CHAPTER 4

程式指令動作

單元目標

4-1 組合語言指令格式

組合語言在撰寫中，對CPU來說，電腦是不懂我們所寫的程式，只能讀取程式轉換之機械碼(Machine Code)。但我們在程式之撰寫上，亦須瞭解其格式，現將敘述如下：其實組合語言是由下列四個部份所組成。

1. 標記欄
2. 運算欄
3. 運算元欄
4. 註解欄

其型式如：

START：	MOV	RO,#00H；	設定初始值
↑	↑	↑	↑
標記欄	運算欄	運算元欄	註解欄

各欄位解說如下：

◇ 標記欄(Label field)

這是由設計者自己定的符號標記，以此代表指令的位址、資料存放地方、副程式的位址等。撰寫時要注意的細節爲：

1. 整個字元，一律用大寫英文字母，不可大小寫字母混在一起。
2. 長度不受限制，但以不超過 32 個位元爲止。
3. 開頭的第一個字爲文字。
4. 標記以(：)冒號作結束時，標記可任一行開始；反之若沒有(：)冒號結束之標記，則一定要在第一行開始寫。如下圖 4-1 所示。

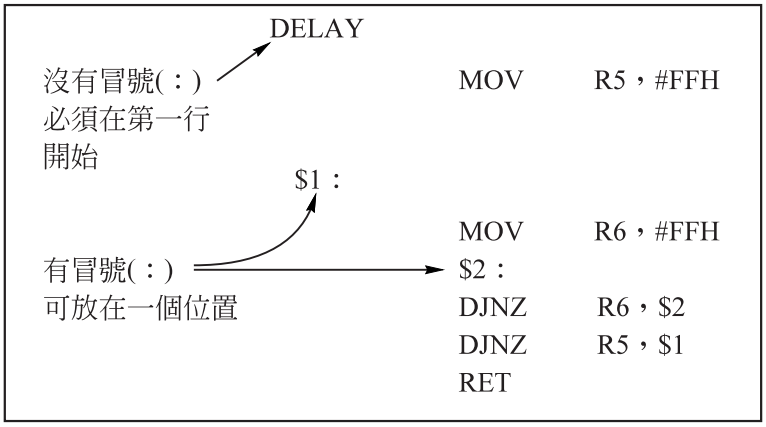


圖 4-1

◇ 運算欄(Operation)

此欄位是指令之助憶符號，一定要放在標記欄的後面，但前面若沒有標記欄出現，可放在第二行後的位置。通常是指出CPU的動作，且爲單字之縮寫；如資料用“MOV”，加法指令用“ADD”，減法指令用“DEC”……等。

◇ 運算元欄(Operands)

利用此欄位來存放指令作運算時所要的資料，且是固定放在運算欄後面，但須留意在運算欄與運算元欄之間，須用空白字元來隔開。

在撰寫運算元時，有2個以上運算元，則要用逗點(，)隔開。如：

```
CJNE  A,#20H          ；三個運算元 A,#20H,NEXT
DEC   A,B             ；二個運算元 A,B
INC   A               ；一個運算元 A
RET                   ；無運算元
```

◇ 註解欄(Comment)

此欄專爲文字說明存放地，可幫助設計者容易瞭解程式之內容，且是以分號(；)爲開頭，在分號後之文字即爲註解，而組譯程式不會對此加以組譯。

4-2 定址模式

由運算元提供CPU運算執行所要的資料，此方式便是定址模式。其定址方式有五種：

1. 直接定址法(Direct Addressing)

   可在指令中直接指出資料所在位址，其位址是爲內部資料記憶體位址由00H～FFH。指令型式如下：

```
MOV  A,10H          ;將位址 10H 之內容存進 Acc 累加器中
MOV  A,PSW          ;將 PSW 的內容值搬移至 Acc 累加器中
MOV  C,10H          ;將位元位址 10H 的值放入進位旗號 C 中
```

2. 立即定址法(Immediate Addressing)

在運算元欄中以資料常數表示，且在常數前要加一個“#”字符號。如：

```
MOV  A,#80H         ;將常數資料值 80H 放入 Acc 累加器中
MOV  DPTR,#data     ;將常數資料 data 值放入 DPTR 暫存器
```

3. 暫存器定址法(Register Addressing)

即是作暫存器對暫存器之間的資料傳送。且在8051的內部RAM中有4組暫存器庫(BK0～BANK3)，每一組暫存器皆含有8個暫存器(R0～R7)。如：

```
MOV  A,R0           ;將暫存器 R0 資料值移入 Acc 累積器中
MOV  R0,A           ;將 Acc 累積器值放入暫存器 R0 中
```

4. 暫存器間接定址法(Register Indirect Addressing)

是以特定的暫存器值爲位址，再以此位址移入A_{cc}累加器中，作爲運算操作對象。且其前置符號爲“@”。

在 R0，R1 執行暫存器間接定址時，其範圍限於 256 Bytes 之內，位址爲 00H～FFH。但若以 DPTR 執行暫存器間接定址時，則適用於外部資料記憶體位址 0000H～FFFFH，即定址在2^{16}＝65536 Bytes。如：

```
MOV  R0,#30H
MOV  A,@R0      ;內部資料記憶體位址 30H 值移入到 Acc 累加器中
MOVX A,@DPTR    ;外部資料記憶體 DPTR 值移入 Acc 累加器中
```

5. 索引定址法(Indirect Addressing)

是基底暫存器加索引暫存器間接定址之內容值的和爲位址簡稱。一般此定址是用在程式記憶體之內容，與程式分支指令群的跳躍(JUMP)指令中。如：

```
MOVC  A,@A+DPTR    ;DPTR 值＋ Acc 之內容值的資料和，再存入
                    Acc 累積器
MOVC  A,@A+PC
```

4-3 指令的動作

指令集共有 111 個指令，依其功能可分爲五大類：

1. 資料轉移指令
2. 邏輯運算指令
3. 算術運算指令
4. 布林運算指令
5. 程式分支指令

◇ 資料轉移指令

如下表 4-1 是爲資料轉移指令之整理與說明，功用主要將來源的資料移到目的地，且不會改變資料來源之內容。

表 4-1 資料轉移指令表

指令	內容說明	長度	機械週期
MOV A,Rn	將暫存器 Rn 值移至 Acc 累積器	1	1
MOV A,direct	將直接(direct)位址值移至 Acc 累積器	2	1
MOV A,@Ri	暫存器Ri的內部資料記憶體值移至Acc累積器	1	1
MOV A,#data	常數資料 data 值移至 Acc 累積器	2	1

表 4-1 資料轉移指令表(續)

指令	內容說明	長度	機械週期
MOV Rn,A	Acc 累積器值移至暫存器 Rn	1	1
MOV Rn,direct	直接(direct)位址值移至暫存器 Rn	2	2
MOV Rn,#data	常數資料 data 值移至暫存器 Rn	2	1
MOV direct,A	累積器 Acc 值移至直接(direct)位址上	2	1
MOV direct,Rn	暫存器 Rn 值移至直接(direct)位址上	2	2
MOV direct,direct	直接(direct)位址值移至直接(direct)位址上	3	2
MOV direct,@Ri	暫存器 Ri 的內部資料記憶體值移至直接(direct)位址上	2	2
MOV direct,#data	常數資料data值移至直接(direct)位址上	3	2
MOV @Ri,A	累積器 Acc 值移至暫存器 Ri 的內部資料記憶體位址上	1	1
MOV @Ri,direct	直接(direct)位址值移至暫存器 Ri 的內部資料記憶體位址上	2	2
MOV @Ri,#data	常數資料 data 值移至暫存器 Ri 的內部資料記憶體位址上	2	1
MOV DPTR,#data 16	16 位元常數資料 data 值移至暫存器 DPTR 上	3	2
MOV A,@A+DPTR	累積器 Acc 與 DPTR 暫存器值之和移至累積器 Acc	1	2
MOV A,@A+PC	PC 值加 1 後再與累積器 Acc 值之和移至累積器 Acc	1	2
MOV A,@Ri	暫存器 Ri 的外部資料記憶體位址值移至累積器 Acc	1	2

