立即數據——十六進數 15 (= 21₁₀) 取入 C 暫存器。

表 4-1 所示即為 Z 80 之八位元取入指令群

來源

		隱含		暫存器								暫存器間接			索引	擴展	立即
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n
暫存器	A	ED57	ED5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD7E d	FD7E d	3A nn	3E n
	B			47	40	41	42	43	44	45	46			DD46 d	FD46 d		06 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD4E d	FD4E d		0E n
	D			57	50	51	52	53	54	55	56			DD56 d	FD56 d		16 n
	E			5F	58	59	5A	5B	5C	5D	5E			DD5E d	FD5E d		1E n
	H			67	60	61	62	63	64	65	66			DD66 d	FD66 d		26 n
	L			6F	68	69	6A	6B	6C	6D	6E			DD6E d	FD6E d		2E n
暫存器間接	(HL)			77	70	71	72	73	74	75							36 n
	(BC)			02													
	(DE)			12													
索引	(IX+d)			DD77 d	DD70 d	DD71 d	DD72 d	DD73 d	DD74 d	DD75 d							DD36 d n
	(IY+d)			FD77 d	FD70 d	FD71 d	FD72 d	FD73 d	FD74 d	FD75 d							FD36 d n
擴展	(nn)			32 nn													
隱含	I			ED47													
	R			ED4F													

目的

4-3-2 十六位元傳送

基本上，我們可將一十六位元之立即數據，某兩連續記憶位置之內含（擴展定址），或堆疊頂端之資料（即堆疊指示器所指之位置內含），取入任一十六位元之暫存器對，BC, DE, HL, SP, IX, 或 IY。相反地，此些暫存器對之內含，亦可以相同方式存至某兩連續記憶位置或堆疊頂端。除此之外，HL, IX, 與 IY 等暫存器之內含，亦可取入 SP，此一能力簡便了多重堆疊器之製作。最後，AF

表 4-2 十六位元取入指令組與“推入”及“拉取”指令

來源

		暫存器							擴展立即	擴展	暫存器間接
		AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
暫存器	AF										F1
	BC								01 nn	ED4B nn	C1
	DE								11 nn	ED5B nn	D1
	HL								21 nn	ED2A nn	E1
	SP				FD nn	DD F9	FD F9		31 nn	ED7B nn	
	IX								DD21 nn	DD2A nn	DD E1
	IY								FD21 nn	FD2A nn	FD E1
擴展定址	(nn)		ED43 nn	ED53 nn	22 nn	ED73 nn	DD22 nn	FD22 nn			
推入指令	暫存器間接 (SP)		FD nn	DD nn	ED nn	DD nn	FD nn				

注意：每次執行後，推入與拉取指令均改變 SP 值

拉取指令

暫存器對之內含，亦可推入堆疊器，或將堆疊頂端資料取出存入 A F 暫存器。

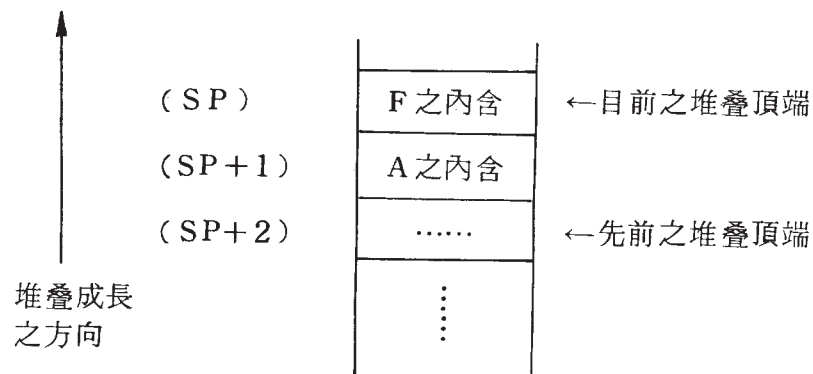
表 4-2 所示即為 Z80 之所有十六位元取入指令。堆疊作業之推入 (PUSH) 與拉取 (POP) 指令，亦算在此一群指令內。留意，Z 80 並無存取某一八位元暫存器之推入或拉取指令。所有堆疊作業指令均將資料來回傳遞於某一暫存器對與堆疊頂端之間。例如，

PUSH AF

指令，即為一將 A F 暫存器對之內含，存入堆疊器之單位元組指令，其運算碼為 F 5 H。該指令執行時所發生之作業系列為：

```
SP 值減 1
LD   (SP), A
SP 值再減 1
LD   (SP), F
```

指令執行過後，堆疊器之情形為



記得，於 Z80，堆疊指示器永遠指至堆疊頂端（即堆疊最頂端之一項資料）位置，因此，在推入之前，堆疊指示器之值必須先減一，以指至次一可供利用之位置。

拉取指令正巧與推入指令完全相反。例如，Z80 執行

POP AF

指令時，所發生之作業系列為：

```
LD   F, (SP)
SP 值加 1
LD   A, (SP)
SP 值再加 1 (指至新堆疊頂端)
```

注意，每拉取一個位元組後，堆疊指示器的內含值即自動加一，以指至新的堆疊頂端。

於堆疊作業時，運算元之十六位元中，高次之位元組恒先推入，後拉取。亦即：

```
PUSH DE 為 D 先推入，然後 E。
PUSH HL 為 H 先推入，然後 L。
POP  HL 為先拉取至 L，後 H。
```

最後，我們分別舉一擴展定址與擴展立即定址之十六位元取入指令為例。

擴展立即定址指令

LD IX, 0379 H

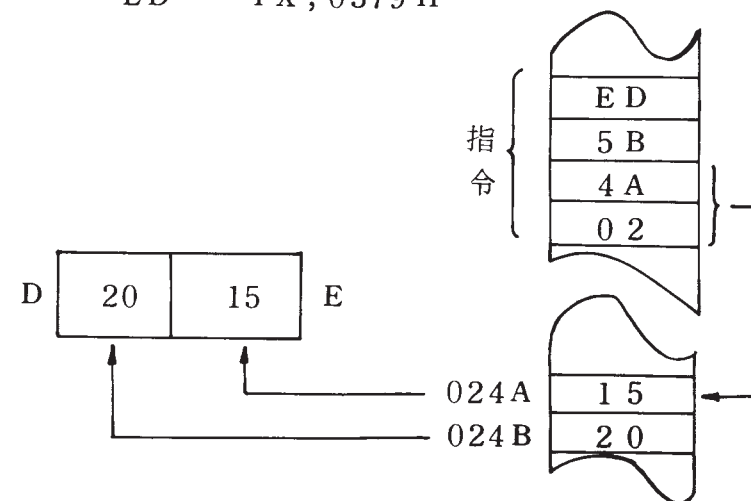


圖 4-3 擴展定址之十六位元“LD”指令

將十六位元之立即數據 0379 取入索引暫存器 IX。指令執行完後，IX 索引暫存器將含 0379（十六進）。

擴展定址指令

LD DE, (024AH)

將位址 024AH 之記憶位置的內含取入 E 暫存器，並且位址 024BH 之記憶位置的內含取入 D 暫存器。

特別注意，於 Z 80，擴展定址指令儲存於記憶器時，運算元位址之低次八位元永遠在前——位址較小之記憶位置，而高次八位元恒在後。此外，所有十六位元資料傳送指令之作業亦同。任何暫存器對（或十六位元暫存器）之內含只要儲存至記憶器，低次之八位元一定儲存在前（位址較小之記憶位置），而高次八位元一定在後。

4-3-3 交換指令

		隱含定址				
		AF'	BC', DE' & HL'	HL	IX	IY
隱含	AF	08				
	BC DE' & HL		D9			
	DE			EB		
暫存器 間接	(SP)			E3	DD E3	FD E3

表 4-3 交換指令 EX 與 EXX

表 4-3 所示即為 Z80 特有之交換指令。交換指令將兩特定位置之內含互換，因此，為一雙重資料傳輸。EX 指令能將堆疊頂端之資料（連續兩個位元組），與 HL，IX，或 IY 暫存器之內含互換。此外，其亦可將 DE 與 HL，或 AF 與 AF' 兩暫存器對之內含互換。

其次，EXX 指令將兩組六個一般用途暫存器之內含互換。

4-3-4 區段（整批）傳輸指令

區段傳輸指令導致整個記憶區段之資料（亦即，整批資料）的傳送，而非僅單位元組或雙位元組之資料。對微處理器之製造商而言，區段傳輸指令較難製作；但對程式設計者而言，其却甚為好用。此種指令增加程式設計之便利，並能提高程式之性能，尤以當輸入／輸出作業時為然。

