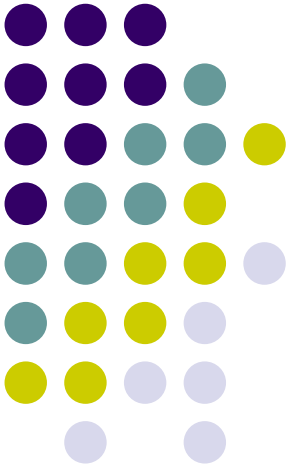
數位信號處理實務

Chap 6. 簡易程式演練



大綱

6.1 實習一：執行 1+2+3+……+14+15 的加法運算

6.2 實習二：數個 Byte 之加法運算

6.3 實習三：清除內部 RAM 某段記憶體

6.4 實習四：將多個 BYTE 的資料內容左移 1 位元

大綱

6.1　實習一：執行 1+2+3+……+14+15 的加法運算

程式功能

流程圖

程式碼

功能驗證

6.2　實習二：數個 Byte 之加法運算

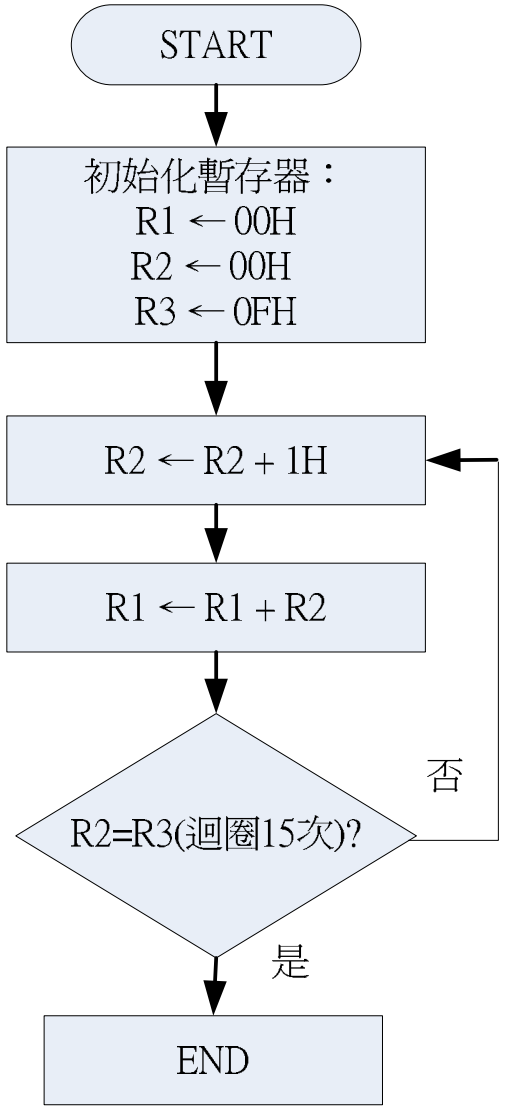
6.3　實習三：清除內部 RAM 某段記憶體

6.4　實習四：將多個 BYTE 的資料內容左移 1 位元

程式功能

- 撰寫執行 1+2+3+……+14+15 的運算程式，其中暫存器 R2 作為累積器，亦可作為計數器 (Counter)，每當完成一次迴圈，累積值會增加一。
- 暫存器 R3 存放比較值 ，此數值用來與 R2 比較以判斷迴圈是否完成。
- 結果存放於暫存器 R1。
- 若欲使累加的數目更大或更小時，調整 R3 的內容即可。

流程圖


START
初始化暫存器：
R1 ← 00H
R2 ← 00H
R3 ← 0FH
R2 ← R2 + 1H
R1 ← R1 + R2
R2=R3(迴圈15次)?
否
是
END

程式碼(1/2)

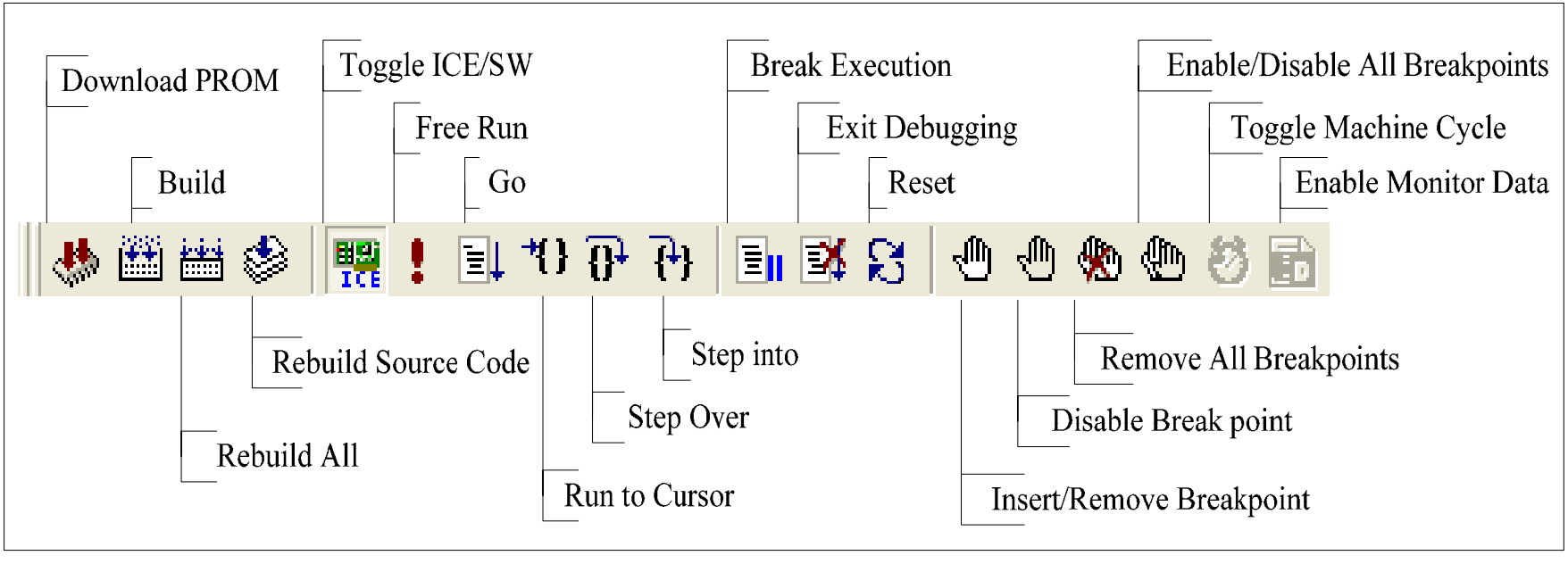
```
;==================================
;程式開始
;==================================
POWERON:
        R1= #0X00        ; 存放累加初值，加法運算的和存於此

        R2= #0X00        ; 清除累積器 R2 ，以便做「累加」運算

        R3= #0X0F        ; 存放次數值為 15 (比較值)，做 15 次
```

程式碼 (2/2)