表 4-1 資料轉移指令表(續)

指令	內容說明	長度	機械週期
MOV A,@DPTR	DPTR的外部資料記憶體位址移至累積器 Acc	1	2
MOVX @Ri,A	累積器 Acc 值移至暫存器 Ri 之外部資料記憶體位址值上	1	2
MOVX @DPTR,A	累積器 Acc 值移至暫存器 DPTR 之外部資料記憶體位址值上	1	2
PUSH direct	直接(direct)位址存至 SP 堆疊內(在 SP 值加 1 後)	2	2
POP direct	由 SP 上取出資料值，再存至直接(direct)位址上	2	2
XCH A,Rn	互換累積器 Acc 與暫存器 Rn 值	1	1
XCH A,direct	互換累積器 Acc 與直接(direct)位址值	2	1
XCH A,@Ri	互換累積器 Acc 與間接位址值	1	1
XCD A,@Ri	互換低位元；但高位元不動	1	1

針對以上之指令，動作說明如下：

MOV (目的位元組),(來源位元組)
功用：位元資料的搬移指令
範例：MOV R0,#10H
MOV A,#01H
MOV @R0,A
設程式執行前：
(Acc)=00H, (R0)=00H, RAM(10H)＝00H
在程式執行 MOV R0,#10H 時
(Acc)=00H, (R0)=10H, RAM(10H)＝00H
在程式執行 MOV A,#01H 時
(Acc)=01H, (R0)=10H, RAM(10H)＝00H
在程式執行 MOV @R0,A 後
(Acc)=01H, (R0)=10H, RAM(10H)＝01H

1. MOV A,Rn

組譯碼：

1	1	1	0	1	r	r	r

位元長度：1

機械週期：1

動作情形：MOV (A)←(Rn)；n＝0～7

2. MOV A,direct

組譯碼：

1	1	1	0	0	1	0	1

位元長度：2

機械週期：1

動作情形：MOV (A)←(direct)

3. MOV A,@Ri

組譯碼：

0	1	1	1	0	1	1	r

位元長度：1

機械週期：1

動作情形：MOV (A)←(i)；@Ri＝內部資料記憶體值

4. MOV A,#data

組譯碼：

1	1	1	1	0	1	0	0

immediate data

位元長度：2

機械週期：1

動作情形：MOV (A)←#data

5. MOV Rn,A

組譯碼：

1	1	1	1	1	r	r	r

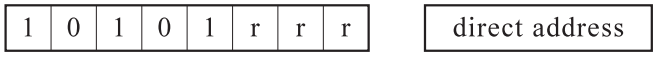
位元長度：1

機械週期：1

動作情形：MOV (Rn)←(A)

6. MOV Rn,direct

組譯碼：



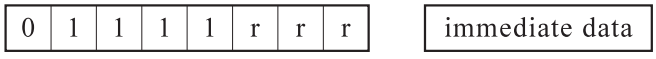
位元長度：2

機械週期：2

動作情形：MOV (Rn)←(direct address)

7. MOV Rn,#data

組譯碼：



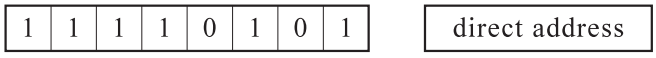
位元長度：2

機械週期：1

動作情形：MOV (Rn)←(#data)

8. MOV direct,A

組譯碼：



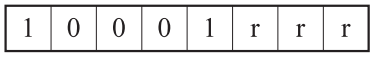
位元長度：2

機械週期：1

動作情形：MOV (direct)←(A)

9. MOV direct,Rn

組譯碼：



位元長度：2

機械週期：2

動作情形：MOV (direct)←Rn；n = 0～7

10. MOV direct,direct

組譯碼：

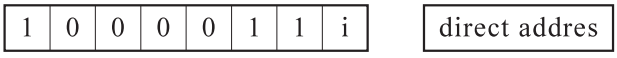
位元長度：3

機械週期：2

動作情形：MOV (direct)←(direct)

11. MOV direct,@Ri

組譯碼：



位元長度：2

機械週期：2

動作情形：MOV (direct)←((Ri))；@Ri＝內部資料記憶體值

12. MOV direct,#data

組譯碼：

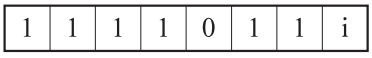
位元長度：3

機械週期：2

動作情形：MOV (direct)←#data

13. MOV i,A

組譯碼：



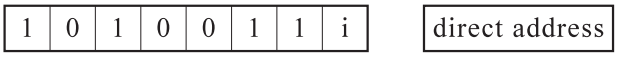
位元長度：1

機械週期：1

動作情形：MOV ((Ri))←A

14. MOV i,direct

組譯碼：



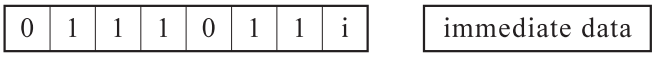
位元長度：2

機械週期：2

動作情形：MOV ((Ri))←direct

15. MOV @Ri,#data

組譯碼：



位元長度：2

機械週期：1

動作情形：MOV ((Ri))←#data

16. MOV DPTR,#data 16

組譯碼：

1	0	0	1	0	0	0	0

immediate data 位元 15～8

immediate data 位元 7～10

位元長度：3

機械週期：2

動作情形：MOV (DPTR)←#data 15～0

範例

```
MOV DPTR,#0015H；16位元常數資料丟入資料指標(DPTR)
```

設程式執行前：

```
(DPTR)=1234H, (DPH)=12H, (DPL)=34H
```

在程式執行 `MOV DPTR,#0015H` 時

```
(DPTR)=0015H
```

結果

```
(DPH)=00H, (DPL)=15H
```

17. MOVC A,＋DPTR

組譯碼：

1	0	0	1	0	0	1	1

位元長度：1

機械週期：2

動作情形：MOV (A)←((A)＋(DPTR))

範例

```
MOV A,#12H
MOV DPTR,#1000H
MOVC A,@A+DPTR
```

讀取程式記憶體值存至Acc累積器

設程式執行前：

```
(Acc)=00H, (DPTR)=0000H
```

在程式執行 `MOV A,#12H` 時

```
(Acc)=12H
```

在程式執行 MOV DPTR,#1000H時
(DPTR)=1000H
在程式執行 MOVX A,@A+DPTR時
(Acc)=(1012H); @A+DPTR=1012H

18. MOVC A,@A+PC

組譯碼：

1	0	0	0	0	0	1	1

位元長度：1
機械週期：2
動作情形：MOVC (PC)←PC + 1
(A)←((A)+(PC))

19. MOVX A,@Ri

組譯碼：

1	1	1	0	0	0	1	i

位元長度：1
機械週期：2
動作情形：MOVX (A)←((Ri)) ；@Ri=外部資料記憶體位址值

範例

MOV R0,#12H
MOVX A,@Ro
} 讀／寫外部擴充資料記憶體值

設程式執行前：
(Acc)=00H, R0=00H, RAM(0012H)=30H
在程式執行 MOV R0,#12H時
(Acc)=00H, R0=12H, RAM(0012H)=30H=外部RAM位址爲30H
在程式執行 MOVX A,@Ro時
(Acc)=30H, R0=12H, RAM(0012H)=30H

20. MOVX A,@DPTR

組譯碼：

1	1	1	0	0	0	0	0

位元長度：1

機械週期：2

動作情形：MOVX (A)←((DPTR))　；@DPTR=外部資料記憶體位址值

21. MOVX @Ri,A

組譯碼：

1	1	1	0	0	0	1	i

位元長度：1

機械週期：2

動作情形：MOVX ((Ri))←(A)