Z 80 有四個區段傳輸指令：

LDI (LoaD and Increment)：取入，加一。

LDIR (LoaD, Increment and Repeat)：取入，加一，並重複。

LDD (LoaD and Decrement)：取入，減一。

LDDR (LoaD, Decrement and Repeat)：取入，減一，並重複。

所有區段傳輸指令皆使用三個暫存器對：

HL 指至資料之來源位置

DE 指至資料之目的位置

BC 為位元組計數器

程式設計者一旦將此些暫存器對佈置妥（置定其起始值），四個區段傳輸指令即可開始使用。LDI 指令將 HL 所指之記憶位置所含的位元組，移（抄）至 DE 所指之記憶位置。然後，HL 與 DE 兩暫存器對之內含均自動加一，指至次一位元組。同時，位元組計數器 (BC)

表 4-4 區段傳輸指令

目的	暫存器 間接	(DE)	來源	
			暫存器 間接	(HL)
			ED A 0	LDI— $((DE)) \leftarrow ((HL))$ HL 及 DE 各加一，BC 減一
			ED B 0	LDIR— $((DE)) \leftarrow ((HL))$ HL 及 DE 各加一，BC 減一， 重複至 BC = 0
			ED A 8	LDD— $((DE)) \leftarrow ((HL))$ HL 及 DE 各減一，BC 減一
			ED B 8	LDDR— $((DE)) \leftarrow ((HL))$, HL, DE, 及 BC 各減一，重複至 BC = 0

HL 指至來源位運

DE 指至目的位置

BC 為位元組計數器

之值亦自動減一。當有整批資料必須移動，而且每移動一者後必須作其它型式之處理時，此一指令甚為好用。LDIR 指令則為 LDI 指令之延伸。資料移動，指示器值加一，計數值減一之作業，一直重複，直至位元組計數值等於零為止。因之，僅此一指令即可將整批資料，自記憶器之某一區域搬至另一區域。圖 4-4 所示即為 LDIR 指令之執行情形。

留意，由於位元組計數器為一十六位元暫存器，因此，區段傳輸指令最高能搬運 64K (1K=1024) 個位元組之資料。同時，資料可由任一記憶位置移至任一其它記憶位置。再者，由於三個暫存器對

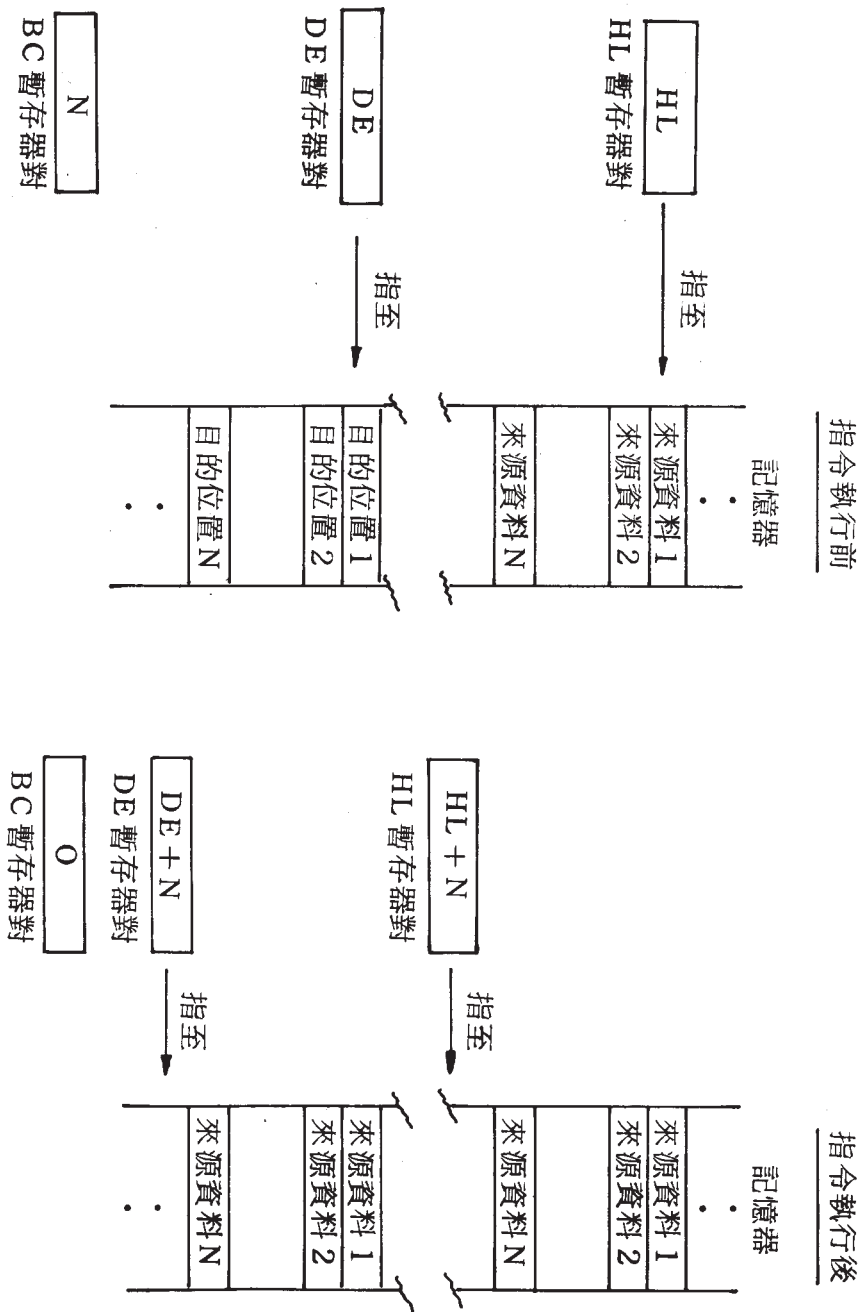


圖 4-4 LDIR 指令之效應

之內含並任何限制，因之，資料之來源區段與目的區段可彼此重疊。

LDD與LDDR指令則分別類似於LDI與LDIR指令。唯一不同的是，每次移動一個位元組後，HL與DE兩暫存器對之內含自動減一（而非加一），致使資料的搬運能自位址最高之位置開始，然後再位址漸小之位置。此一設施主要用於資料之來源區與目的區互相重疊時。於此種情況下，為免於來源區之資料於移走前先被破壞，資料搬運工作必須自記憶位置最高之位置開始，逐漸往位址漸小的方向進行。

4-3-5 區段搜尋指令

區段搜尋指令在此順便介紹。Z80有四個區段搜尋指令：

CPI (Compare and increment)：比較，並且加一。

CPIR (Compare, increment and repeat)：比較，加一，然後重複。

CPD (Compare and decrement)：比較，並且減一。

CPDR (Compare, decrement and repeat)：比較，減一，然後再重複。

CPI指令將累加器之內含，與HL暫存器所指之記憶位置的內含相比，比較結果存於其中一旗號位元。然後，HL暫存器對之內含加一，位元組計數器(BC)之值減一。

CPIR則是CPI指令之延伸。該指令即等於CPI指令之一直反復，直至找到欲搜取值或位元組計數值(BC之內含)為零時為止。因此，僅此一指令即可搜尋整個記憶器。

CPD與CPDR指令則分別類似於CPI與CPIR指令。唯一的差別是，每次搜尋後，HL暫存器之內含減一（而非加一），致使搜尋能反向進行（由記憶位址最高之位置開始）。

有一點要指出的是，區段傳輸與搜尋指令於文字串(string)處理應用上甚為有用。

表 4-5 區段搜尋指令

搜尋位置	
暫存器間接	
(HL)	
ED A1	'CPI'—比較 HL 加一，BC 減一
ED B1	'CPIR'—比較，HL 加一，BC 減一， 重複至 BC = 0 或找到為止
ED A9	'CPD'—比較，HL 及 BC 各減一
ED B9	'CPDR'—比較，HL 及 BC 各減一， 重複至 BC = 0 或找到為止