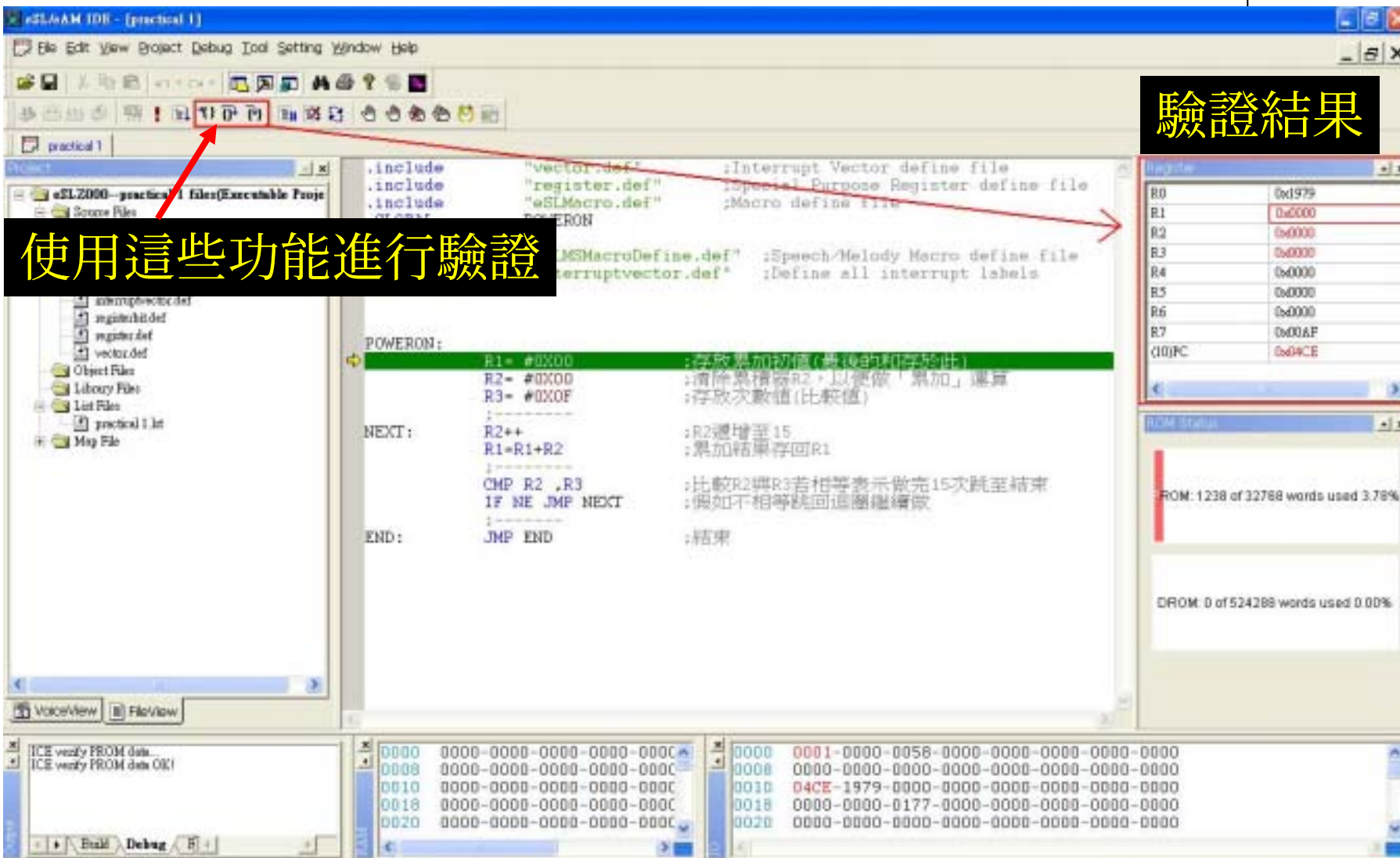
```
NEXT:
        R2++                    ; R2 遞增至 15
        R1=R1+R2                ; 累加結果存回 R1
        CMP R2, R3              ; 比較 R2 與 R3，若相等表示
                                ; 做完 15 次跳至結束

        IF NE JMP NEXT ；假如不相等跳回迴圈繼續執行

END:
        JMP     END
```

功能驗證(1/3)

- 編譯、執行及中斷等功能按鈕：

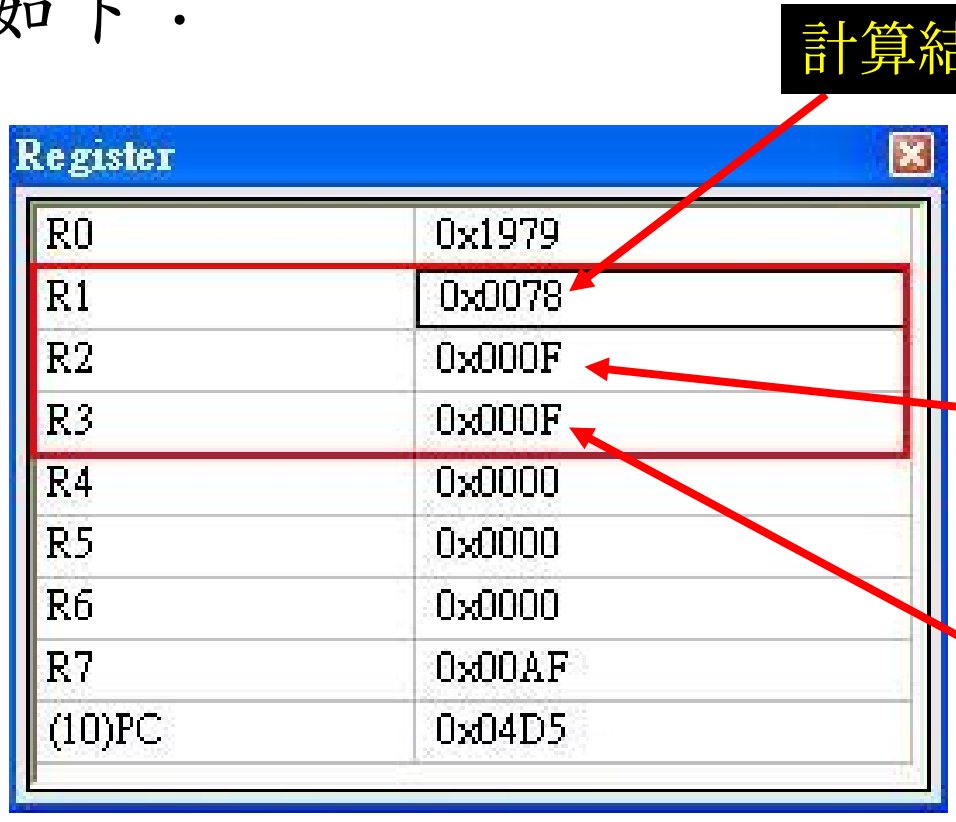


功能驗證 (2/3)


使用這些功能進行驗證
驗證結果
POWERON:
R1= #0X00 ;存放累加初值(最後的和存於此)
R2= #0X00 ;清除累積器R2，以便做「累加」運算
R3= #0X0F ;存放次數值(比較值)
NEXT:
R2++ ;R2遞增至15
R1=R1+R2 ;累加結果存回R1
CMP R2 ,R3 ;比較R2與R3若相等表示做完15次跳至結束
IF NE JMP NEXT ;假如不相等跳回迴圈繼續做
END:
JMP END ;結束

功能驗證 (3/3)