22. MOVX @DPTR,A

組譯碼：

1	1	1	1	0	0	0	0

位元長度：1

機械週期：2

動作情形：MOVX ((DPTR))←(A)

23. PUSH direct

組譯碼：

1	1	0	0	0	0	0	0

direct address

位元長度：1

機械週期：2

動作情形：PUSH (SP)←(SP)＋1

((SP))←(direct)

範例

```
PUSH DPL
PUSH DPH   } 資料值丟入堆疊(SP)
```

設程式執行前：

```
SP=01H, DPH=12H, DPL=34H
RAM(01H)=00H, RAM(02H)=00H, RAM(03H)=00H
```

在程式執行 PUSH DPL 時

```
SP=02H, DPH=12H, DPL=34H
RAM(01H)=00H, RAM(02H)=34H, RAM(03H)=00H
```

在程式執行 PUSH DPH 時

```
SP=03H, DPH=12H, DPL=34H
RAM(01H)=00H, RAM(02H)=34H, RAM(03H)=12H
```

24. XCH A,Rn

組譯碼：

1	1	0	0	1	r	r	r

位元長度：1

機械週期：1

動作情形：XCH (A)←(Rn)

25. XCH A,direct

組譯碼：

1	1	0	0	0	1	0	1

direct address

位元長度：2

機械週期：1

動作情形：XCH (A)⇄(direct)

26. XCH A,@Ri

組譯碼：

1	1	0	0	0	1	1	i

位元長度：1

機械週期：1

動作情形：XCH (A)⇄((Ri)) ；Ri=間接位址值

範例

```
XCH  A,06H
XCH  @A,@Ro   } Acc 累積器值與暫存器 R0 值互換
XCH  A,Ro
```

設程式執行前：

```
(Acc)=00H, R0=10H, RAM(06H)=22H, RAM(07H)=33H
```

在程式執行 `XCH  A,06H` 時

```
(Acc)=22H, R0=10H, RAM(06H)=00H, RAM(07H)=33H
```

在程式執行 `XCH  A,@Ro` 時

```
(Acc)=33H, R0=10H, RAM(06H)=00H, RAM(07H)=22H
```

在程式執行 `XCH  A,R0` 時

```
(Acc)=10H, R0=33H, RAM(06H)=00H, RAM(07H)=22H
```

27. POP direct

組譯碼：

1	1	0	1	0	0	0	0

direct address

位元長度：2

機械週期：2

動作情形：POP (direct)←((SP))

(SP)←(SP)-1

範例

```
POP  DPH
POP  SP    } 取回堆疊 (SP) 資料
```

設程式執行前：

```
SP=05H, DPH=00H, DPL=00H
RAM(03H)=34H, RAM(04H)=56H, RAM(05H)=78H
```

在程式執行 `POP  DPH` 時

```
SP=04H, DPH=78H, DPL=00H
RAM(03H)=34H, RAM(04H)=56H, RAM(05H)=78H
```

在程式執行 `POP  SP` 時

```
SP=56H, DPH=78H, DPL=00H
RAM(03H)=34H, RAM(04H)=56H, RAM(05H)=78H
```

28. XCHD A,@Ri

組譯碼：

1	1	0	0	0	1	1	i

位元長度：1

機械週期：1

動作情形：XCHD (A3-0)←((Ri3-0))

範例

```
XCHD A,@R0
XCHD A,@R0
```
兩組低四位元資料互換

設程式執行前：

```
Acc=11H, R0=30H, RAM(30H)=78H
```

在程式執行 `XCHD A,@R0` 時

```
Acc=18H, R0=30H, RAM(30H)=71H
```

在程式執行 `XCHD A,@R0` 時

```
Acc=11H, R0=30H, RAM(30H)=78H
```

◇ 邏輯運算指令

如下表 4-2 是爲邏輯運算指令之整理與說明，功用以 8 位元之位元組作布林代數運作。

針對以上之指令，動作說明如下：

1. ANL A,Rn

組譯碼：

0	1	0	1	1	r	r	r

位元長度：1

機械週期：1

動作情形：ANL (A)←(A) AND(Rn)；n = 0～7

表 4-2 邏輯運算指令表

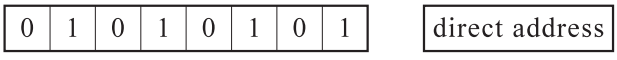
指令	內容說明	長度	機械週期
ANL A,Rn	累積器 Acc 值與暫存器 Rn 值作 AND 運算	1	1
ANL A,direct	累積器Acc值與直接(direct)位址作AND運算	2	1
ANL A,@Ri	累積器Acc值與間接位址Ri值作AND運算	1	1
ANL A,#data	累積器 Acc 值與常數值作 AND 運算	2	1
ANL direct,A	直接(direct)位址值與累積器 Acc 值作 AND 運算	2	1
ANL direct,#data	直接(direct)位址值與常數值作AND運算	3	2
ORL A,Rn	累積器Acc值與暫存器Rn值作OR運算	1	1
ORL A,direct	累積器Acc值與直接(direct)位址作OR運算	2	1
ORL A,@Ri	累積器 Acc 值與間接位址 Ri 值作 OR 運算	1	1
ORL A,#data	累積器 Acc 值與常數值作 OR 運算	2	1
ORL direct,A	直接(direct)位址值與累積器 Acc 值作 OR 運算	2	1
ORL direct,#data	直接(direct)位址值與常數值作OR運算	3	2
XRL A,Rn	累積器 Acc 值與暫存器 Rn 值作互斥或運算	1	1
XRL A,direct	累積器 Acc 值與直接(direct)位址作互斥或運算	2	1
XRL A,@Ri	累積器 Acc 值與間接位址 Ri 值作互斥或運算	1	1

表 4-2 邏輯運算指令表(續)

指令	內容說明	長度	機械週期
XRL A,#data	累積器 Acc 值與常數值作互斥或運算	2	1
XRL direct,A	直接(direct)位址值與累積器 Acc 值作互斥或運算	2	1
XRL direct,#data	直接(direct)位址值與常數值作互斥或運算	3	2
CLR A	累積器 Acc 值=0	1	1
CPL A	累積器 Acc 值作補數	1	1
RL A	累積器 Acc 值向左移動一位元	1	1
RR A	累積器 Acc 值向右移動一位元	1	1
RLC A	累積器 Acc 值與進位旗號 C 值向左移動一位元	1	1
RRC A	累積器 Acc 值與進位旗號 C 值向右移動一位元	1	1
SWAP A	累積器Acc值的高 4 位元與低 4 位元值互換	1	1

2. ANL A,direct

組譯碼：



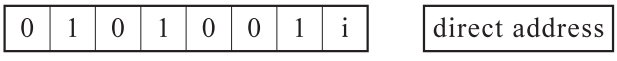
位元長度：2

機械週期：1

動作情形：ANL (A)←(A) AND (direct)

3. ANL A,@Ri

組譯碼：



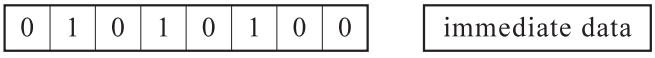
位元長度：1

機械週期：1

動作情形：ANL (A)←(A) AND((Ri))

4. ANL A,#data

組譯碼：



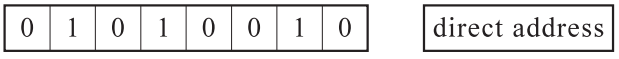
位元長度：2

機械週期：1

動作情形：ANL (A)←(A) AND (#data)

5. ANL direct,A

組譯碼：



位元長度：2

機械週期：1

動作情形：ANL (direct)←(direct) AND (Acc)

6. ANL direct,#data

組譯碼：

位元長度：3

機械週期：2

動作情形：ANL (direct)←(direct) AND (#data)