HL 指至欲與累加器內含相比之記憶位置。

BC 為位元組計數器。

4-4 資料處理指令

Z80之資料處理指令可分為三類來說：算術與邏輯指令，移位與旋轉指令，以及位元運作指令。

4-4-1 算術與邏輯指令

Z80之算術與邏輯指令可區分為三組：八位元算術與邏輯指令，十六位元算術與邏輯指令，與一般用途之AF作業指令。

八位元算術與邏輯指令用以加、減、AND、OR、exclusive OR、或比較兩個八位元之運算元，其中一運算元恒來自累加器。另

一運算元則可為立即數據，亦可來自某一 CPU 暫存器，或來自 HL 暫存器間接定址或索引定址所選取之記憶位置。兩運算元經運算後，最後之結果則存於累加器。因此，運算後，累加器之原有內含已遭破壞，但另一運算元則保持不變。

十六位元之算術指令履行兩十六位元暫存器間之算術（加或減）運算。加一與減一指令亦能對某一暫存器對運算。

另外，有五個一般用途之算術指令能對累加器或旗號暫存器運算。

一、算術指令

Z80 提供了兩種主要之算術運算指令：加與減。加法指令有一般加法（ADD）指令與進位加法（ADC）指令兩種。八位元之一般加法指令

ADD A, s

將累加器 A 之內含，加上第二運算元（以 s 代表），結果存回累加器。進位加法指令

ADC A, s

將累加器 A 之內含，加上第二運算元以及現有進位旗號值，並將結果存回累加器。

減法指令則類似於加法指令。八位元一般減法指令

SUB A, s

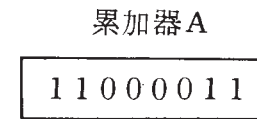
將累加器內含減去第二運算元（s），結果存回累加器。借位減法指令

SBC A, s

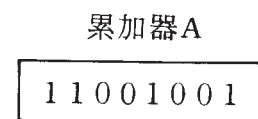
將累加器內含減去第二運算元及現有進位旗號值，結果存回累加器 A。八位元之算術運算指令如表 4-6 所示。這些指令可使用暫存器、暫存器間接、索引、與立即等定址法。

例如，若累加器之原有內含為 11000011（十六進數 C3），則立即定址之加法指令

ADD A, 6



(a) ADD A, 6 執行前



(b) ADD A, 6 執行後

圖 4-5 加法運算

執行完後，累加器內含將變為 11001001（十六進數 C9）。又例，若累加器內含原為 00100011（十六進數 23），B 暫存器之內含為 00110101（十六進數 35），且進位旗號為 1，則暫存器定址之加法指令

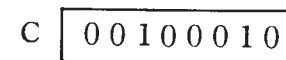
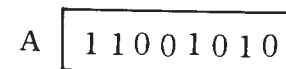
ADC A, B

執行完後，累加器內含將變為 01011001（十六進數 59），B 暫存器內含不變。

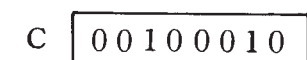
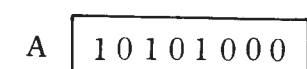
再舉一減法指令之例子。若累加器 A 之內含為 11001010（十六進數 CA），C 暫存器之內含為 00100010（十六進數 22），則暫存器定址之

SUB A, C

指令執行完後，累加器內含將變為 10101000（十六進數 A8），而 C 暫存器之內含維持不變。



(a) SUB A, C 執行前



(b) SUB A, C 執行後

圖 4-6 減法運算

表 4-6 八位元算術與邏輯運算指令

來 源

	暫存器定址							暫存器 間接	索 引		立即
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
進位加法 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
減算 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
進位減算 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
比較 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
加一 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
減一 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

表 4-7 一般用途之 AF 作業

十進調整 'DAA'	27
1 補數 'CPL'	2F
2 補數 'NEG'	ED 44
取進位旗號補數 'CCF'	3F
置定進位旗號 'SCF'	37

表 4-8 十六位元之算術運算指令

來源位置

		BC	DE	HL	SP	IX	IY
		HL	09	19	29	39	
'ADD'	IX	DD 09	DD 19		DD 39	DD 29	
	IY	FD 09	FD 19		FD 39		FD 29
進位加法且置定 旗號 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
進位減法且置定 旗號 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
加一 'INC'		03	13	23	33	DD 23	FD 23
減一 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

Z80 有兩個指令，INC (加一) 與 DEC (減一)，能將運算元之值加一或減一。八位元之 INC 與 DEC 指令能將任一八位元 CPU 暫存器，HL 暫存器間接定址，或索引定址所選取之記憶位置的內含加一或減一。十六位元之 INC 與 DEC 指令則能對任一十六位元暫存器作業。

表 4-7 中含有三個特殊之算術運算指令：DAA, CPL, 與 NEG。如圖 4-7 所示，CPL (ones-complement) 指令將累加器 A 之內含取 1 補數——所有的 0 變 1，1 變 0。NEG (NEGate) 指令將累加器 A 之內含取 1 補數，然後再加 1 (亦即取 2 補數)。其實際效應即將累加器所含之數目變號——正變負，負變正。十進制調整 DAA (Decimal Adjust Accumulator) 指令，主要則用於 BCD 加減算術。該指令通常緊接於 ADD, ADC, INC, SUB, SBC, DEC, 或 NEG

CPL

(A) 指令執行前	1 0 1 1 0 1 1 0	- 74
	1 變 0 ↓ 0 變 1	
(B) 指令執行後	0 1 0 0 1 0 0 1	+ 73

NEG

(A) 指令執行前	1 0 1 1 0 1 1 0	- 74
	1 變 0 ↓ 0 變 1	
	0 1 0 0 1 0 0 1	
		+ 1
	0 1 0 0 1 0 1 0	+ 74

圖 4-7 CPL 與 NEG 指令之例子

指令後，將運算所得之二進結果，調整成符合十進制 (BCD) 算術之結果。

二、邏輯指令

如表 4-6 所示，Z80 提供了三個邏輯運算指令，AND, OR, XOR (exclusive OR)，以及一比較指令 CP。此些指令皆對八位元資料運算，並且能使用暫存器，暫存器間接，索引，與立即等定址法。茲分別將其介紹如下：

AND

每一邏輯運算皆以一真值表 (truth table) 表示。真值表顯示了在每一種可能之輸入狀況下，邏輯運算結果之值。AND 運算之真值表為：

0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1

或

AND	0	1
0	0	0
1	0	1

AND 運算之特性為：當兩輸入均為 1 時，輸出 1，否則，輸出為 0。換言之，若其中有一位元為 0，則兩位元之 AND 運算結果必為 0。此一特性常用以將資料之某一 (或某些) 位元變為 0。此一技巧即稱為罩蓋 (masking) 或遮蓋。

AND 指令之最大用途即用以清除，或“遮蓋掉” (mask out) 某項資料之某些位元。譬如，若我們欲將 WORD 位置之內含的低次四位元清除為零，則下列三個指令即為我們所需。

LD A, (WORD) ; 資料字組取入 A。
AND 11110000B ; “11110000” 為面罩。
LD (WORD), A ; 結果存回原位置。

倘若位址WORD之記憶位址原來含10101010，則上述指令執行後，該位置之內含將變為10100000，最低次四位被清除為零。注意，“11110000B”即等於“二進數11110000”，B代表二進資料。

練習4—1：寫出將某一字組之第1與第6位元清除為零之三行指令。

練習4—2：若面罩為“11111111”，則結果如何？

OR

邏輯OR運算之真值表為

0 OR 0 = 0	或	OR	0	1
0 OR 1 = 1				
1 OR 0 = 1		0	0	1
1 OR 1 = 1		1	1	1

顯然，邏輯OR運算之特性為：若兩者中有任一者為1，則運算結果必為1。否則，若兩輸入皆為0，則運算結果為0。

OR指令最常用以將某一或某些位元置定為1。例如，

```
LD      A, (WORD)
OR      A, 00001111B
LD      (WORD), A
```

三個指令即將WORD記憶位置之內含的低次四位元置定為1。

練習4—3：立即定址之OR A, 10101111B 指令的效應為何？

練習4—4：OR A, FFH之效應為何？

練習4—5：您能以OR指令，將兩BCD數字“濃縮”合併於一個位元組嗎？

XOR

XOR代表“exclusive OR”。此一邏輯運算之真值表為

0 XOR 0 = 0	或	XOR	0	1
0 XOR 1 = 1				
1 XOR 0 = 1		0	0	1
1 XOR 1 = 0		1	1	0

顯然，XOR運算之特性為：若兩輸入有一者，而且僅有一者為1，則結果為1。否則，若兩輸入同為0或同為1，則運算結果為0。

XOR指令主要用於比較兩項資料是否完全相同。若兩項資料中有任一位元不同，則兩者XOR之結果必不為0。換言之，若兩字組XOR之結果為零，則兩字組必然完全相等。除此之外，於Z80，XOR指令亦可用以求某一資料字組之補數。利用XOR指令求補數之方法，是將其每一位元與“1”作XOR運算。例如，

```
LD      A, (WORD)
XOR     11111111B
```