- 執行結果如下：



大綱

6.1 實習一：執行 1+2+3+……+14+15 的加法運算

6.2 實習二：數個 Byte 之加法運算

程式功能

流程圖

程式碼

功能驗證

6.3 實習三：清除內部 RAM 某段記憶體

6.4 實習四：將多個 BYTE 的資料內容左移 1 位元

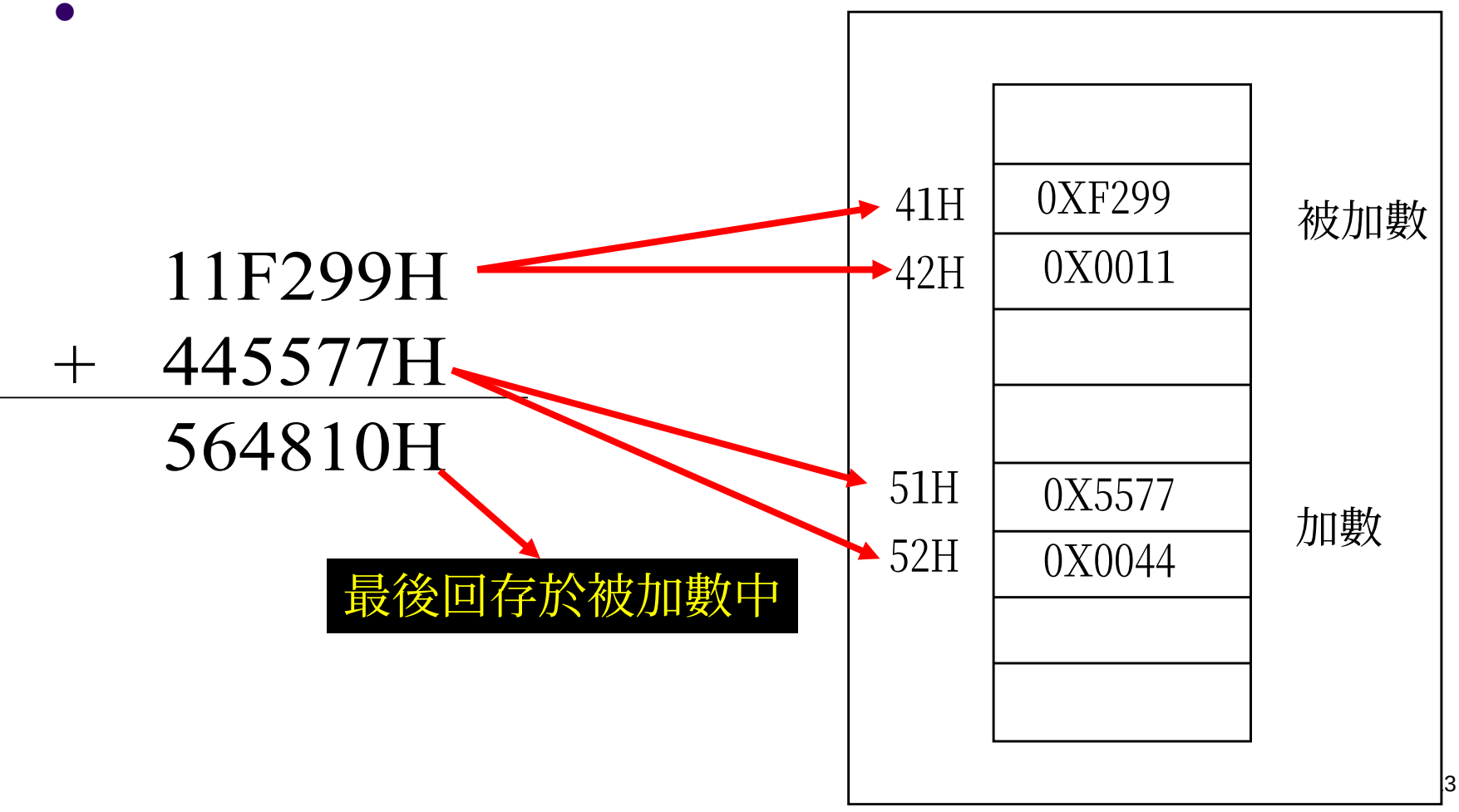
程式功能(1/2)

- 撰寫長度均為數個 Byte 之「被加數」與「加數」的加法運算程式，舉例如下：

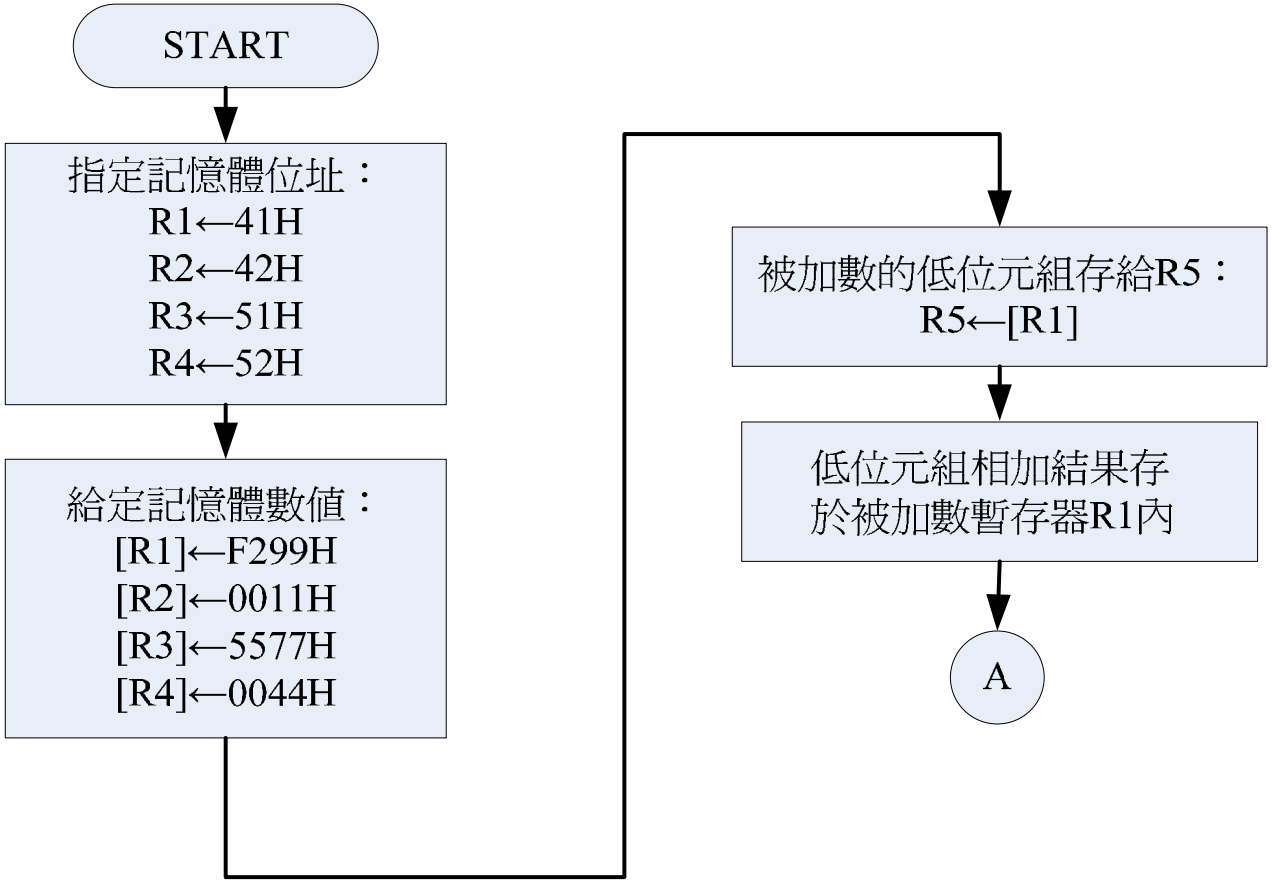
$$\begin{array}{r} 11F299H \\ +\quad 445577H \\ \hline 564810H \end{array}$$

- 利用記憶體不同位址分別存放資料方式進行數個 Byte 的加法運算
- 被加數的低位元組由暫存器 R1 來定址，高位元組由暫存器 R2 來定址
- 而加數的低位元組由暫存器 R3 來定址，高位元組由暫存器 R4 來定址。
- 相加之結果存放回被加數之位址。