範例

```
MOV  R0,#11H  }
ANL  A,R0     }資料位元組作AND運算
```

設程式執行前

Acc=25H=0010 0101B, R0=00H=0000 0000B

在程式執行MOV R0,#11H時

Acc=25H=0010 0101B, R0=11H=0001 0001B

在程式執行ANL A,R0時

```
      Acc = 25H = 0010 0101  B
AND ) Ro  = 11H = 0001 0001  B
                  ------------
                  0000 0001  B ──► (Acc)
```

結果(Acc)=0000 0001B=01H

7. ORL A,Rn

組譯碼：

0	1	0	0	1	r	r	r

位元長度：1

機械週期：1

動作情形：ORL (A)←(A) OR (Rn)

8. ORL A,direct

組譯碼：

0	1	0	0	0	1	0	1

direct address

位元長度：2

機械週期：1

動作情形：ORL (A)←(A) OR (direct)

9. ORL A,@Ri

組譯碼：

0	1	0	0	0	1	1	i

位元長度：1

機械週期：1

動作情形：ORL (A)←(A) OR((Ri))

10. ORL A,#data

組譯碼：

0	1	0	0	0	1	0	0

immediate data

位元長度：2

機械週期：1

動作情形：ORL (A)←(A) OR (#data)

11. ORL direct,A

組譯碼：

0	1	0	0	0	0	1	0

direct address

位元長度：2

機械週期：1

動作情形：ORL (direct)←(direct) OR (Acc)

12. ORL direct,#data

組譯碼：

位元長度：3

機械週期：2

動作情形：ORL (direct)←(direct) OR (#data)

範例

```
MOV  18H,#35H
ORL  A,18H
```

設程式執行前

(18H)=00H=0000 0000B, (Acc)=23H=0010 0011B

在程式執行 MOV 18H,#35H 時

(18H)=35H=0011 0101B, (Acc)=23H=0010 0011B

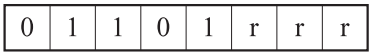
在程式執行 ANL A,R0 時

```
     (18H) = 35H = 0011 0101  B
OR ) (Acc) = 23H = 0010 0011  B
-------------------------------
                   0101 1000  B ──► Acc
```

結果 Acc=0101 1000B=58H

13. XRL A,Rn

組譯碼：



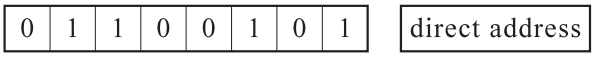
位元長度：1

機械週期：1

動作情形：XRL (A)←(A) ⊕ (Rn)

14. XRL A,direct

組譯碼：



位元長度：2

機械週期：1

動作情形：XRL (A)←(A) ⊕ (direct)

15. XRL A,@Ri

組譯碼：

0	1	1	0	0	1	1	i

位元長度：1

機械週期：1

動作情形：XRL (A)←(A) ⊕((Ri))

16. XRL A,#data

組譯碼：

0	1	1	0	0	1	0	0	immediate data

位元長度：2

機械週期：1

動作情形：XRL (A)←(A) ⊕ (#data)

17. XRL direct,A

組譯碼：

0	1	1	0	0	0	1	0	direct address

位元長度：2

機械週期：1

動作情形：XRL (direct)←(direct) ⊕ (Acc)

18. XRL direct,#data

組譯碼：

0	1	1	0	0	0	1	1	direct address	immediate data

位元長度：3

機械週期：2

動作情形：XRL (direct)←(direct) ⊕ (#data)

範例

```
MOV  R0,#47H
XRL  A,R0
```
互斥或運算

設程式執行前

(R0)=00H=0000 0000B, Acc=27H=0010 0111B

在程式執行MOV R0,#47H時

(R0)=47H=0100 0110B, Acc=27H=0010 0111B

在程式執行 XRL A,R0 時

```
    (Ro)  = 47H = 0100 0111  B
⊕)  (Acc) = 27H = 0010 0111  B
                  0110 0000  B ──→ Acc
```

結果 Acc=0110 0000B=60H

19. CPL direct,A

組譯碼：

1	1	1	1	0	1	0	0

位元長度：1

機械週期：1

動作情形：CPL (A)←$\overline{(A)}$

範例

CPL A

設程式執行前

Acc=7BH=0111 1011B

在程式執行 CPL A 時

Acc=$\overline{Acc}$=$\overline{01111011B}$=1000 0100B=84H

結果 Acc=84H

20. CLR A

組譯碼：

1	1	1	0	0	1	0	0

位元長度：1

機械週期：1

動作情形：CLR (A)←0

範例

```
MOV  A,#27H
CLR  A
```

設程式執行前

(Acc)=00H

在程式執行 MOV A,#27H 時

(Acc)=27H

在程式執行 CLR A 時

(Acc)=0H

21. RL A

組譯碼：

0	0	1	0	0	0	1	1

位元長度：1

機械週期：2

動作情形：RL (A0)←(A7)

範例

```
RL  A
```

設程式執行前

(Acc)=25H=0010 0101B

在程式執行 RL A 時

(Acc)=0100 1010B=4AH

22. RLC A

組譯碼：

0	0	1	1	0	0	1	1

位元長度：1

機械週期：1

動作情形：RLC (A)←(A7)

(C)←(A7)

範例

```
RLC  A
RLC  A
設程式執行前
(Acc)=84H=1011 0100B
(CY)=0
在程式執行RLC  A時
(Acc)=0110 1000B=68H
(CY)=1
在程式執行RLC  A時
(Acc)=1101 0000B=D0H
(CY)=0
```

23. RR A

組譯碼：

0	0	1	1	0	0	1	1

位元長度：1

機械週期：1

動作情形：RR (A7)←(A0)

範例

```
RR   A
設程式執行前
(Acc)=84H=1000 0100B
在程式執行RR   A時
(Acc)=0100 0010B=42H
```

24. RRC A

組譯碼：

0	0	0	1	0	0	1	1

位元長度：1

機械週期：1

動作情形：RRC (A7)←(C)

(C)←(A0)

範例

```
RRC  A
RRC  A
設程式執行前
(Acc)=56H=0101 0110B,(CY)=0
在程式執行RRC  A時
(Acc)=0010 1011B=2BH
(CY)=0
在程式執行RRC  A時
(Acc)=0001 0101B=15H
(CY)=0
```

25. SWAP A

組譯碼：

1	1	0	0	0	1	1	0

位元長度：1

機械週期：1

動作情形：SWAP (A3-0)←(A7-4)

範例

```
SWAP  A
SWAP  A
設程式執行前
(Acc)=64H=0110 0100B
在程式執行SWAP A時
(Acc)=0100 0110B=46H
在程式執行SWAP A時
(Acc)=0110 0100B=64H
```

◇ 算術運算指令

如下表 4-3 是爲算術運算指令之整理與說明，而算術運算指令執行時，是將影響 PSW 值。

表 4-3 算術運算指令表

指令	內容說明	長度	機械週期
ADD A,Rn	暫存器 Rn 值加至累積器 Acc 中	1	1
ADD A,direct	直接(direct)位址加至累積器 Acc 中	2	1
ADD A,@Ri	間接地址 Ri 值加至累積器 Acc 中	1	1
ADD A,#data	常數資料值加至累積器 Acc 中	2	1
ADDC A,Rn	暫存器 Rn 值與進位旗號 C 值加至累積器 Acc 中	1	1
ADDC A,direct	直接(direct)位址與進位旗號 C 值加至累積器 Acc 中	2	1
ADDC A,@Ri	間接地址 Ri 值與進位旗號 C 值加至累積器 Acc 中	1	1
ADDC A,#data	常數資料值與進位旗號 C 值加至累積器 Acc 中	2	1
SUBB A,Rn	累積器 Acc 減進位旗號 C 值與暫存器 Rn 值	1	1
SUBB A,direct	累積器 Acc 減進位旗號 C 值與直接(direct)位址	2	2
SUBB A,@Ri	累積器 Acc 減進位旗號 C 值與間接地址 Ri 值	1	1
SUBB A,#data	累積器 Acc 減進位旗號 C 值與常數資料值	2	1
INC A	累積器 Acc 值加 1	1	1
INC Rn	暫存器 Rn 值加 1	1	1