若WORD位置原先含10101010，則上述指令執行後，累加器A之內含將變為01010101。是以，XOR指令可用以製作一“位元正反器”(bit toggle)，以作計時或為數2之計數器。下面之指令系列恰巧使由LOOP開始之迴路被執行兩次。

```
LD      A, 0           ; 計數為零。
LOOP    :
        (處理)
        :
        XOR 1           ; 2 數計數器。
        JP  NZ, LOOP
DONE    :
```

習題4—6：XOR 00000000B之效應為何？

4-4-2 移位與旋轉指令

Z 80 的主要能力之一，是能將累加器、任一般用途暫存器、或任一記憶位置（由暫存器間接定址或索引定址選取）之內含移位或旋轉。此些指令之運算碼如表 4-9 所示。表右邊之圖形解釋了各指令之功能。

Z 80 有七個移位與旋轉指令，以及兩個特殊之 BCD 數字旋轉指令。SLA (Shift Left Arithmetic, 算術左移) 與 SRL (Shift Right Logical, 邏輯右移) 指令，分別將運算元向左與向右移動一個位元位置。於此兩種運算，被移出字組之位元恒進入進位位元，而尾端空出之位元位置則補 0。

SRA (Shift Right Arithmetic, 算術右移) 則為一特殊之右移指令。於 2 補數形式之負數運算，最左（高次）位元為符號位元。負數時，該位元值為 1；正數時，該位元值為 0。當以右移方式將某負數除 2 時，符號位元必須保持 1，數目方能繼續保持於負數。SRL 指令顯然無法達成此一任務，因此必須採用 SRA 指令。於算術右移運算，進入最左邊位元者恒為**符號位元**。換言之，於連續右移之過程中，符號位元一直向右複製。例如，若累加器 A 原先之內含為

1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

- 124

則

SRA A

執行後，累加器內含變為

1	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---

- 62

注意，符號位元已向右複製。若再一次算術右移，則累加器內含將變為

1	1	1	0	0	0	0	1
---	---	---	---	---	---	---	---

- 31

表 4-9 移位與旋轉指令

來源目的		旋轉或移位	
A	16	16	16
RLC	07	07	07
RRC	0F	0F	0F
RL	17	17	17
RR	1F	1F	1F
SLA	27	27	27
SRA	2F	2F	2F
SRL	3F	3F	3F
RLD			
RAD			

A	07	17	1F
RLCA	0F		
RRCA			

圖 4-8 所示，即為上述介紹之三個移位指令運算的情形。

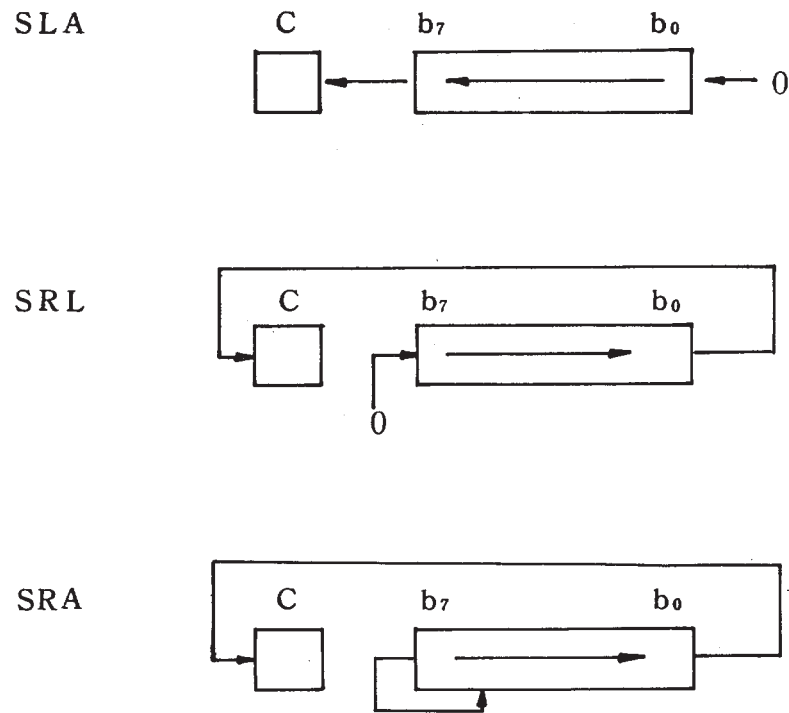


圖 4-8 移位指令

Z80 之旋轉運算有兩種：八位元旋轉與九位元旋轉。圖 4-9 所示即為九位元旋轉的情形。於此一運算，暫存器之八位元加上進位之第九位元，串聯成一位元串，同時向右或向左旋轉。換言之，移出資料字組之位元進入進位位元，而先前進位位元之值則進入資料字組之另一端。九位元旋轉之兩個指令為：RL (Rotate Left, 左旋轉) 與 RR (Rotate Right, 右旋轉)。例如，

RL (HL)

即將 HL 所指之記憶位置的內含以及進位，向左旋轉一位元位置，結果存回原記憶位置。

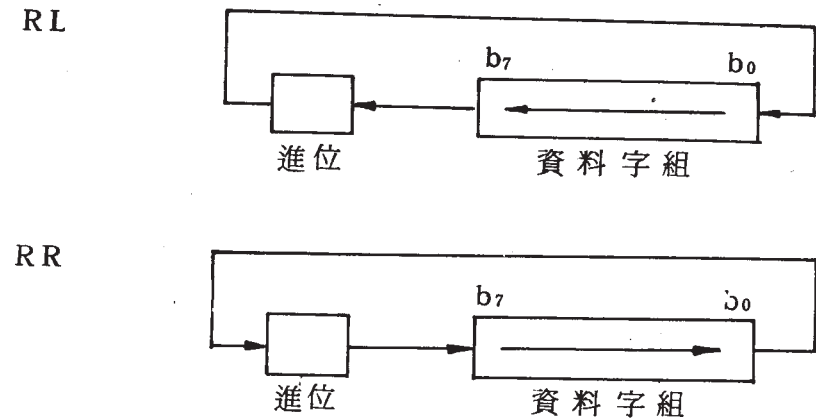


圖 4-9 九位元之旋轉

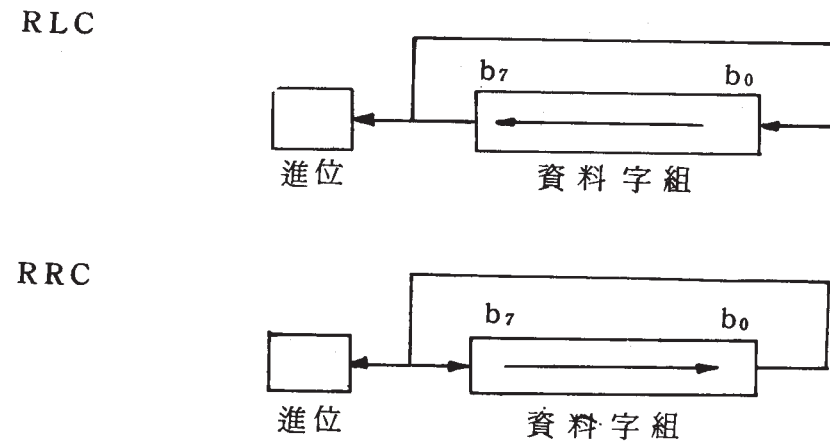


圖 4-10 八位元之旋轉

八位元之旋轉運算如圖 4-10 所示。左旋轉時，被移出第 7 位元位置之位元進入第 0 位元，同時亦進入進位位元。右旋轉時，移出第 0 位元位置之位元進入第 7 位元，同時亦進入進位位元。兩八位元之旋轉指令分別記為：RLC (Rotate Left Circular, 循環左旋轉) 以及 RRC (Rotate Right Circular, 循環右旋轉)。