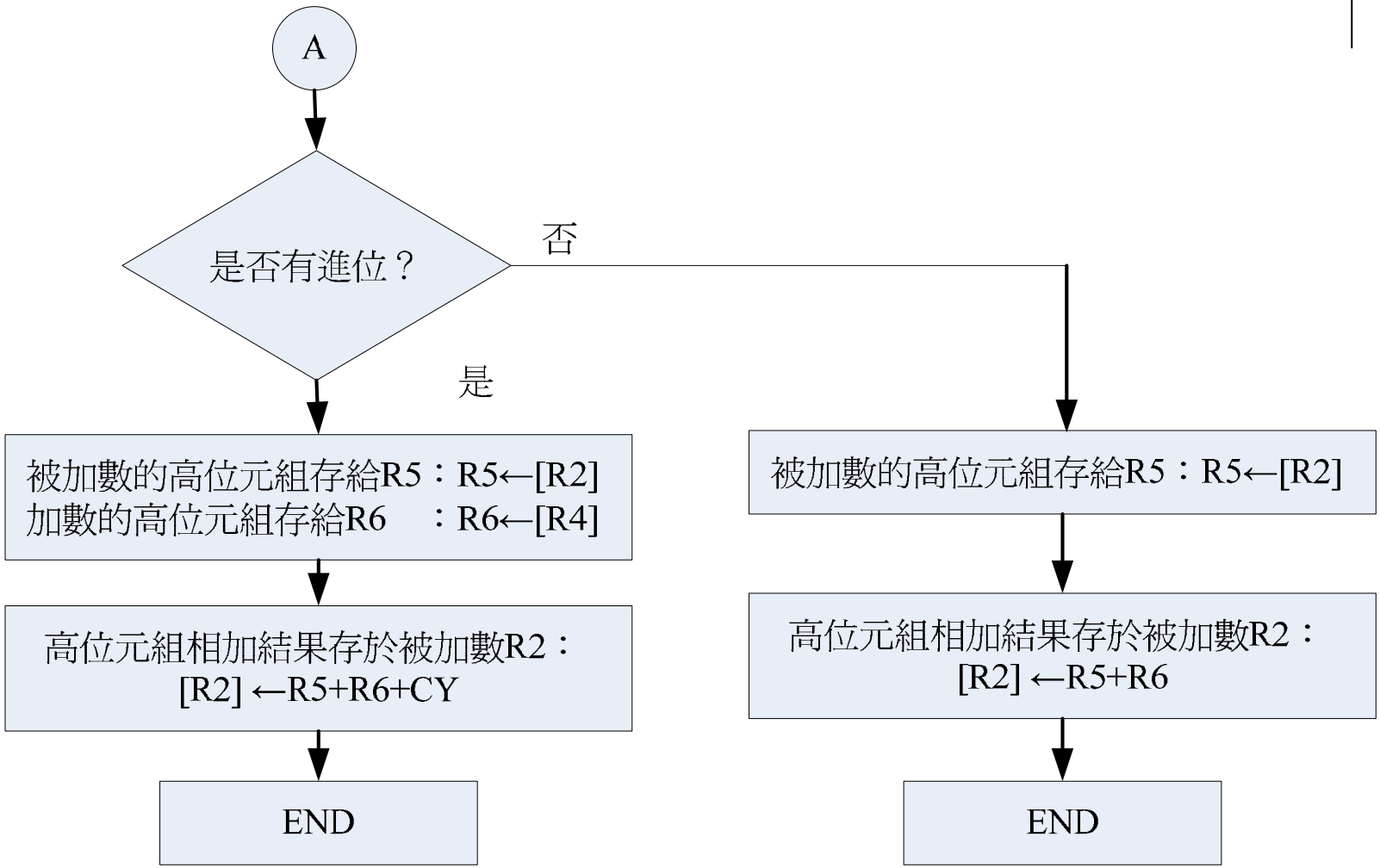
程式功能 (2/2)


11F299H
+ 445577H
564810H
最後回存於被加數中
41H
42H
51H
52H
0XF299
0X0011
0X5577
0X0044
被加數
加數

流程圖(1/2)


START
指定記憶體位址：
R1←41H
R2←42H
R3←51H
R4←52H
給定記憶體數值：
[R1]←F299H
[R2]←0011H
[R3]←5577H
[R4]←0044H
被加數的低位元組存給R5：
R5←[R1]
低位元組相加結果存
於被加數暫存器R1內
A

流程圖 (2/2)


A
是否有進位？
否
是
被加數的高位元組存給R5：R5←[R2]
加數的高位元組存給R6　：R6←[R4]
高位元組相加結果存於被加數R2：
[R2] ←R5+R6+CY
END
被加數的高位元組存給R5：R5←[R2]
高位元組相加結果存於被加數R2：
[R2] ←R5+R6
END

程式碼(1/3)

```
POWERON:
; ====================================
; 定義暫存器位址
; ====================================
                    R1=#0X41
                    R2=#0X42
                    R3=#0X51
                    R4=#0X52
```

程式碼 (1/2)

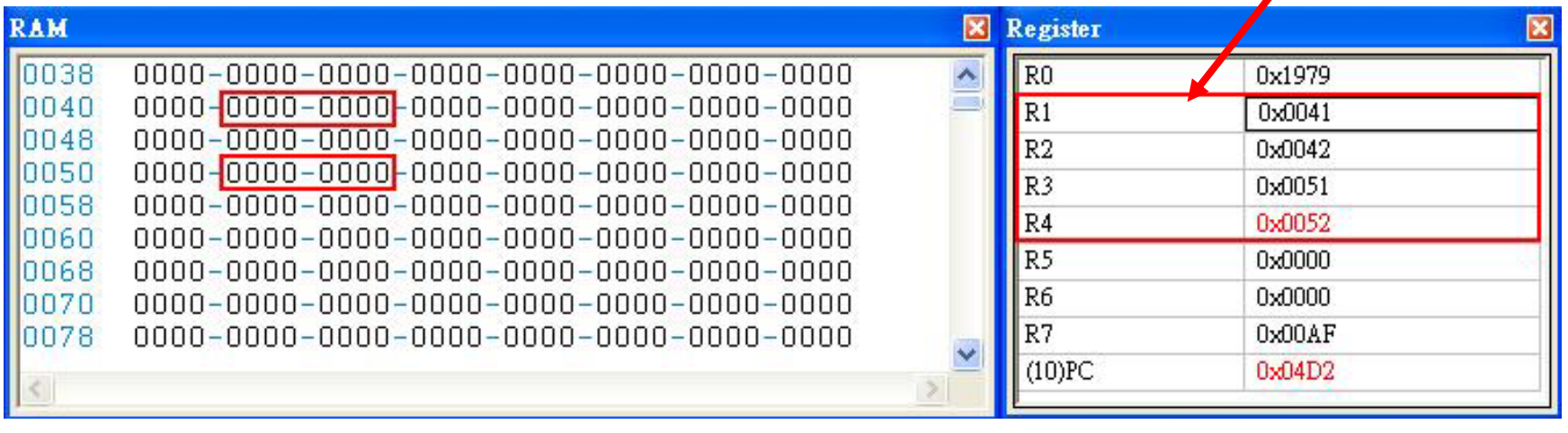
```
; ====================================
; 將值存給暫存器所指的位址
; ====================================
                    [R1]=#0XF299
                    [R2]=#0X0011
                    [R3]=#0X5577
                    [R4]=#0X0044
; ====================================
; 將 RAM 內的數值作加法運算，結果存放於被加數的位址
; ====================================
                    R5=[R1]             ; R5 = #0XF299
                    R6=[R3]             ; R6 = #0X5577
                    [R1]=R5+R6                  ; [R1] = #0X4810, CY=1
                    IF CS     JMP NEXT2         ; 有進位跳至 NEXT2
                                                ; 無進位直接做 NEXT1
```

程式碼 (2/2)

```
;======= NEXT1=========================
NEXT1:
    R5=[R2]
    R6=[R4]
    [R3]=R5+R6
END1:
    JMP     END1
;========NEXT2=========================
NEXT2:
    R5=[R2]                 ; R5 = #0X0011
    R6=[R4]                 ; R6 = #0X0044
    [R2]=R5+R6+C            ; [R2] = #0X0056
END2:
    JMP     END2
```