表 4-3 算術運算指令表(續)

指令	內容說明	長度	機械週期
INC direct	直接(direct)位址值加 1	2	1
INC @Ri	間接位址 Ri 值加 1	1	1
INC DPTR	資料指標 DPTR 值加 1	1	2
DEC A	累積器 Acc 值減 1	1	1
DEC Rn	暫存器 Rn 值減 1	1	1
DEC direct	直接(direct)位址值減 1	2	1
DEC @Ri	間接位址 Ri 值減 1	1	1
MUL AB	累積器 Acc 值與暫存器 B 值相乘，再把結果的高位元存至暫存器 B 中，低位元存至累積器 Acc 中	1	4
DIV AB	累積器Acc值除暫存器B，再把商數存至累積器Acc中，餘數存至暫存器B中	1	4
DA A	累積器 Acc 值調整成十進制	1	1

針對以上之指令，動作說明如下：

1. ADD A,Rn

   組譯碼：

0	0	1	0	1	r	r	r

   位元長度：1

   機械週期：1

   動作情形：ADD (A)←(A)＋(Rn)；n＝0～7

2. ADD A,direct

   組譯碼：

0	0	1	0	0	1	0	1

direct address

   位元長度：2

   機械週期：1

   動作情形：ADD (A)←(A)＋(direct)

3. ADD A,@Ri

組譯碼：

0	0	1	0	0	1	1	i

位元長度：1

機械週期：1

動作情形：ADD (A)←(A)＋((Ri))；@Ri=內部資料記憶體位址值

4. ADD A,#data

組譯碼：

0	0	1	0	0	1	0	0

immediate data

位元長度：2

機械週期：1

動作情形：ADD (A)←(A)＋(#data)

範例

```
MOV   R0,#12H
ADD   A,R0
設程式執行前
Acc=57H=01010111B
R0=00H=00000000B
在程式執行MOV   R0,#12H時
Acc=57H=01010111B, R0=12H=00010010B
在程式執行ADD   A,R0時

      Acc = 57H = 0101 0111  B
ADD ) Ro  = 12H = 0001 0010  B
                  0110 1001  B ──► Acc

結果：Acc=01101001B=69H
進位旗號：CY=0⇔因C7=0
輔助進位旗號：AC=1⇔因C3=1
溢位旗號：0V=0⇔因C6⊕C7=0⊕0=0
```

5. ADDC A,Rn

組譯碼：

0	0	1	1	1	r	r	r

位元長度：1

機械週期：1

動作情形：ADDC (A)←(A)＋(Rn)＋(C)

6. ADDC A,direct

組譯碼：

0	0	1	1	0	1	0	1	direct address

位元長度：2

機械週期：1

動作情形：ADDC (A)←(A)＋(direct)＋(C)

7. ADDC A,@Ri

組譯碼：

0	0	1	1	0	1	1	i

位元長度：1

機械週期：1

動作情形：ADDC (A)←(A)＋((Ri))＋(C)

8. ADDC A,#data

組譯碼：

0	0	1	1	0	1	0	0	immediate data

位元長度：2

機械週期：1

動作情形：ADDC (A)←(A)＋(#data)＋(C)

範例

```
MOV  R0,#35H
ADDC A,R0
設程式執行前
Acc=88H=10001000B, R0=00H=00000000B, CY=1
在程式執行MOV R0,#35H時
Acc=88H=10001000B, R0=35H=00110101B, CY=1
```

在程式執行 ADDC A,R0 時

```
        Acc = 88H = 1000 1000  B
        Ro  = 35H = 0011 0101  B
ADDC )  CY  =  1H =         1  B
                    1011 1110  B ──→ Acc
```

結果：Acc= 1 0 1 1 1 1 1 0B

C7 C6 C5 C4 C3 C2 C1 C0

進位旗號：CY=1⇔因 C7=1

輔助進位旗號：AC=1⇔因 C3=1

溢位旗號：0V=1⇔因 C6⊕C7=0⊕1=1

9. SUBB A,Rn

組譯碼：

1	0	0	1	1	r	r	r

位元長度：1

機械週期：1

動作情形：SUBB (A)←(A)-(Rn)-(C)

10. SUBB A,direct

組譯碼：

0	1	1	0	0	0	1	1	direct address

位元長度：2

機械週期：2

動作情形：SUBB (A)←(A)-(direct)-(C)

11. SUBB A,i

組譯碼：

1	0	0	1	0	1	1	i

位元長度：1

機械週期：1

動作情形：SUBB (A)←(A)-((Ri))-(C)

12. SUBB A,#data

組譯碼：

1	0	0	1	0	1	0	0	immediate data

位元長度：2

機械週期：1

動作情形：SUBB (A)←(A)-(#data)-(C)

範例

```
MOV   R0,#07H
SUBB  A,R0
設程式執行前
Acc=ABH=10101011B, R0=00000000B,CY=1B
在程式執行MOV   R0,#07H時
Acc=ABH=10101011B, R0=00000111B,CY=1B
在程式執行SUBB  A,R0時

        Acc = 1010 1011  B
 SUBB ) Ro  = 0000 0111  B
        ------------------
             1010 0100  B
   SUBB )           1  B = Cy
        ------------------
             1001 1011  B ──► Acc
```

結果：Acc=10011111B=9FH

CY=1⇔因C7=1

AC=1⇔因C3=1

0V=1⇔因C6⊕C7=0⊕1=1

13. INC A

組譯碼：

0	0	0	0	0	1	0	0

位元長度：1

機械週期：1

動作情形：INC (A)←(A)＋1

14. INC Rn

組譯碼：

0	0	0	0	1	r	r	r

位元長度：1

機械週期：1

動作情形：INC (Rn)←(Rn)＋1；n=0～7

15. INC direct

組譯碼：

0	0	0	0	0	1	0	1	direct address

位元長度：2

機械週期：1

動作情形：INC (direct)←(direct)＋1

16. INC @Ri

組譯碼：

0	0	0	0	0	1	1	i

位元長度：1

機械週期：1

動作情形：INC ((Ri))←((Ri))＋1

17. INC DPTR

組譯碼：

1	0	1	0	0	0	1	1

位元長度：1

機械週期：2

動作情形：INC (DPTR)←(DPTR)＋1

範例

```
INC   R1
INC   1
INC   R1
```

設程式執行前

R1=06H, RAM(07H)=00H

在程式執行 INC R1 時

```
R1=07H, RAM(07H)=00H
在程式執行 INC @R1 時
R1=07H, RAM(07H)=01H
在程式執行 INC R1 時
R1=08H, RAM(07H)=08H
```

範例

```
INC  DPTR
INC  DPTR
INC  DPTR
設程式執行前
DPTR=11BEH⇒DPL=BEH, DPH=11H
在程式執行 INC  DPTR 時
DPTR=11BFH⇒DPL=BFH, DPH=11H
在程式執行 INC  DPTR 時
DPTR=11C0H⇒DPL=C0H, DPH=11H
在程式執行 INC  DPTR 時
DPTR=11C1H⇒DPL=C1H, DPH=11H
```