習題 4-7：設 B 暫存器之內含為 10011100_2 ，進位旗號值為 1，試問，經上述之移位或旋轉運算後，B 暫存器之內含為何？

BCD 數字旋轉指令

Z80 有兩個特殊之 BCD 數字旋轉指令 (RRD 與 RLD)，以利 BCD 算術運算。如圖 4-11 所示，這兩指令導至 HL 暫存器指令及之記憶位置所含的兩位 BCD 數字，與累加器之低次四位元間的四位元旋轉。

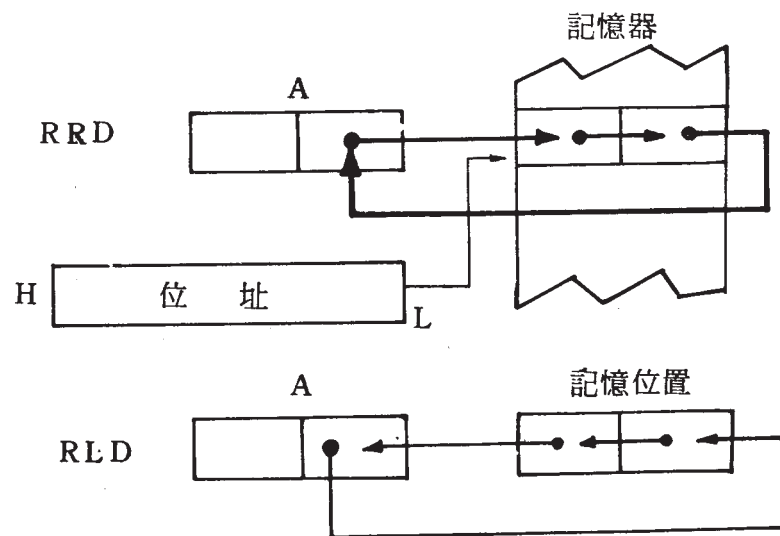


圖 4-11 BCD 數字旋轉指令

習題 4-8：若 HL 暫存器對之內含所指之記憶位置含濃縮 BCD 數 94，且累加器之低次四位元為 0010 (十進數 2)，則經 RRD 與 RLD 後，記憶位置之內含各如何？

4-4-3 位元運作指令

許多程式經常需置定，清除，或測試某一暫存器或某一記憶位置之某些位元。前面已提過，邏輯指令可用以將某一特定暫存器之某些位元置定為 1 或清除為 0。不過，若僅一個指令即能將任一暫存器或記憶位置之任意位元置定或清除，那將更方便了。由於這需相當多之運算碼，致一般微處理器均未具備。Z80 微處理器則例外，其獨具備極廣泛之位元運作指令。此些指令之運算碼如表 4-10 所示。(此一表所含之測試指令將於下節討論)。

於位元運作指令，暫存器定址可用以選取累加器或任一 CPU 一般用途暫存器。而記憶位置則可以 HL 暫存器間接或索引定址選取。

例如，

SET 7, B

指令即將 B 暫存器之第 7 (最高次) 位元置定為 1。

附帶一提的是，表 4-7 中有兩個指令能將進位旗號之值置定為 1，或將其互補。這兩指令分別是

SCF (Set Carry Flag)：進位旗號置定為 1。

CCF (Complement Carry Flag)：進位旗號之值互補 (0 變 1，1 變 0)。

4-5 測試與控制轉移指令

4-5-1 旗號

由於測試運算與旗號暫存器之應用關係密切，因此，在這兒，我們再詳細說明 Z80 之旗號暫存器內，每一旗號之功能與其受影響之情形。如圖 4-12 所示，Z80 之旗號暫存器包含六種旗號：S, Z, H, P/V, N, 與 C。除了 H 與 N 旗號是用於 BCD 算術而且不能測試外，其餘之四個旗號均可以條件跳越、條件叫用、或條件回返指令加以測試。

BIT	暫存器定址							暫存器 間接 索引	
	A	B	C	D	E	H	L	(HL)	(IX+d) (IY+d)
位元測試 BTI	0	CB 47	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	DD CB 4E
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	DD CB 4E
	2	CB 57	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	DD CB 5E
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	DD CB 5E
	4	CB 67	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	DD CB 6E
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	DD CB 6E
	6	CB 77	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	DD CB 7E
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	DD CB 7E
位元清除 RESET	0	CB 87	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	DD CB 8E
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	DD CB 8E
	2	CB 97	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	DD CB 9E
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	DD CB 9E
	4	CB A7	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	DD CB AE
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	DD CB AE
	6	CB B7	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	DD CB BE
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	DD CB BE
位元置定 SET	0	CB C7	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	DD CB CE
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	DD CB CE
	2	CB D7	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	DD CB DE
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	DD CB DE
	4	CB E7	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	DD CB EE
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	DD CB EE
	6	CB F7	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	DD CB FE
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	DD CB FE

表 4-10 位元運作指令組

進位旗號 (C)

幾乎於所有微處理器，包括 Z80 在內，進位旗號均具有雙重角色。其不僅用以顯示加或減運算是否產生進位（或借位），同時亦用以作為移位與旋轉運算之第九位元。此種雙重角色使得諸如乘法等之某些運算，變得相當容易。

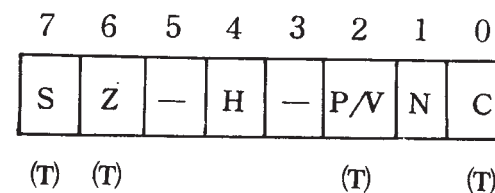


圖 4-12 旗號暫存器

關於進位旗號，有一點必須牢記的是，所有算術運算均影響進位位元；視運算之不同結果而定，該位元被置定為 1 或清除為 0。同樣地，所有移位與旋轉指令亦皆影響進位位元；視被移出運算元之位元值而定，進位旗號值被置定為 1 或清除為 0。於 Z80，所有的邏輯運算 (AND, OR, XOR) 恒使進位位元清除為 0。是以，此些指令可專門用以將進位旗號清除為零。

影響進位旗號值之指令可由表 4-11 查出。

減法旗號 (N)

此一旗號通常於 BCD 運算時由 Z80 自己使用，程式設計者無法加以利用。由於計算機永遠只能作二進制算術，因此，於 BCD 算術時，為使計算機運算之結果能合乎十進制算術，加或減指令後必須立即緊接一 DAA 指令，將運算結果調整成有效之 BCD 結果。然而，此一“調整”作業對加與減運算有所不同。N 旗號即應此一目的而生

表 4-11 Z80 旗號變化情形摘要

指 令	C	Z	P / V	S	N	H	註 解
ADD A,s; ADC A,s	↑	↑	V	↑	0	↑	八位元加算或進位加算
SUB s; SBC A,s, CP s, NEG	↑	↑	V	↑	1	↑	八位元減算、進位減算 、比較、與累加器值 2 補數。
AND s	0	↑	P	↑	0	1	邏輯運算
OR s; XOR s	0	↑	P	↑	0	0	
INC s	•	↑	V	↑	0	↑	八位元加一
DEC m	•	↑	V	↑	1	↑	八位元減一
ADD DD ,ss	↑	•	•	•	0	X	十六位元加算
ADC HL ,ss	↑	↑	V	↑	0	X	十六位元進位加算
SBC HL ,ss	↑	↑	V	↑	1	X	十六位元進位減算
RLA; RLCA, RRA, RRCA	↑	•	•	•	0	0	累加器旋轉
RL m; RLC m; RR m; RRC m; SLA m; SRA m; SRL m	↑	↑	P	↑	0	0	m 位置旋轉與移位
RLD , RRD	•	↑	P	↑	0	0	BCD 數字旋轉
DAA	↑	↑	P	↑	•	↑	十進制調整
CPL	•	•	•	•	1	1	累加器內含取 1 補數
SCF	1	•	•	•	0	0	置定進位旗號
CCF	↑	•	•	•	0	X	進位旗號反相
IN r, (C)	•	↑	P	↑	0	0	暫存器間接定址輸入
INI; IND; OUTI; OUTE	•	↓	•	X	↑	X	區段輸入與輸出， 若 $B \neq 0$ 則 $Z = 0$ ， 否則 $Z = 1$ 。
INIR; INDR; OTIR; OTDR	•	1	X	X	1	X	
LDI , LDD	•	X	↑	X	0	0	區段傳輸指令
LDIR , LDDR	•	X	0	X	0	0	若 $BC \neq 0$ 則 $P/V = 1$ ， 否則 $P/V = 0$ 。
CPI , CPIR, CPD , CPDR	•	↑	↑	↑	1	X	區段搜尋指令 若 $A = (HL)$ 則 $Z = 1$ ， 否則 $Z = 0$ 。若 $BC \neq 0$ 則 $P/V = 1$ ，否則 $P/V = 0$