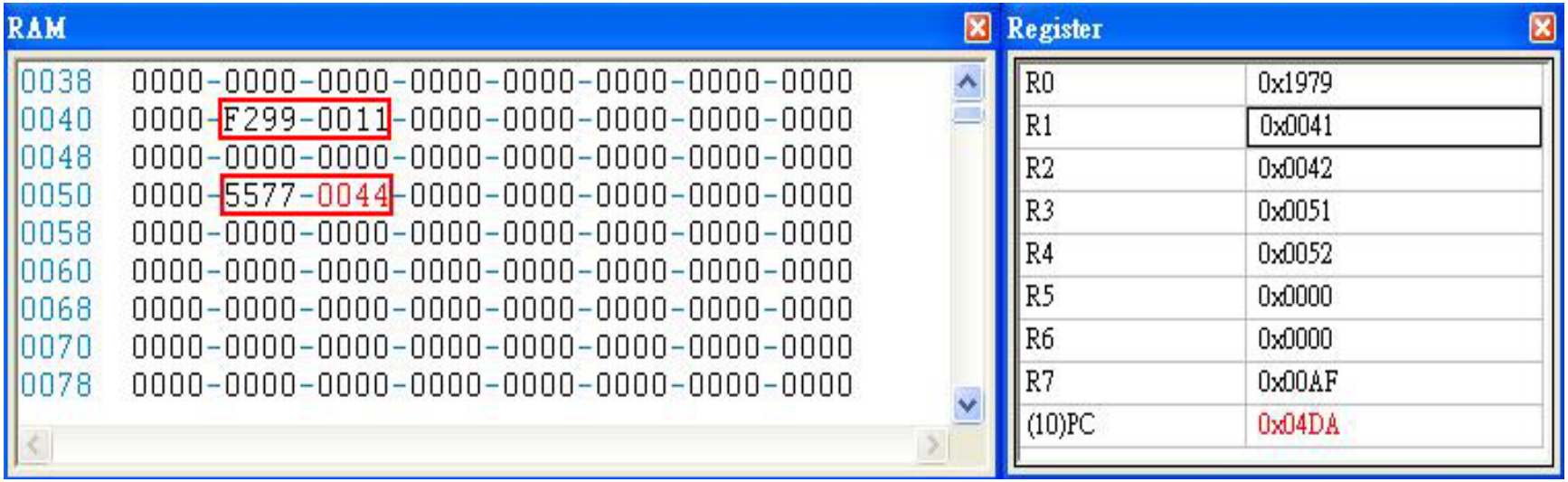
功能驗證(1/3)

- 定義暫存器與數值設定 ：



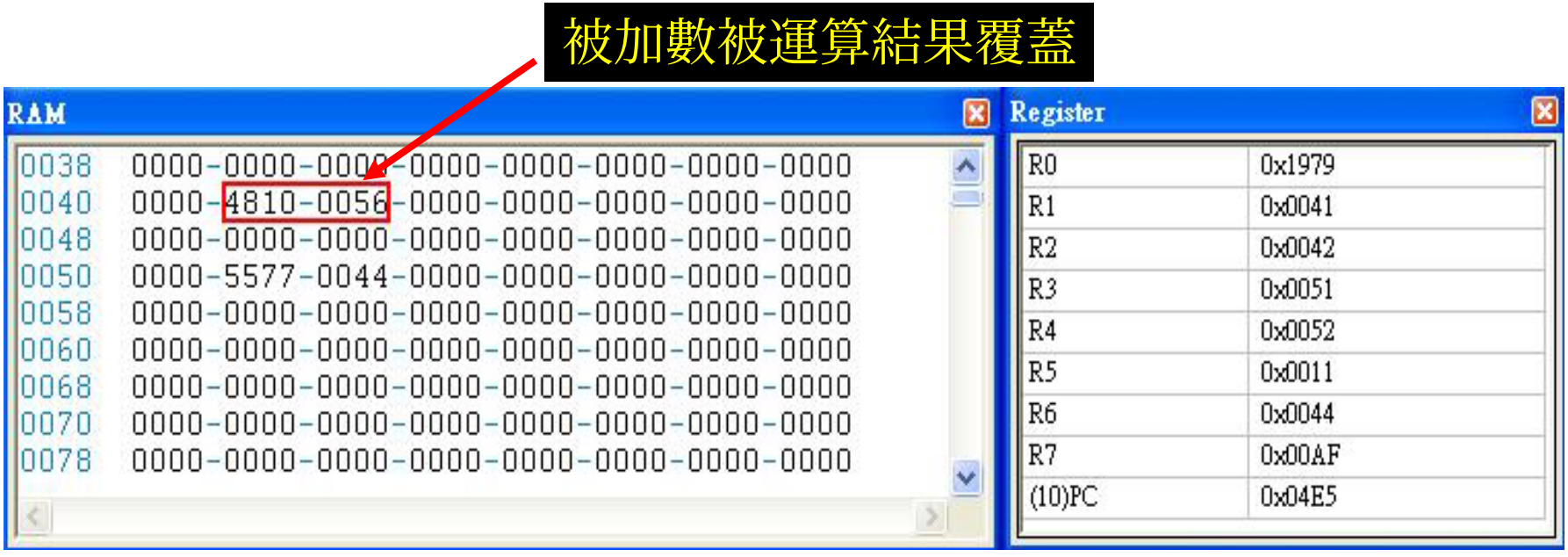
功能驗證 (2/3)

- 設定暫存器所指記憶體位址之資料 ：



功能驗證 (3/3)

- 執行完加法運算後的狀態 ：



大綱

6.1 實習一：執行 1+2+3+……+14+15 的加法運算

6.2 實習二：數個 Byte 之加法運算

6.3 實習三：清除內部 RAM 某段記憶體

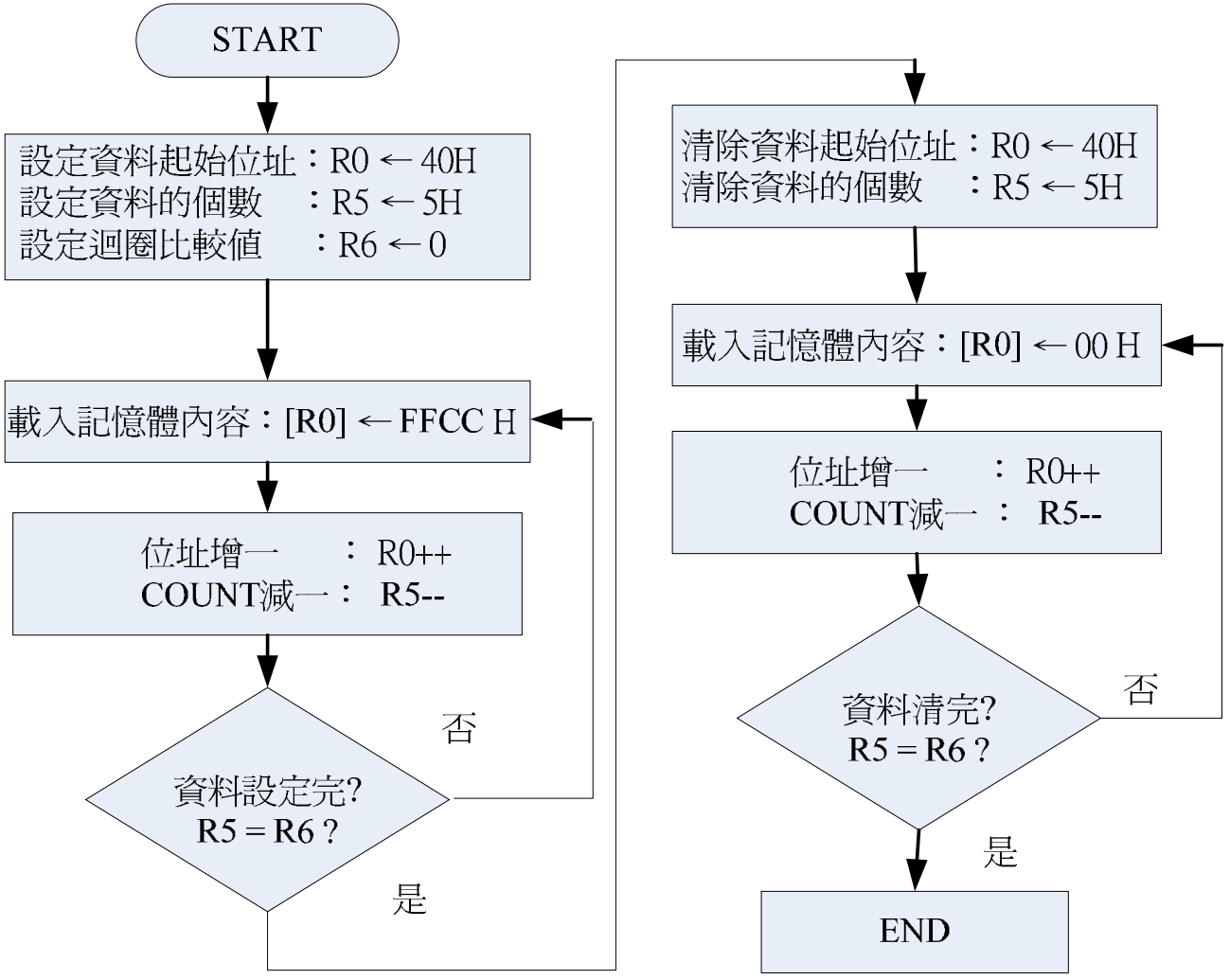
- **程式功能**
- **流程圖**
- **程式碼**
- **功能驗證**

