常用指令排行榜(六)

談完 JMP，就一定要談談 CALL，因為這兩個類別的指令，在 8051 組合語言裡所扮演的重要性是相輔相成的，然而 JMP 與 CALL 都不是正式的指令，在前幾篇文章中，我們已經介紹過 JMP 的用法，緊接著我們要來探討 CALL 的用法。

CALL 在組合語言裡使用得非常頻繁，越是大型的程式，用到它的機會就越多，更不用提是模組化的程式了。它和 JMP 的用法非常接近，連佔用的空間，所需的機械週期也一模一樣。如 JMP 有 AJMP，CALL 則有 ACALL；JMP 有 LJMP，CALL 則有 LCALL，然而在用法上最大的差別在於：JMP 是單向的，一旦離開了該位址就回不來了；CALL 是雙向的，離開了該位址等一下一定會再回來。底下是 CALL 的指令列表。

助憶碼	功能	呼叫範圍	使用空間	機器週期
ACALL	近程呼叫	2K Bytes	2 Bytes	2 Cycles
LCALL	遠程呼叫	64K Bytes	3 Bytes	2 Cycles

您或許會產生一個疑惑：基本的 JMP 不是有 SJMP 的指令嗎？為什麼 CALL 的指令裡沒有 SCALL？這牽涉到 CALL 的用法與性質，本文一開始就強調了 JMP 與 CALL 的最大差別，在於單向與雙向的運作差異，而雙向的運作需要將離開主程式時的絕對位址記錄在堆疊區裡，以便執行完副程式後再回到主程式。問題的解答就在這裡：因為 CALL 所需要的是絕對式的呼叫，SCALL 若比照 SJMP 的用法，相較之下不旦完全無法節省程式的空間，反而只是增加使用上的限制與不便。因為除了必須在堆疊區記錄離開主程式的位址外，可能還必須記錄對應的副程式啟始位址 (這是筆者假設的情況，畢竟在實際的指令裡並沒有安排 SCALL)，造成堆疊區的空間必須宣告得更大，而系統可應用的記憶體空間更小，間接地影響到系統運作的效能。如此一來，SCALL 既不能精簡程式空間，又不能加快系統運作的效能，所能呼叫的副程式範圍又更小，佔用的堆疊又更多，也就失去了 SCALL 存在的意義。

看了上面筆者天馬行空的描述後，會不會有點模糊的感覺？在這長長一串的文字裡其實包含了兩個很重要的觀念：***第一、CALL 的使用與堆疊的宣告是習習相關的，牽一髮則動全身***。如果撰寫的程式必須使用好幾次連續的呼叫，那麼勢必會耗掉很多記憶體空間用來宣告堆疊區，降低了系統在應用上的靈活度，這是必須盡量避免的。***第二、***CALL 是雙向的指令，那麼一旦離開了就一定要回去，因此***在副程式裡一定要加入 RET 的指令***，不然就會看到你的系統誤動作頻頻出現的情況。另一個值得留意的是：**我們所寫的任何組合語言程式，一定要在第一次呼叫前設定 8051 堆疊 STACK 的擺放位置**，例如 MOV SP,#4FH，這代表 DATA MEMORY 的 50H　7FH 共 48 個 Bytes 是供系統堆疊使用，若你把其他變數擺在這個區域中，絕對會有當機或意外發生，請務必要避免這種情況。

CALL 的指令或許不多，然而在使用上卻有不少限制，下一篇文章，我們要利用程式範例，探討關於 CALL 的用法與堆疊宣告之間的關係，拭目以待吧！

旗威科技有限公司　地址：高雄市三民區昌裕街 18-1 號　網址：http://www.chipware.com.tw
技術專線：07-395-5152　技術支援傳真：07-395-5155　E-mail：chipware@chipware.com.tw