LD A,I; LD A,R	•	↑	IFF	↑	0	0	插斷致能正反器 (IFF) 之內含抄入 P/V 旗號。
BIT b,s	•	↑	X	X	0	1	S 位置之第 b 位元狀態 抄入 Z 旗號。
NEG	↑	↑	V	↑	1	↑	累加器內含變號。

此一表格使用下列符號：

↑ 旗號受運算結果影響。

• 旗號不受運算影響。

0 運算恒將該旗號清除為 0。

1 運算恒將該旗號置定為 1。

X 該旗號無關。

V 運算之溢位結果影響該旗號。

P 運算之極性結果影響該旗號。

r 表 A, B, C, D, E, H, L 等任一暫存器。

S 指令可用之定址法選取之八位元位置。

SS 指令允許之定址法所選取之十六位元位置。

ii IX 或 IY 兩索引暫存器中任一者。

I 插斷向量暫存器。

R 記憶復新計數器。

n 0 至 255 範圍內之八位元值。

nm 0 至 65535 範圍內之十六位元值。

m 指令允許之定址法所選取之八位元位置。

，用以記錄 Z80 微處理器方才所執行的是加法或減法運算。於加法運算後，N 旗號被清除為 0，於減法運算後，N 旗號被置定為 1。（注意，有些其它微處理器以 N 代表負值旗號，切勿弄混了！）

許多指令執行之結果將 N 旗號值清除為 0，亦有許多指令執行之結果使 N 旗號置定為 1，此些指令可由表 4-11 查出。

極性 / 溢位旗號 (P / V)

基本上，極性 / 溢位旗號具有兩種不同之功能。於邏輯，移位與旋轉，DAA，及 $IN\ r, (C)$ 等運算時，該旗號為極性 (P) 旗號。於所有算術與比較指令時，該旗號則為溢位 (V) 旗號。

所謂極性 (parity)，即為某項資料所含之“1”位元的總個數。若總數為奇數，則稱奇極性；若總數為偶數，則稱偶極性。於作為極性旗號時，若指令運算所得結果為奇極性（結果所含“1”之總個數為奇數），則極性旗號值清除為0。若運算所得結果為偶極性，則極性旗號值置定為1。極性主要用於文數字 (alphanumeric) 資料之傳輸。於代表文數字之七位元電碼中加上一額外的極性位元，可偵測出資料在傳遞過程中是否發生錯誤。

於算術運算時，P/V 旗號主要作為溢位 (V) 旗號。該旗號用以顯示算術運算所得之結果，是否越過八位元 2 補數所能表示之最大範圍 (−128 至 +127)。若是，則溢位旗號置定為1；否則，其值為0。

除以上所述外，Z80 之 P/V 旗號尚有另外兩種功能。

於整批 (區段) 傳輸與搜尋指令時，該旗號用以顯示計數器暫存器 B 值之是否已為0。於遞減指令，若位元組計數器之暫存器對內含已遞減至零，則旗號值清除為0。於遞增指令，若指令開始前 $BC-1=0$ ，則旗號值置定為1。

此外，當執行 $LD\ A, I$ 與 $LD\ A, R$ 兩指令時，P/V 旗號則反映了插斷致能正反器 (IFF2) 之值。此一特色可用以保留或測試該位元值。

半進位旗號 (H)

於算術運算時，半進位旗號顯示第3位元 (b_3) 是否產生進位至第4位元 (b_4)。換言之，其顯示低次四位元是否有進位至高次四位元

。顯然，這是用於BCD 算術。特別，DAA 指令可用之作為調整運算結果之依據。

若加法運算時第3位元產生進位至第4位元，則半進位旗號值置定為1。否則，（若無進位）其值清除為0。反之，於減法運算，若第3位元向第4位元借位1，則H之值置定為1。否則，若無借位，則H旗號值清除為0。

半進位旗號受指令影響之情形如表4-11所示，注意，H位元並不受十六位元加與減指令之影響。

零值旗號 (Z)

零值旗號用以顯示剛剛被計算或傳輸之位元組是否為零。除此之外，其亦用以顯示比較運算是否已找到匹配（相同）值，以及若干其它之功用。

於資料計算與傳輸時，若最後導致之結果為零，則零值旗號置定為1；否則，零值旗號為0。於比較運算，若比較找到匹配值（即比較成功），則零值旗號亦置定為1；否則，其值為零。

除此之外，於Z80，零值旗號尚有三種其它功能：第一，其與BIT 指令併用，以顯示被測試之位元值是否為零。若是，則其值被置定為1；否則，其值為0。第二，於整批輸入 / 輸出指令 ($INI, IND, OUTI, OUTD$)，若 $D-1=0$ ，則Z旗號置定為1；否則，其值清除為0；若位元組計數器將遞減成0 ($INIR, INDR, OTIR, OTDR$)，其值置定為1。第三，於特殊指令 $IN\ r, (C)$ ，零值旗號被置定為1，以顯示輸入位元組之值為零。

正負數旗號 (S)

該旗號反映指令運算結果或被傳遞位元組之最高次位元值（第7位元）。於2補數數目表示法，最高次位元代表正負號：“0”表正數，“1”表負數。因此，第7位元又稱為符號（正負號）位元。

在大多數微處理器，符號位元於輸入／輸出設備溝通時，扮演了相當重要之角色。由於大多數微處理器均未具備能測試某一暫存器或記憶位置之任意位元的指令。以致，符號位元變成一最容易測試之位元。在讀取某一輸入／輸出設備之狀態資訊，以檢查其狀態時，我們可將狀態位元置於狀態暫存器之最高次位元。如此，程式即能輕易測試狀態位元之值。此即為為何大多數連至微處理器之輸入／輸出設備的最重要狀態位元，均設於狀態暫存器之第 7 位元的原故。

旗號摘要

微處理器之旗號位元用以自動偵測算術／邏輯單元之某些特殊狀態。此些旗號位元能方便地以各種條件式的控制轉移指令加以測試，以便程式能依據測試結果，決定進一步應採取何種反應。由於程式內所發生之決策大多根據此些旗號。因此，程式設計者必須熟悉每一旗號之角色，俾能善以利用。目前，您僅需記住每一旗號之主要功能即可。在實際從事程式設計時，若遇有不解之處，再回頭逐一查閱。

Z80 各旗號位元受指令執行影響的情形，特摘要於表 4-11，以便參閱。除了此些各種指令之運算會自動影響旗號之值外，Z80 尚具有一特殊之位元測試指令 BIT。該指令能用以測試任一 CPU 暫存器或記憶位置之任一位元值，然後影響某些旗號值，以作為緊接條件式控制轉移指令之判斷依據。位元測試之運算並不影響被測試位元之原有內含值。

4-5-2 控制轉移指令

本章第一節曾經說過，程式控制轉移指令改變程式之正常（循序）執行次序，而令控制轉移至儲存於其它記憶區之程式指令。換言之，此種指令執行之結果，將一新數值存入程式計數器，使次一被執行之指令來自其它記憶位置，而非立即緊接目前指令後之指令。Z80 之程式控制指令可分成跳越（jump），副程式叫用（call），與回返（

表 4-12 JUMP, CALL, 與 RETURN 指令

			條 件									
			無條件	有進位	無進位	零值	非零	偶極性	奇極性	負數	正數	REG
JUMP 'JP'	擴展立即	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JR'	相對	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP 'JP'	暫存器 間 接	(HL)	E9									
JUMP 'JP'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
'CALL'	擴展立即	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
B 減一，非 零則再跳 越 'DJNZ'	相對	PC+e										10 e-2
回 返 'RET'	暫存器 間 接	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
插斷回返 'RETI'	暫存器 間 接	(SP) (SP+1)	ED 4D									
不可罩蓋 插斷回返 'RETN'	暫存器 間 接	(SP) (SP+1)	ED 45									

注意：某些旗號有多種用途。

表 4-13 重始指令

		運算碼	
叫 用 位 址	0000 _H	C7	'RST 0'
	0008 _H	CF	'RST 8'
	0010 _H	D7	'RST 16'
	0018 _H	DF	'RST 24'
	0020 _H	E7	'RST 32'
	0028 _H	EF	'RST 40'
	0030 _H	F7	'RST 48'
	0038 _H	FF	'RST 56'

return) 三種。**跳越**指令將程式控制轉移至另一記憶位置，但並不將程式計數器之內含存起。**叫用**指令同樣將程式控制轉移至另一記憶位置，但却將程式計數器之內含推入堆疊器存起，以便副程式結束後能回返。**回返**指令則自堆疊器拉取先前存起之程式計數器內含，將程式控制傳回至緊接叫用指令後之指令。叫用與回返指令皆用於副程式之處理。