6.4 實習四：將多個 BYTE 的資料內容左移 1 位元

程式功能

- 設計一程式清除 RAM 某段位址內的資料。
- 使用 LP_SET 迴圈於一段記憶體存入資料，本實習起始位址設為 40H，記憶體區段長度利用暫存器 R5 設定，
- 迴圈 LP_CLR 將資料清除為 0 。
- 其中欲設定/清除資料的起始位址存於暫存器 R0 ，設定/清除資料的個數存於暫存器 R5。

流程圖


START
設定資料起始位址：R0 ← 40H
設定資料的個數　：R5 ← 5H
設定迴圈比較值　：R6 ← 0
載入記憶體內容：[R0] ← FFCC H
位址增一　：R0++
COUNT減一：R5--
資料設定完?
R5 = R6 ?
否
是
清除資料起始位址：R0 ← 40H
清除資料的個數　：R5 ← 5H
載入記憶體內容：[R0] ← 00 H
位址增一　：R0++
COUNT減一：R5--
資料清完?
R5 = R6 ?
否
是
END

程式碼(1/3)

```
POWERON:
;=====================================
; 設定初值
;=====================================
        R0 = #0X40          ; 欲設定數值的起始位址 40H 存入 R0
        R5 = #0X05          ; 計數器，計數 5 個位址
        R6 = #0X00          ; 迴圈完畢的比較值
```

程式碼 (2/3)

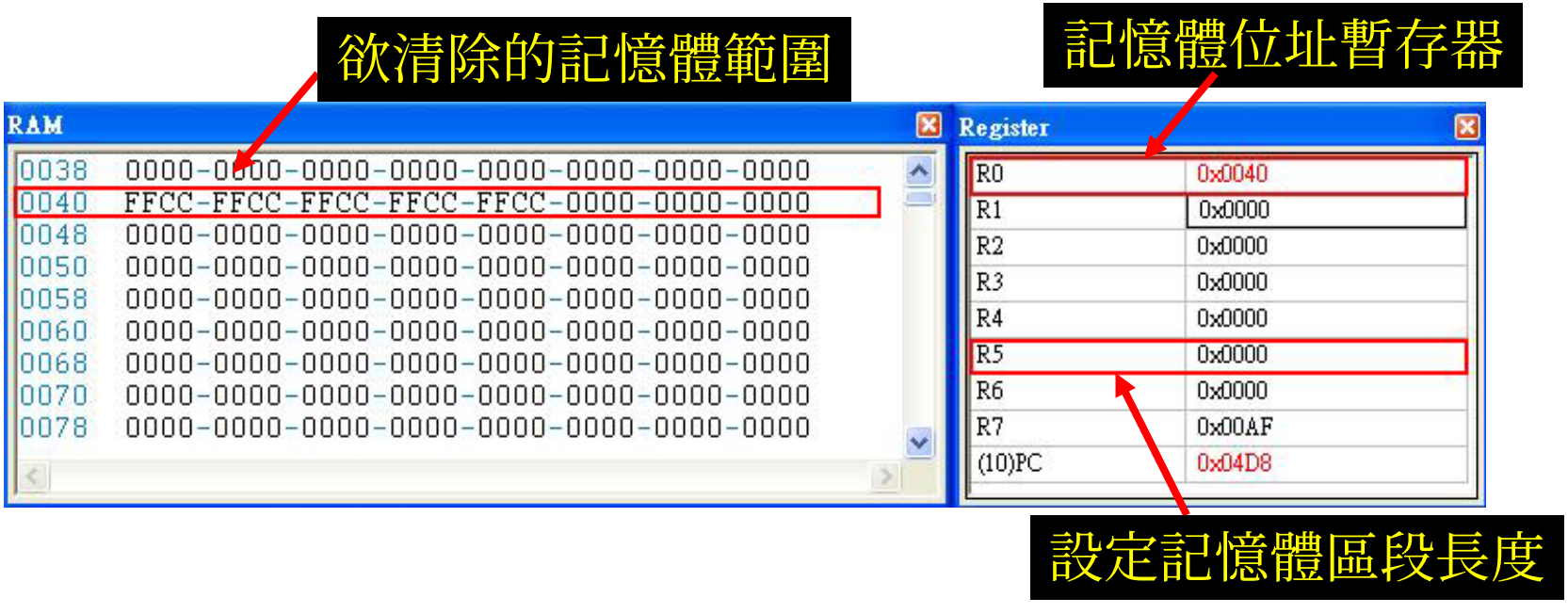
```
;=====================================
; 此段迴圈設定資料給 RAM 的位址
;=====================================
LP_SET:
    [R0] = #0XFFCC          ; 將資料丟給 R0 所指的位址
    R0++                    ; 位址增一
    R5--                    ; 計數值 COUNT 減一
    CMP  R5, R6    ; 檢測 5 個資料是否配置完成，
                            ; 若完成繼續以下程式
    IF  NE JMP LP_SET       ; 若未配置完成則跳回LP_SET 繼續配置
    R0 = #0X40              ; 欲清除之資料的起始位址40H 存入 R0
    R5 = #0X05              ; 計數器，計數 5 個位址
```

程式碼 (3/3)

```
; ===================================
; 此段迴圈將 RAM 位址的部分區段清除為 0
; ===================================
LP_CLR:
    [R0]= #0X00              ; 將資料存入 R0 所指的位址，即清除資料
    R0++                     ; 位址增一
    R5--                     ; 計數值 COUNT 減一
    CMP R5,R6                ; 比較,若資料未清除完畢繼續做程式 (END)

    IF  NE  JMP  LP_CLR      ; 若未清除完成則跳回LP_CLR 繼續清除

END:
    JMP     END              ; END
; ===================================
```

功能驗證(1/2)