18. DEC A

組譯碼：

0	0	0	1	0	1	0	0

位元長度：1

機械週期：1

動作情形：DEC (A)←(A)-1

19. DEC Rn

組譯碼：

0	0	0	1	1	r	r	r

位元長度：1

機械週期：1

動作情形：DEC (Rn)←(Rn)-1;n=0～7

20. DEC direct

組譯碼：

0	0	0	1	0	1	0	1	direct address

位元長度：2

機械週期：1

動作情形：DEC (direct)←(direct)-1

21. DEC @Ri

組譯碼：

0	0	0	1	0	1	1	i

位元長度：1

機械週期：1

動作情形：DEC ((Ri))←((Ri))-1

範例

```
DEC R1
DEC @r1
DEC R1
設程式執行前
R1=ABH, RAM(01H)=00H
在程式執行 DEC R1 時
R1=AAH, RAM(07H)=00H
在程式執行 DEC 1 時
R1=AAH, RAM(07H)=FFH
在程式執行 DEC R1 時
R1=A9H, RAM(07H)=FFH
```

22. MUL AB

組譯碼：

1	0	1	0	0	1	0	0

位元長度：1

機械週期：4

動作情形：MUL (BA)←(A)*(B)

範例 MUL AB

設程式執行前 Acc=84H=8×16¹＋4×16⁰=132D,

B=24H=2×16¹＋4×16⁰=36D

在程式執行 MUL AB 時

A*B=132*36=4752D=1188H

結果：Acc=88H

R0=11H

CY=0

0V=1↔因 A*B 乘積超過 225(FFH)

23. DIV AB

組譯碼：

1	0	0	0	0	1	0	0

位元長度：1

機械週期：4

動作情形：DIV (BA)←(A)÷(B)

範例 DIV AB

設程式執行前

Acc=B2H=11×16¹＋2×16⁰=178D

B=43H=4×16¹＋3×16⁰=67D

在程式執行 DIV AB 時

A÷B=178D÷67D=2…44

結果：Acc=02H

R0=44D

CY=0

0V=0

24. DA A

組譯碼：

1	1	0	1	0	1	0	0

位元長度：1

機械週期：1

動作情形：DA (A)←BCD(A)

範例

```
ADD   A,R0
DA    A
設程式執行前
Acc=71H, R0=40H, CY＝0
在程式執行ADD  A,R0時

      0111 0001 = Acc
ADD ) 0100 0000 = Ro
      1011 0001 ⟶ Acc

結果：Acc=B1H, R0=40H, CY=0
在程式執行DA   A時
Acc=B1H=10110001B在C3=1，所以AC=1，須加06H

      0111 0001 = Acc
ADD ) 0100 0000 = Ro
      1011 0001 ⟶ Acc

在高4位元爲B(大於9)故Acc＋60H
      結果Acc=17H，但CY=1
```

◇ 布林運算指令

如下表 4-4 是爲布林運算指令之整理與說明，其能與定址之內部資料記憶體及暫存器 Rn 直接作運算。

表 4-4 布林運算指令表

指令	內容說明	長度	機械週期
ANL C,bit	進位旗號(C)與位元位址，作AND運算	2	2
ANL C,/bit	位元位址取補數，與進位旗號，作AND運算	2	2
ORL C,bit	進位旗號(C)與位元位址，作OR運算	2	2
ORL C,/bit	位元位址取補數，與進位旗號，作OR運算	2	2
CPL C	進位旗號(C)值取補數	1	1
CPL bit	位元位址值取補數	2	1
CLR C	進位旗號(C)值清除爲0	1	1
CLR bit	位元位址值清除爲0	2	1
SETB C	進位旗號(C)值設爲1	1	1
SETB bit	位元位址值設爲1	2	1
MOV C,bit	位元位址值移至進位旗號(C)中	2	2
MOV bit,C	進位旗號(C)移至位元位址值中	2	2
JC rel	進位旗號(C)=1時，會跳至相對位址rel執行程式	2	2
JNC rel	進位旗號(C)=0時，會跳至相對位址rel執行程式	2	2
JB bit,rel	(位元位址)=1時，會跳至相對位址 rel執行程式	3	2
JNB bit,rel	(位元位址)=0時，會跳至相對位址 rel執行程式	3	2
JBC bit,rel	(位元位址)=1時，會跳至相對位址 rel執行程式，且同時將(位元位址)=0	3	2

針對以上之指令，動作說明如下：

1. ANL C,bit

組譯碼：

1	0	0	0	0	0	1	0	bit address

位元長度：2

機械週期：2

動作情形：ANL (C)←(C)AND(bit)

2. ANL C,/bit

組譯碼：

1	0	1	1	0	0	0	0	bit address

位元長度：2

機械週期：2

動作情形：ANL (C)←(C)AND(not(bit))

範例

```
ANL C,A.1  }
ANL C,/A.1 }作 AND 運算
設程式執行前
Acc=BAH=10111010B
CY=1B
在程式執行 ANL  C,A.1 時
Acc=10111010B
CY=1B
在程式執行 ANL  C,/A.1 時
Acc=10111010B
CY=0B
```

3. ORL C,bit

組譯碼：

0	1	1	1	0	0	1	0	bit address	immediate data

位元長度：2

機械週期：2

動作情形：ORL (C)←(C)OR(bit)

4. ORL C,/bit

組譯碼：

0	1	1	0	0	0	1	1	bit address

位元長度：2

機械週期：2

動作情形：ORL (C)←(C)OR ($\overline{bit}$)

範例

```
ORL C,A.2
ORL C,/A.2
設程式執行前
Acc=B8H=10111000B
CY=0B
在程式執行ORL  C,A.2時
Acc=10111000B
CY=0B
在程式執行ORL  C,/A.2時
Acc=10111000B
CY=1B
```

5. CPL C

組譯碼：

1	0	1	1	0	0	1	1

位元長度：1

機械週期：1

動作情形：CPL (C)←($\overline{C}$)

6. CPL bit

組譯碼：

1	0	1	1	0	0	1	0	bit address

位元長度：2

機械週期：1

動作情形：CPL (bit)←($\overline{bit}$)

範例

```
CPL  P1.3
CPL  P1.4
設程式執行前
P1=B7H=10110111B
在程式執行CPL   P1.3時
P1=10111111B=BFH
在程式執行CPL   P1.4時
P1=10101111B=AFH
```

7. CLR C

組譯碼：

1	1	0	0	0	0	1	1

位元長度：1

機械週期：1

動作情形：CLR (C)←0

8. CLR bit

組譯碼：

1	1	0	0	0	0	1	0	bit address

位元長度：2

機械週期：1

動作情形：CLR (bit)←0

範例

```
CLR  P1.2 }
CLR  P1.4 } 清除位元運算
設程式執行前
P1=98H=10011000B
在程式執行CLR   P1.2時
P1=10011000B=98H
在程式執行CLR   P1.4時
P1=10001000B=88H
```

9. SETB C

組譯碼：

1	1	0	1	0	0	1	1

位元長度：1

機械週期：1

動作情形：SETB (C)←1

10. SETB bit

組譯碼：

1	1	0	1	0	0	1	0

bit address

位元長度：2

機械週期：1

動作情形：SETB (bit)←1

範例

```
SETB A.2
SETB A.3
設程式執行前
Acc=65H=01100101B
在程式執行 SETB  A.2 時
Acc=01100101B=65H
在程式執行 SETB  A.3 時
Acc=01101101B=6DH
```

11. MOV C,bit

組譯碼：

1	0	1	0	0	0	1	1

bit address

位元長度：2

機械週期：1

動作情形：MOV (C)←(bit)

12. MOV bit,C

組譯碼：

1	0	0	1	0	0	1	0

位元長度：2

機械週期：2

動作情形：MOV (bit)←C

範例

```
MOV A.3,C
MOV C,A.0
MOV A.2,C
設程式執行前
Acc=5CH=01011100B, CY=0B
              ↑
            A.3
在程式執行MOV A.3,C時
Acc=01010100B=54H
            ↑
          A.0
在程式執行MOV C,A.0時
CY=0
在程式執行MOV A.2,C時
結果：Acc=01010000B=50H
              ↑
            A.2
```