跳越指令

Z80之跳越指令有無條件跳越與條件跳越兩種。無條件跳越指令可使用擴展立即，相對，與暫存器間接等三種定址，其指令格式分別為

擴展立即： JP nn nn 為一十六位元位址。
相 對： JR e e 為八位元之位移(2補數形式)。
暫存器間接： JP (HL)
(IX)

(IY)

例如，若 Z80 微處理器執行 JP nn 指令，則該指令執行完後，緊接被執行的指令將來自位址 nn 之記憶位置。

條件跳越指令是當所指明之條件滿足時，跳越才發生；而條件不滿足時，微處理器繼續執行次一緊接指令之跳越指令。此種指令除必須指明跳越之目的位址外，尚須指明一欲測試之條件(或狀態)，致其格式為

JP C, nn

其中，C 即為所指明之條件，而 nn 為跳越之目的位址。於 Z80，條件跳越指令可使用相對與擴展立即兩種定址法。其指令格式分別為

擴展立即： JP C, nn

相 對： JR C, e

其中，擴展立即定址之條件跳越指令可測試 Z, C, P/V，與 S 等四種旗號，而相對定址之條件跳越指令則僅能測試 Z 與 C 兩種旗號。由於每一旗號有兩種可能狀態，故擴展立即定址之

JP C, nn

指令中，陳述條件 C 可為下列八種情況之任一種：

Z : 零值 (Z = 1)

NZ : 非零值 (Z = 0)

C : 有進位 (C = 1)

NC : 無進位 (C = 0)

PO : 奇極性

PE : 偶極性

P : 正值 (S = 0)

M : 負值 (S = 1)

。而相對定址

JR C, e

指令中之 C，則僅能為 Z, NZ, C, 與 NC 四種情況之任一種。

例如，

JP NZ, LOOP

指令即為一擴展立即定址之條件跳越指令。當 Z80 微處理器執行此一指令時，若零值旗號之值為 0，則程式控制將跳至儲存於位址 LOOP 之記憶位置上的指令。否則，若零值旗號為 1（表前一運算之結果為零），則微處理器繼續執行次一緊接指令。

注意無條件跳越指令與條件跳越指令之格式差異：條件跳越指令上多一“陳述條件” C。

於運算碼寫碼時，上述所列之條件的寫碼情形為：

條件碼	條 件
0 0 0	NZ
0 0 1	Z
0 1 0	NC
0 1 1	C
1 0 0	PO
1 0 1	PE
1 1 0	P
1 1 1	M

圖 4-13 測試條件（狀態）之寫碼

Z80 有一特殊之條件跳越指令：DJNZ。該指令將 B 暫存器之內含值減一，若結果不為零，則跳越發生；否則，若 B 暫存器之值為零，則跳越不發生。此一指令主要用於**程式迴路**（program loop）之製作，其僅能使用相對定址。

副程式叫用指令

Z80 之副程式叫用指令亦有條件式與無條件式之分。該指令只能使用擴展立即定址，其格式為

無條件叫用：CALL nn

條件叫用：CALL C, nn

其中，陳述條件 C 如擴展立即定址之條件跳越指令。微處理器執行 CALL nn 指令時，首先將現有程式計數器內含推入堆疊器存起，然後將 nn 存入程式計數器，使次一被執行之指令來自位址 nn 之記憶位置。

回返指令

同樣地，Z80 之副程式回返指令（RET）亦有條件與無條件式之分。RET 指令所能測試之條件同於條件副程式叫用指令。微處理器執行回返指令時，即將堆疊頂端連續兩位元組之資料，拉回至程式計數器。使控制回至緊接叫用指令後之指令。副程式叫用與回返指令之功能如圖 4-14 所示。

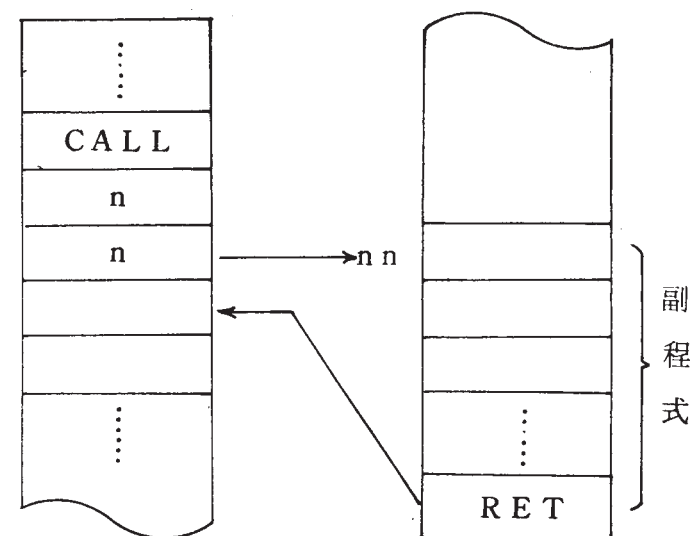


圖 4-14 副程式叫用與回返指令

Z80另有兩個特殊之回返指令。RETI 指令用於可罩蓋插斷之回返，RETN 指令則用於不可罩蓋插斷(nonmaskable interrupt)之回返。此兩指令在後面討論插斷時，會再作更進一步之解釋。

重始指令

Z80具有另一種特殊程式控制轉移指令——重始指令RST。RST 指令有兩個主要用途：第一，插斷設備用其將八位元之插斷向量置於資料巴士上。第二，其令控制能跳至八個零頁位置中之任一個。用以叫用零頁位置之重始指令的格式為RST P，其中P為跳越之目的零頁位址。RST指令之運算碼寫碼情形為

RST P



表 4-13 所示即為 RST 指令之運算碼情形

P	T 欄
00 H	000
08 H	001
10 H	010
18 H	011
20 H	100
28 H	101
30 H	110
38 H	111

表 4-13 RST 指令之運算碼寫碼

4-6 輸入 / 輸出指令

簡言之，輸入 / 輸出設備可以兩種方式選取：將之當成記憶位置

，以前面介紹過之任一指令存取，或以特定之輸入 / 輸出指令存取。尋常之記憶選取指令必須使用三個位元組：運算碼一個位元組，位址兩個位元組。結果，由於指令執行需作三次記憶器存取，因此，速度相當緩慢。特定輸入 / 輸出指令之主要目的即在於提供較短，因此亦較快之指令。但是，輸入 / 輸出指令亦有兩個缺點。

首先，其浪費了珍貴有限之運算碼寫碼空間（通常微處理器僅以八位元作運算碼之寫碼）。其次，其必須產生一個或一個以上之特別輸入 / 輸出信號，致浪費了微處理器之有限接腳。由於具有此些缺點，致一般微處理器皆無特別之輸入 / 輸出指令。不過，由於最原始之8080微處理器（世界上第一個能幹之八位元一般用途微處理器）上有，故與之匹配之Z80上亦有。

固然輸入 / 輸出指令執行上較迅速，因其僅長兩個位元組，但零頁定址亦可達成同樣效果。致某些微處理器採取了此一方式。

兩個基本之輸入 / 輸出指令為IN與OUT。IN指令將資料自某一特定輸入 / 輸出位置讀進某一CPU暫存器，OUT指令則將某一CPU暫存器之資料輸出至某一輸入 / 輸出設備。Z80之輸入 / 輸出有兩種定址方式。立即定址之輸入 / 輸出指令僅能將資料來回傳遞於累加器A與輸入 / 輸出設備之間，暫存器間接定址之輸入 / 輸出指令則能將資料來回傳遞於任一一般用途暫存器與輸入 / 輸出設備間。其指令格式分別為