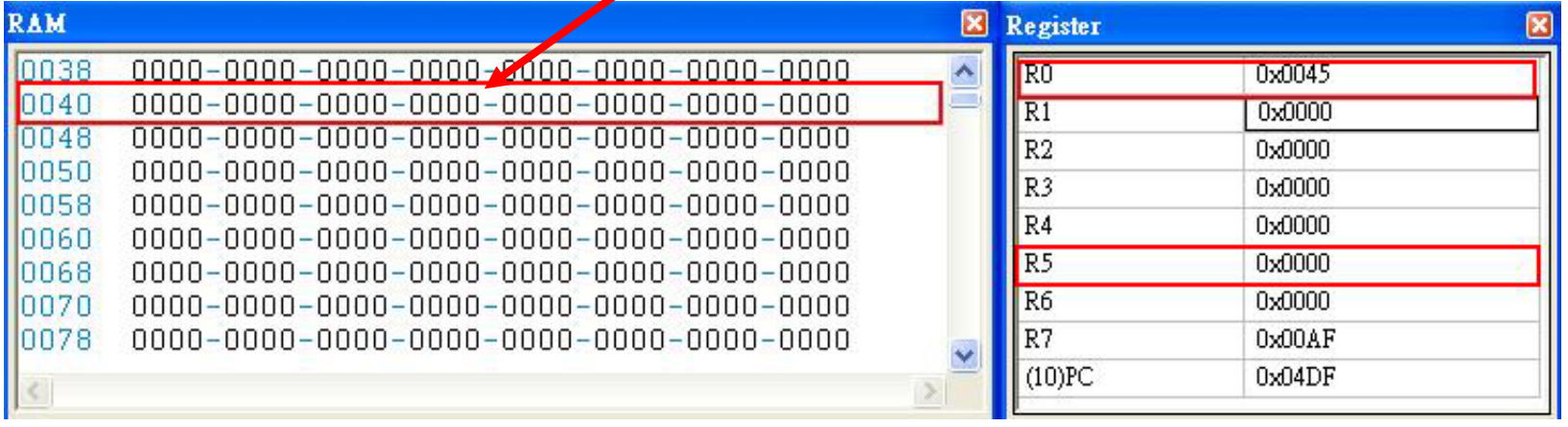
- 設定記憶體資料 ：



功能驗證 (2/2)

- 清除記憶體資料 ：

依序被清除，而不是一次就全部清除



大綱

6.1 實習一：執行 1+2+3+……+14+15 的加法運算

6.2 實習二：數個 Byte 之加法運算

6.3 實習三：清除內部 RAM 某段記憶體

6.4 實習四：將多個 BYTE 的資料內容左移 1 位元

程式功能

流程圖

程式碼

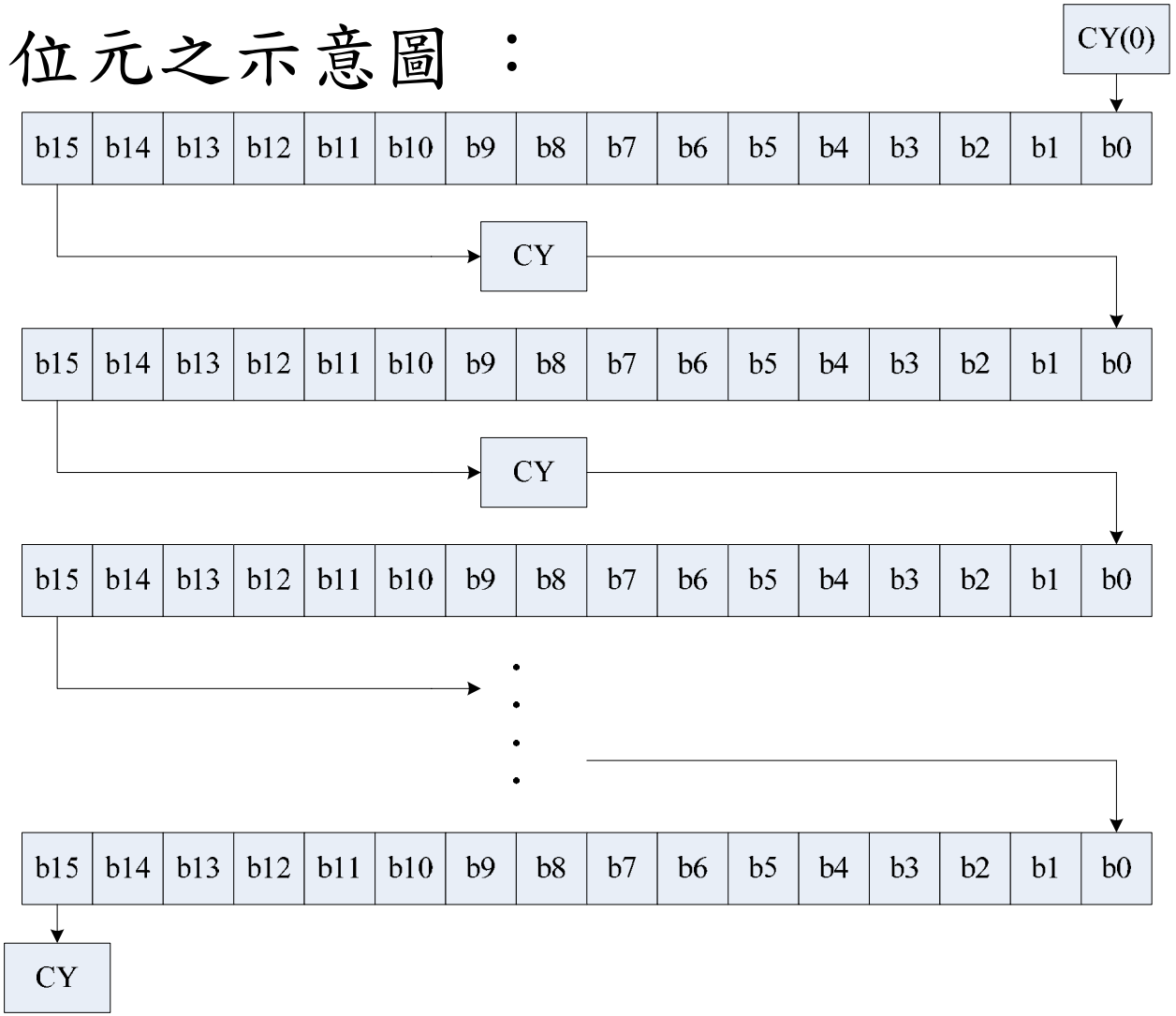
功能驗證

程式功能(1/2)