13. JC rel

組譯碼：

0	1	0	0	0	0	0	0	rel address

位元長度：2

機械週期：2

動作情形：JC (PC)←(PC)＋2

若(C)=1 則(PC)←(PC)＋rel

範例

```
JC    LAB1
CPL   C
JC    LAB2
設程式執行前
CY=0
在程式執行 JC    LAB1 時
CY=0→將執行下一行指令
在程式執行 CPL   C 時
CY=1
在程式執行 JC    LAB2 時
CY=1→直接到 LAB2 處，執行程式
```

14. JNC rel

組譯碼：

0	1	0	1	0	0	0	0	rel address

位元長度：2

機械週期：2

動作情形：JNC (PC)←(PC)＋2

若(C)=0 則(PC)←(PC)＋rel

範例

```
JNC   LAB1
CPL   C
JNC   LAB2
設程式執行前  CY=1
在程式執行 JC    LAB1 時
CY=1→可執行下一行指令
在程式執行 CPL   C 時
CY=0
在程式執行 JNC   LAB2 時
CY=0→直接到 LAB2 處執行程式
```

15. JB bit,rel

組譯碼：

0	0	1	0	0	0	0	0	bit address	rel address

位元長度：3

機械週期：2

動作情形：JB (PC)←(PC)＋3

若 (bit)=1 則(PC)←(PC)＋rel

範例

```
JB  P2.2,LAB1
JB  Acc.1,LAB2
設程式執行前  P2=ABH=10101011B, Acc=93H=10010011B
在程式執行 JB    P2.2,LAB1 時
P2=10101011B
       ↑
     P2.2=0，所以執行下一行指令
在程式執行 JB   Acc.1,LAB2 時
Acc=10010011B
           ↑
     Acc.1，所以直接跳至 LAB2 處執行程式
```

16. JNB bit,rel

組譯碼：

0	0	0	1	0	0	0	0	bit address	rel address

位元長度：3

機械週期：2

動作情形：JNB (PC)←(PC)＋3

若(bit)=0 則(PC)←(PC)＋rel

範例

```
JB    P2.3,LAB1
JB    Acc.2,LAB2
設程式執行前
P2=ADH=10101101B, Acc=71H=01110001B
在程式執行 JB    P2.3,LAB1 時
P2=10101101B
        ↑
      P2.3=1，所以執行下一行指令
在程式執行 JB    Acc.2,LAB2 時
Acc=01110001B
          ↑
      Acc.2=0，直接跳至 LAB2 處執行程式
```

17. JBC bit,rel

組譯碼：

0	0	0	1	0	0	0	0	bit address	rel address

位元長度：3

機械週期：2

動作情形：JBC (PC)←(PC)＋3

若(bit)=1 則(PC)←(PC)＋rel，且同時(bit)=0

範例

```
JBC   Acc.3,LAB1
JBC   Acc.1,LAB2
設程式執行前  Acc=63H=01100011B
在程式執行 JBC   Acc.3,LAB1 時
P2=01100011B
        ↑
      Acc.3=0，所以執行下一行指令
```

在程式執行 JBC　Acc.1,LAB2 時
Acc=01100011B
↑
Acc.1=1，所以直接跳至 LAB2 處執行程式
且 Acc.1=0，
結果 Acc=01100001B=61H

◇ 程式分支指令

如下表 4-5，是爲程式分支指令之整理與說明，其功用是作改變 CPU 執行程式的流程，使程式具彈性與可讀性。

表 4-5　程式分支指令表

指令	內容說明	長度	機械週期
ACALL addr11	呼叫副程式(可定址 2K 範圍，會改變 SP，PC 值)	2	2
ACALL addr16	呼叫副程式(可定址 64K 範圍，會改變 SP，PC 值)	3	2
RET	由副程式返回主程式，且改變 SP，PC 值	1	2
RETI	由中斷副程式返回主程式，且改變 SP，PC 值	1	2
AJMP addr11	程式記憶體 2k，絕對跳躍，會改變 PC 值	2	2
LJMP addr16	程式記憶體 64K，長程跳躍	3	2
SJMP rel	程式記憶體 256 bytes，短程跳躍	2	2
JMP　@A + DPTR	間接跳躍至 Acc 值加上 DPTR 值爲位址	1	2
JZ　rel	若 V_{cc} = 0 跳至 rel 位址，否則往下執行	2	2

表 4-5 程式分支指令表(續)

指令	內容說明	長度	機械週期
JWZ rel	若V_{cc}≠0 跳至 rel 位址執行，否則往下執行	2	2
CJNE A,direct,rel	若V_{cc}≠direct 值，跳至 rel 位址執行，且 Acc 小於 direct 值時，C = 1；否則往下執行	3	2
CJNE A,#data,rel	若V_{cc}≠常數資料，跳至rel位址執行，Acc 值小於常數值時，C = 1；否則往下執行	3	2
CJNE Rn,#data,rel	若 Rn 值≠常數資料，跳至 rel 位址執行，在Rn值小於常數資料時，C = 1；否則往下執行	3	2
CJNE @Ri,#data,rel	若(Ri)值≠常數資料，跳至 rel 位址執行，在(Ri)值小於常數資料時，C = 1；否則往下執行	3	2
CJNE Rn,rel	若(Rn-1)值≠0，跳至rel位址執行，否則往下執行	2	2
CJNE direct,rel	若(direct-1)≠0，跳至 rel 位址執行，否則往下執行	3	2
NOP	沒有動作	1	1

針對以上指令，動作說明如下：

1. ACALL addr11

組譯碼：

a_{10}	a_9	a_8	1	0	0	0	1	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

位元長度：2

機械週期：2

動作情形：ACALL (PC)←(PC)＋2
(SP)←(SP)＋1
(SP)←(PC)
(PC)←addr11

範例

```
          ORG     30H
          ACALL   SUB1
          MOV     A,#02H
          ⋮
SUB1:
          MOV     B,#02H
          RET
```

設程式執行前
PC=30H
SP=07H, RAM(07H)=00H, RAM(09H)=00H
在程式執行ACALL SUB1時
PC=60H
SP=09H, RAM(07H)=00H, RAM(08H)=32H,
RAM(09H)=00H
在程式執行MOV B,#02H時
PC=63H
SP=09H, RAM(07H)=00H, RAM(08H)=32H,
RAM(09H)=00H
在程式執行RET時
PC=32H
SP=07H, RAM(07H)=00H, RAM(08H)=32H,
RAM(09H)=00H

2. LCALL addr16

組譯碼：

0	0	0	1	0	0	1	0	$a_{15}\sim a_8$	$a_7\sim a_0$

位元長度：3

機械週期：2

動作情形：LCALL (PC)←(PC)＋2,(SP)←(SP)＋1

((SP))←(PC)7-0,(SP)←(SP)＋1

((SP))←(PC)15-8,(PC)←addr16

範例

```
        ORG     00H
        LCALL   SUB
        MOV     A,#01H
          ⋮
        ORG     EF0H
SUB:
        MOV     B,#02H
        RET
```