立即： IN A, (n)
 OUT (n), A
 暫存器間接： IN r, (C)
 OUT (C), r

其中，n為一八位元立即位址，C為C暫存器，r表任一Z80之八位元一般用途暫存器。

於執行立即定址之輸入 / 輸出指令時，輸入 / 輸出設備之位址n，出現於位址巴士之低次八位元(A₀~A₇)，而累加器之內含則被置

表 4-14 輸入指令

		資料來源之 輸入口位置		
輸入 目的位置		立即	暫存器間接	
		(n)	(c)	
輸入 'IN'	暫存器定址	A	DB n	ED 78
		B		ED 40
		C		ED 48
		D		ED 50
		E		ED 58
		H		ED 60
		L		ED 68
'INI'—輸入， HL 加一，B 減一。	暫存器間接			ED A2
'INIR'—輸入， HL 加一，B 減一， 若 B ≠ 0、再重複。		(HL)		ED B2
'IND'—輸入， HL 及 B 各減一。				ED AA
'INDR'—輸入， HL 及 B 各減一， 若 B ≠ 0、再重複。				ED BA

整批輸入指令

表 4-15 輸出指令

		來源		暫存器								暫存器間接
				A	B	C	D	E	H	L	(HL)	
'OUT'	立即	(n)	D3 _n									
	暫存器間接	(c)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69			
輸出，HL 加一， B 減一。		(c)										ED A3
'OTIR' - 輸出 HL 加一，B 減一 若 B ≠ 0，再重複。		(c)										ED B3
輸出，HL 及 B 各減一。		(c)										ED AB
'OTDR' - 輸出 HL 與 B 各減一， 若 B ≠ 0，再重複。		(c)										ED BB
		整批輸出指令										
		目的輸出口位址										

於位址巴士之高次八位元 (A₈~A₁₅)。於暫存器間接定址之輸入 / 輸出指令時，C 暫存器之內含出現於位址巴士之低次八位元，而 B 暫存器之內含則被置於位址巴士之高次八位元。如表 4-14，4-15 所示，上述之輸入 / 輸出指令均長兩位元組，並能選取 256 個可能輸入 / 輸出位置。

除了上述能傳輸八位元資料之輸入 / 輸出指令外，Z80 亦提供了

整批傳輸之輸入／輸出指令。四個整批傳輸之輸入指令為：INI，INIR，IND，與 INDR。同樣地，四個整批傳輸之輸出指令為：OUTI，OUTIR，OUTD，與 OUTDR。於此些自動之整批傳輸，HL 暫存器對均作為目的區指示器，暫存器C作為輸入／輸出設備之選擇（由256個設備中選取一個），而B暫存器則作為計數器，其內含值可加一或減一。

INI 為自動之單位元組傳輸。暫存器C之內含選取某一輸入設備。資料位元組然後由該設備讀取，並傳至HL所指之記憶位置。之後，HL之內含值加1，計數器B之內含值減1。

INIR則為INI之重複執行。重複程序一直進行至計數器B之值，遞減至零時為止。因此，該指令最高能自動傳輸256個位元組。注意，欲作正巧256個位元組之傳輸，B暫存器之值必須先置定為0。

輸入與輸出指令之運算碼分別如表4-14與4-15所示。

4-7 各種CPU控制指令

控制指令乃改變CPU之作業型態或經繕(manipulate)CPU之內部狀態資訊的指令。Z80具有七個此種指令。

NOP 指令為**無運算**指令，於此一指令之機器週期（僅一個）期間，Z80 CPU什麼事都沒做。典型上，該指令有兩種用途：用以**產生延遲**（4個T週期=2 us，就2MHz之時序言），或**填補偵錯結果所造成之程式空缺**。傳統上，NOP指令之運算碼為所有位元皆0。此乃因為程式執行期間，記憶器通常清除為0。執行NOP指令可確保程式不受損壞，且程式執行不致停止。

HALT 指令用於插斷或重置。其延遲了CPU之作業。當Z80微處理器因執行HALT指令而處於**停止狀態**時，插斷或重置信號可用以令其重新恢復作業。於停止(HALT)狀態期間，Z80微處理器一直執行NOP指令。於偵錯期間，程式之最後指令經常為HALT指令。

表4-16 各種CPU控制指令

'NOP'	00
'HALT'	76
INT 禁能 'DI'	F3
INT 致能 'EI'	FB
設定插斷型態 0 'IM0'	ED 46
設定插斷型態 1 'IM1'	ED 56
設定插斷型態 2 'IM2'	ED 5E

8080A 型

叫用 0038H 位置

以 I 暫存器及插斷設備來之八位元作指示器之間接叫用。

EI與**DI**指令則分別用以將Z80 CPU內之插斷旗號置定為1與清除為0，令外部插斷分別致能與禁能。插斷在後面會有更詳盡之討論。

最後，Z80提供了三種插斷型態（8080僅有一種）。插斷型態0為8080型態，插斷型態1叫用位址0038H記憶位置之副程式，插斷型態2則為間接副程式叫用，此一型態以特殊暫存器I之內含，加上插斷設備所提供之八位元，作為指示器，指至記憶器中之插斷處理常式。此些型態於後面會更加詳細討論。

摘 要

至此，我們已將Z80微處理器之指令集，分類作了一概要介紹。本章緊接之篇幅，即對此些指令再詳細個別加以介紹。介紹之內容包括指令功能、格式、運算說明、資料流程、時序、定址法、運算碼、影響旗號等，最後，並舉一實例加以說明。其中，各段落說明之意義如下：

功 能：為指令功能或運算之圖解或符號說明。

格 式：為指令之機器碼形式。依位元組次序列出。

說 明：指令功能（運算情形）之進一步文字說明。

資料流程：指令被執行時，資料（或稱數據）的流動情形。

時 序：指令所需之執行時間，以M及T週期數表示，並以
2 MHz 之時序頻率舉例。

運 算 碼：運算碼中，含暫存器碼或位元碼之位元組的進一步說明。

對初學者而言，一開始時並不須將每一指令徹底熟記，而只須記得每一指令種類之幾個主要指令就夠了。畢竟，學習程式設計最迅速有效之辦法，乃為**做中學**——多寫程式，需用到什麼指令而不明瞭時，多查表格資料，進步就會很神速。最後，自然而然，你就會記得且熟悉整個指令集之每一指令——這是當您自行設計程式，且欲設計出好的程式所必備的！

為了使您由實際程式設計過程中達熟練之境界，下面幾章，我們開始介紹各種程式之寫作。

習題解答

4—1 LD A, (WORD)
AND 10111101B
LD (WORD), A

4—2 AND運算後結果不變。

4—3 將累加器第4與第6位元外之所有位元置定為1，前述兩位元值保持不變。

4—4 將累加器A之內含變為FF。

4—5 若兩BCD數字分別儲存於A與B暫存器之低次四位元，B為高次數字，則下列指令系列將兩數字“濃縮”，結果

存於A。

SLA B
SLA B
SLA B
SLA B
OR B

4—6 運算結果不變。

4—7 SLA: 00111000
SRL: 01001110
SRA: 11001110
RL: 00111001
RR: 11001110
RLC: 00111001
RRC: 01001110

4—8 RRD後: ((HL)) = 29
RLD後: ((HL)) = 42

文件名稱： Z80 微電腦軟體硬體第 3 及 4 章前半部份

文件分類	I
文件編號	00028
文件批號	02

製作群	原稿掃圖	文稿編輯
	原稿圖文分離	文稿整合
	原稿辨識	文稿校對
	文稿成品輸出	特別感謝名單

文件完成日期	初版	2007-02-08	其他
	再版		加註

文件出處	原圖書書名	Z80 微電腦軟體硬體
	原圖書作者	陳金迫
	原圖書出版者	儒林圖書有限公司
	原圖書出版日	民國 70 年 8 月

DDSC 文件版權宣告	本文件版權屬原輸出公司、出版社、圖書公司或原著作人所有，作商業用途者請自行洽上述公司，本文件僅可在非商業上流傳或供私人收集資料用。另由於資料老舊 DDSC 不對原書內的內容負責，且除了更正原書內的錯字、漏字之外一切照原書內容所用的文字顯示。
-------------	---

檔名格式說明：

DDSC — 文件分類 — 文件編號 — 文件批號 — 文件名.PDF

以 DDSC 為起頭，加上 1 個字母為分類代碼，再加上以 5 位數由 00001 起的編號，加上 2 位數由 01 起的編號，加上完整的文件名稱而成的。

其中分類代碼詳見下面列表。文件批號指該文件為非合訂版的，可能因書的內容過多而分批完成的，此項可有可無。

文件分類代碼說明	
代 碼	說 明
A	小說／文學類文章類
B	娛樂類
C	天文類
D	科學類
E	古文明事物類
F	自然界類
G	古怪事物類
H	動／植物類
I	電子類
J	電腦類
K	教育教學類