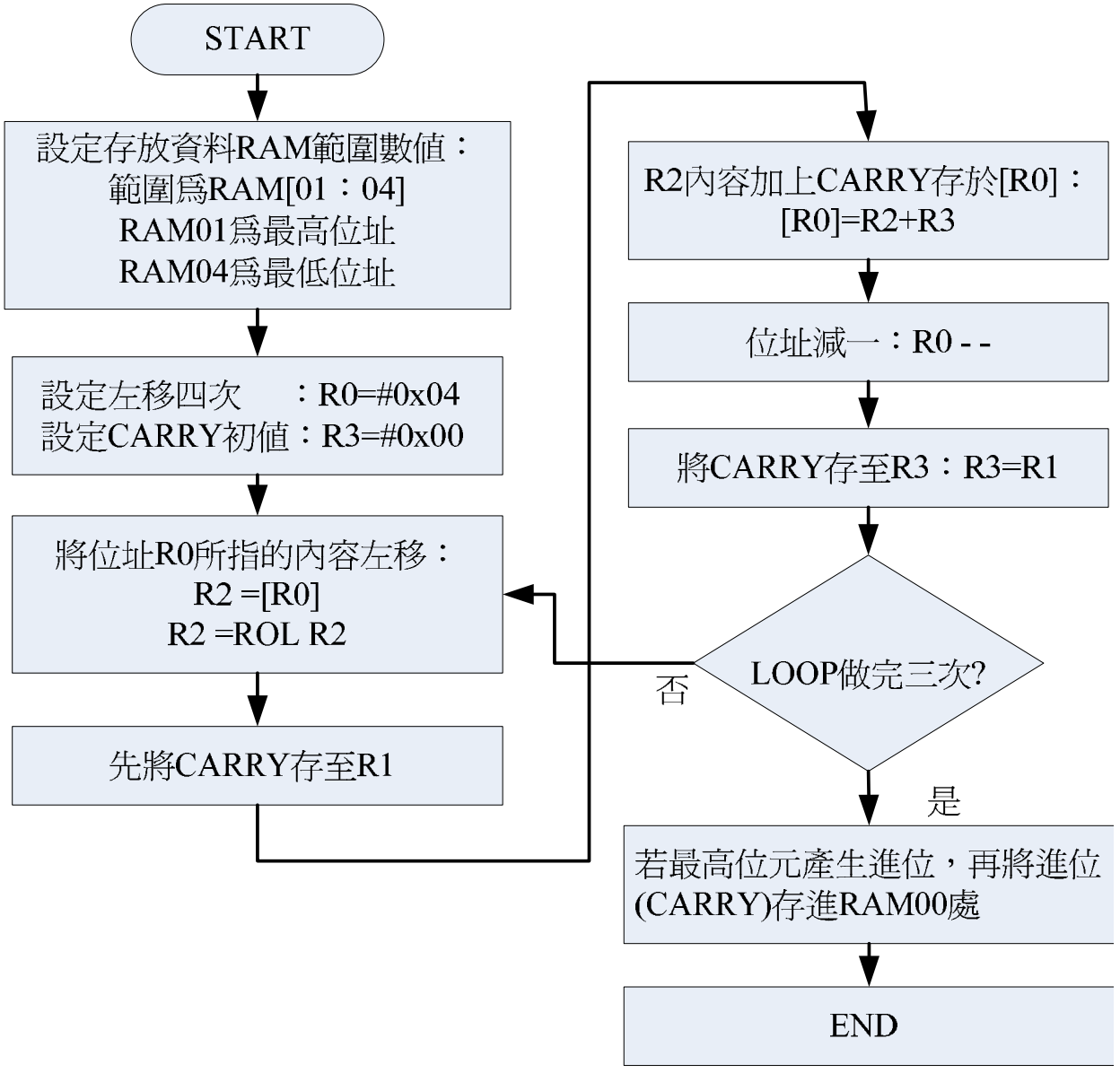
- 設計一程式將多個 BYTE 的內容一起往左移 1 位元 (可視為乘 2)。
- 為了測試數值是否能左移，故分別將四筆數值存放在 RAM[04：01]，其中 RAM01 為最高位址，RAM04 為最低位址。
- 將移位的位址由暫存器 R0 來定址，當其中的位元組發生進位時，較高的位元組必須增一。
- 當最高位元組發生進位時，要持續的進位，因此將 RAM00 當做最高位元產生進位的存放位址。

程式功能 (2/2)

- 左移 1 位元之示意圖 ：



流程圖


START
設定存放資料RAM範圍數值：
範圍爲RAM[01：04]
RAM01爲最高位址
RAM04爲最低位址
設定左移四次 ：R0=#0x04
設定CARRY初值：R3=#0x00
將位址R0所指的內容左移：
R2 =[R0]
R2 =ROL R2
先將CARRY存至R1
R2內容加上CARRY存於[R0]：
[R0]=R2+R3
位址減一：R0 - -
將CARRY存至R3：R3=R1
LOOP做完三次?
否
是
若最高位元產生進位，再將進位
(CARRY)存進RAM00處
END

程式碼(1/3)

```
POWERON:
; ===================================
; 設定(初始化)數值至 RAM[01:04]
; ===================================
        R0= #0x04                   ; 定址由RAM04開始
        [R0]= #0x8888               ; 存入數值
        R0--                        ; RAM位址減一
        [R0]= #0x8017               ; 存入數值
        R0--                        ; RAM位址減一
        [R0]= #0x45A0               ; 存入數值
        R0--                        ; RAM位址減一
        [R0]= #0xF0F6               ; 存入數值
```

程式碼 (2/3)

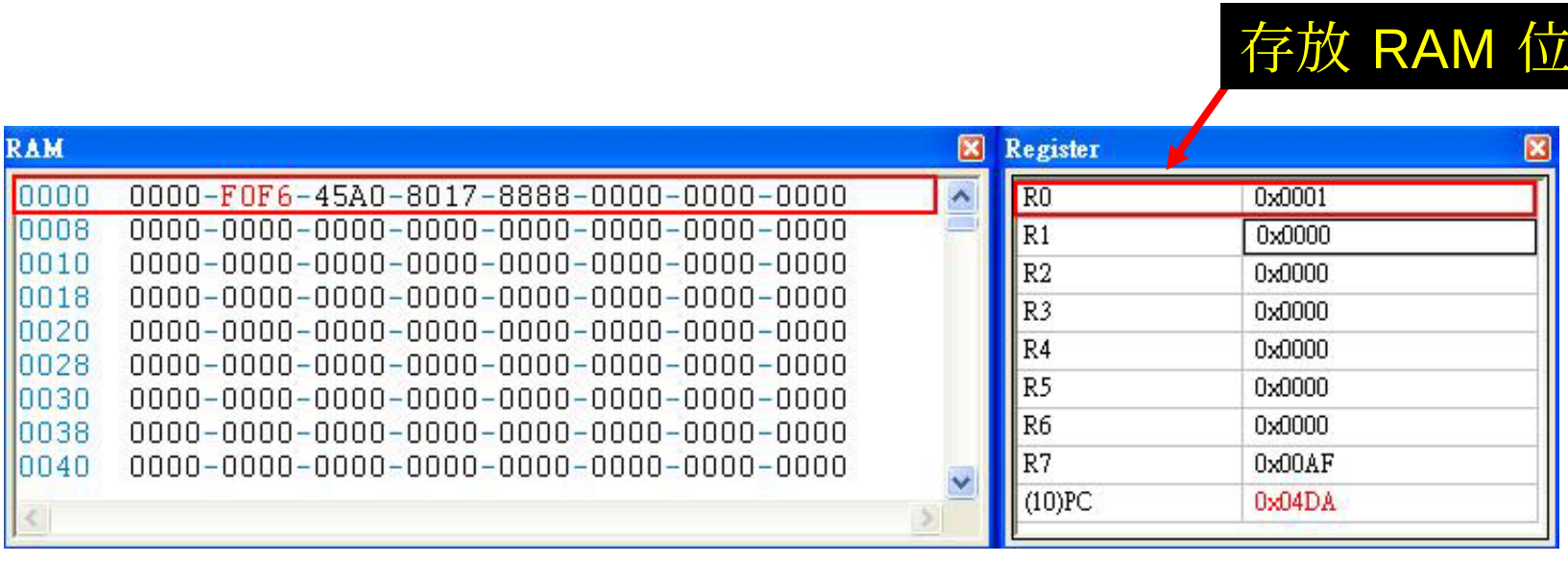
```
; ===================================
; 開始將 RAM[01:04] 左移
; ===================================
MAIN:
        R0= #0x04                ; 從最低位元開始左移
        R3= #0x00                ; CARRY 初值
        LOOP    #3               ; 開始進行左移迴圈
        R2=[R0]                  ; 將 R0 指定 RAM 位址的內容放
                                 ; 至 R2
        R2=SHL R2                ; 整個數值左移 1Bit

        R1=IO[SR]                ; 儲存 SR 暫存器內容
```

程式碼 (3/3)

```
        R1=R1 AND #0x01          ; 並清除不需要的狀態位元
        [R0]=R2+R3               ; R2 的內容加上 CARRY 放回指定
                                 ; RAM 位址內
        R0--                     ; RAM 位址減 1
        R3=R1                    ; CARRY 存至 R3
        .ENDL                    ; 如果完成則結束程式，若未完成
                                 ; 則繼續執行
        [R0]=R1                  ; 若最高 Bit 產生進位，將進位放
                                 ; 到記憶體位址
END:
        JMP      END             ; 結束
```

功能驗證(1/2)

- 設定記憶體資料 ：



功能驗證 (2/2)

- 清除記憶體資料左移一位元 ：