設程式執行前

PC=00H, SP=05H

RAM(05H)=00H, RAM(06H)=00H, RAM(07H)=00H

在程式執行 LCALL SUB 時

PC=EF0H, SP=07H

RAM(05H)=00H, RAM(06H)=02H, RAM(07H)=00H

在程式執行 MOV B,#02H 時

PC=EF3H, SP=07H

RAM(05H)=00H, RAM(06H)=02H, RAM(07H)=00H

在程式執行 RET 時

PC=02H, SP=05H

RAM(05H)=00H, RAM(06H)=02H, RAM(07H)=00H

3. RET

組譯碼：

0	0	1	0	0	0	1	0

位元長度：1

機械週期：2

動作情形：RET (PC)15-8←((SP)), (SP)←(SP)-1

(PC)7-0←((SP)), (SP)←(SP)-1

4. RETI

組譯碼：

0	0	1	1	0	0	1	0

位元長度：1

機械週期：2

動作情形：RETI (PC)15-8←((SP)), (SP)←(SP)-1

(PC)7-0←((SP)), (SP)←(SP)-1

5. AJMP addr11

組譯碼：

a_{10}	a_9	a_8	a	0	0	0	0	$a_7a_6a_5a_4a_3a_2a_1a_0$

位元長度：2

機械週期：2

動作情形：AJMP (PC)10-0←addr11, (PC)←(PC)＋2

6. LJMP addr16

組譯碼：

0	0	0	0	0	0	1	0	addr 15～8	addr 7～0

位元長度：3

機械週期：2

動作情形：LJMP (PC)←addr15-0

7. SJMP rel

組譯碼：

1	0	1	0	0	0	0	0	rel address

位元長度：2

機械週期：2

動作情形：SJMP (PC)←(PC)＋2

(PC)←(PC)＋rel

8. JMP ＋DPTR

組譯碼：

0	1	1	1	0	0	1	1

位元長度：1

機械週期：2

動作情形：JMP (PC)←(A)＋(DPTR)

範例

```
          ORG   20H
          MOV   A,R0 ····· 程式位址 20H
          RL    A ····· 程式位址 22H
          MOV   DPTR,#LAB ····· 程式位址 22H
          MOV   @A+DPTR ····· 程式位址 25H
          ⋮
          ORG   60H
LAB:
          AJMP  LAB0 ····· 程式位址 60H
          AJMP  LAB1 ····· 程式位址 62H
          AJMP  LAB2 ····· 程式位址 64H
          AJMP  LAB2 ····· 程式位址 66H
```

設程式執行前

PC=20H, Acc=00H, R0=01H, DPTR=00H

在程式執行 MOV A,R0 時

PC=21H, Acc=01H, R0=01H, DPTR=00H

在程式執行 RL A 時

PC=22H, Acc=02H, R0=01H, DPTR=00H

在程式執行 MOV DPTR,#LAB 時

PC=25H, Acc=02H, R0=01H, DPTR=01H

在程式執行 JMP @A+DPTR 時

PC=62H, Acc=02H, R0=01H, DPTR=01H

9. JZ rel

組譯碼：

0	1	1	0	0	0	0	0	rel. address

位元長度：2

機械週期：2

動作情形：JZ (PC)←(PC)＋2

在(A)≠0則(PC)←(PC)＋rel

範例

```
JZ    LAB1
DEC   A
JZ    LAB2
```

設程式執行前

Acc≠0=01H

在程式執行 JZ LAB1 時

Acc=01H，執行下一個程式

在程式執行 DEC A 時

Acc=00H

在程式執行 JZ LAB2 時

Acc=00H→此時將直接跳到 LAB2 處執行

10. JNZ rel

組譯碼：

0	1	0	1	0	0	0	1	direct address	rel. address

位元長度：2

機械週期：2

動作情形：JNZ (PC)←(PC)＋2

若 Acc≠0則(PC)←(PC)＋rel

範例

```
JNZ   Level0
INC   A
JZ    Level2
設程式執行前
Acc=00H
在程式執行 JNZ   Level0 時
Acc=00H→執行下一行程式
在程式執行 INC   A 時
Acc=01H
在程式執行 JNZ   Level2 時
Acc=01H→因 Acc≠0，可跳到 Level2 的標記執行程式
```

11. CJNE A,direct,rel

組譯碼：

1	0	1	1	0	1	0	1	direct address	rel. address

位元長度：3

機械週期：2

動作情形：CJNE (PC)←(PC)＋3

若 A≠direct 則(PC)←(PC)＋rel

若 A＜direct 則(C)←1

12. CJNE A,#data,rel

組譯碼：

1	0	1	1	0	1	0	0	immediate data	rel. address

位元長度：3

機械週期：2

動作情形：CJNE (PC)←(PC)＋3

若 Acc≠data 值則(PC)←(PC)＋rel

若 A＜data 值則(C)←1

13. CJNE Rn,#data,rel

組譯碼：

1	0	1	1	1	r	r	r	immediate data	rel. address

位元長度：3

機械週期：2

動作情形：CJNE (PC)←(PC)＋3

若 Rn≠data 值則(PC)←(PC)＋rel

若 Rn＜data 值則(C)←1

14. CJNE @Ri,#data,rel

組譯碼：

1	0	1	1	0	1	1	1	immediate data	rel. address

位元長度：3

機械週期：2

動作情形：CJNE (PC)←(PC)＋3

若((Ri))≠data 值則(PC)←(PC)＋rel

若((Ri))＜data 值則(C)←1

範例

```
CJNZ  A,#03H,EQU
   ⋮
EQU:
JC    SMALL
   ⋮
SMALL:
     MOV   B,#04H
```

設程式執行前

Acc=00H

在程式執行 CJNE A,#03H,EQU 時

Acc=00H

CY=1↔因 Acc 值＜03H，跳到 EQU 執行程式

在程式執行 JC SMALL 時

Acc=00H, CY=1↔跳到 SMALL 處執行程式

15. DJNZ Rn,rel

組譯碼：

1	1	0	1	1	r	r	r	rel. address

位元長度：2

機械週期：2

動作情形：DJNZ (PC)←(PC)＋2, (Rn)←(Rn)-1

若 Rn 值≠0 則(PC)←(PC)＋rel

16. DJNZ direct,rel

組譯碼：

1	1	0	1	0	1	0	1	direct address	rel. address

位元長度：3

機械週期：2

動作情形：DJNZ (PC)←(PC)＋2, (direct)←(direct)-1

若(direct)值≠0 則(PC)←(PC)＋rel

範例

```
DJNZ R0,LAB1
DJNZ 60H,LAB2
DJNZ 50H,LAB3
設程式執行前
R0=01H, RAM(50H)=00H, RAM(60H)=01H
在程式執行 DJNZ R0,LAB1 時
R0=R0-1=00H↔因 R0=0，可執行下一個指令
RAM(50H)=00H
RAM(60H)=01H
在程式執行 DJNZ 60H,LAB2 時
R0=00H
RAM(50H)=00H
RAM(60H)=01H-1H=00H→可執行下一個指令
在程式執行 DJNZ 50H,LAB3 時
```

```
R0=00H
RAM(50H)=00H-1H=FFH↔因 RAM(50H)≠0，可跳到
LAB3處執行程式
RAM(60H)=00H
```

17. NOP

組譯碼：

0	0	0	0	0	0	0	0

位元長度：1

機械週期：1

動作情形：沒有

習題

1. 請試解釋下列指令的意義：

ADDC A,<src-byte>：_____

ANL C,<src-bit>：_____

DEC byte：_____

DJNZ <byte>,<rel address>：_____

JBC bit,rel：_____

LCALL addr16：_____

MOV <dest-bit>,<src-bit>：_____

MOVC A,＋<base-reg>：_____

MOVX <dest-byte>,<src-byte>：_____

ORL C,<src-bit>：_____

RRC A：_____

SWAP A：_____

XCH A,<byte>：_____

XCHD A,i：_____

XRL <dest-byte>,<src-byte>：_____

2. 請試寫出下列運算結果：

0	1	1	0	1011 1001	1001 0111
− 0	− 1	− 0	− 1	+) 0001 0011	−) 0101 0011
(　　)	(　　)	(　　)	(　　)	(　　)	(　　)

3. 假設 Acc=58H,R0=23H 在執行 ADD A,R0 時，結果 Acc=？
4. 假設 Acc=A3H,R0=62H 在執行 ANL A,R0 時，結果 Acc=？
5. 假設 Acc=E1H,R0=68H 在執行 XRL A,R0 時，結果 Acc=？
6. 已知Acc=BCH，問其進位旗號(CY)＝？輔助進位旗號(AC)＝？溢位旗號(0V)＝？