# 嵌入式微處理機系統

05_ARM 指令

國立台灣科技大學

Feb. 2008



## ARM指令集-分類

- ARM指令集可以分為六大類
  - 跳躍指令
  - 資料處理指令
  - 程式狀態暫存器（PSR）處理指令
  - 載入/儲存指令
  - 輔助運算器指令
  - 異常產生指令

# ARM指令及功能描述 (1/2)

| 注記符 | 指令功能描述 |
|---|---|
| ADC | 帶進位元加法指令 |
| ADD | 加法指令 |
| AND | 邏輯AND指令 |
| B | 跳躍指令 |
| BIC | 位元清零指令 |
| BL | 帶返回的跳躍指令 |
| BLX | 帶返回和狀態切換的跳躍指令 |
| BX | 帶狀態切換的跳躍指令 |
| CDP | 輔助運算器資料操作指令 |
| CMN | 比較反值指令 |
| CMP | 比較指令 |
| EOR | 互次或指令 |
| LDC | 記憶體到輔助運算器的資料傳輸指令 |
| LDM | 載入多個暫存器指令 |

| | |
|---|---|
| LDR | 記憶體到暫存器的資料傳輸指令 |
| MCR | 從ARM暫存器到輔助運算器暫存器的資料傳輸指令 |
| MLA | 乘加運算指令 |
| MOV | 資料傳送指令 |
| MRC | 從輔助運算器暫存器到ARM暫存器的資料傳輸指令 |
| MRS | 傳送CPSR或SPSR的內容到通用暫存器指令 |
| MSR | 傳送通用暫存器到CPSR或SPSR的指令 |
| MUL | 32位元乘法指令 |
| MLA | 32位元乘加指令 |
| MVN | 資料取反傳送指令 |
| ORR | 邏輯OR指令 |
| RSB | 逆向減法指令 |
| RSC | 帶借位的逆向減法指令 |



# ARM指令及功能描述 (2/2)

| | |
|---|---|
| SBC | 帶借位減法指令 |
| STC | 輔助運算器暫存器寫入記憶體指令 |
| STM | 多重記憶體字寫入指令 |
| STR | 暫存器到記憶體的資料傳輸指令 |
| SUB | 減法指令 |
| SWI | 軟體中斷指令 |
| SWP | 交換指令 |
| TEQ | 相等測試指令 |
| TST | 位元測試指令 |

# ARM指令集 - 條件域

- 當處理器工作在ARM state時，幾乎所有的指令均根據CPSR中條件碼的狀態和指令的條件域有條件的執行。
- 每一條ARM指令包含4位元的條件碼，位於指令的最高4位元[31:28]。條件碼共有16種，每種條件碼可用兩個字元表示，這兩個字元可以添加在指令助記符號的後面和指令同時使用。




# ARM指令集 - 條件域

| 條件碼 | 助記符尾碼 | 標 志 | 含 義 |
|---|---|---|---|
| 0000 | EQ | Z置位 | 相等 |
| 0001 | NE | Z清零 | 不相等 |
| 0010 | CS | C置位 | 無符號數大於或等於 |
| 0011 | CC | C清零 | 無符號數小於 |
| 0100 | MI | N置位 | 負數 |
| 0101 | PL | N清零 | 正數或零 |
| 0110 | VS | V置位 | 溢出 |
| 0111 | VC | V清零 | 未溢出 |
| 1000 | HI | C置位Z清零 | 無符號數大於 |
| 1001 | LS | C清零Z置位 | 無符號數小於或等於 |
| 1010 | GE | N等於V | 帶符號數大於或等於 |
| 1011 | LT | N不等於V | 帶符號數小於 |
| 1100 | GT | Z清零且（N等於V） | 帶符號數大於 |
| 1101 | LE | Z置位或（N不等於V） | 帶符號數小於或等於 |
| 1110 | AL | 忽略 | 無條件執行 |

- 例如，跳躍指令B可以加上尾碼EQ變為BEQ表示“相等則跳躍”，即當CPSR中的Z標誌置位元時發生跳躍。

# ARM指令的定址方式

- 立即定址
- 暫存器定址
- 暫存器間接定址
- 基底定址
- 多暫存器定址
- 相對定址
- 堆疊定址



# ARM指令的定址方式

- 目前ARM指令系統支援如下幾種常見的定址方式：
  - 立即定址
    - ADD R0,R0,#1 ; R0←R0＋1
    - ADD R0,R0,#0x3f ; R0←R0＋0x3f
    - 在以上兩條指令中，第二個來源運算元即為立即常數，要求以“#”為首碼，對於以十六進位表示的立即常數，還要求在“#”後加上“0x”或“&”
  - 暫存器定址
    - ADD R0,R1,R2 ; R0←R1＋R2
  - 暫存器間接定址
    - ADD R0,R1,[R2] ; R0←R1＋[R2]
    - LDR R0,[R1] ; R0←[R1]
    - STR R0,[R1] ; [R1]←R0

# ARM指令的定址方式

- **基底定址**
  - LDR R0,[R1,＃4] ；R0←[R1＋4]
  - LDR R0,[R1,＃4]！；R0←[R1＋4]、R1←R1＋4
  - LDR R0,[R1] ,＃4 ；R0←[R1]、R1←R1＋4
  - LDR R0,[R1,R2] ；R0←[R1＋R2]

- **多暫存器定址**
  - 一條指令可以完成多個暫存器值的傳送。這種定址方式可以用一條指令完成傳送最多16個通用暫存器的值
  - LDMIA R0, {R1,R2,R3,R4} ；R1←[R0]
  - ；R2←[R0＋4]
  - ；R3←[R0＋8]
  - ；R4←[R0＋12]



# ARM指令的定址方式

- **相對定址**
  - BL NEXT ；跳躍到副程式NEXT處執行
  - ……
  - NEXT
  - ……
  - MOV PC, LR ；從副程式返回

- **堆疊定址**
  - 堆疊是一種資料結構，按先進後出（First In Last Out，FILO）的方式工作，使用一個稱作堆疊指標的專用暫存器指示當前的操作位置，堆疊指標總是指向頂端。

# ARM-Instruction Sets

- ARM指令集
  - 跳躍指令
  - 資料處理指令
  - 乘法指令與乘加指令
  - 程式狀態暫存器存取指令
  - 載入/儲存指令
  - 多重資料載入/儲存指令
  - 資料交換指令
  - 移位元指令（操作）
  - 輔助運算器指令
  - 異常產生指令



# ARM指令集 - 跳躍指令

- 跳躍指令用於實現程式流程的跳躍，在ARM程式中有兩種方法可以實現程式流程的跳躍：
  - 使用專門的跳躍指令。
    - ARM指令集中的跳躍指令可以完成從當前指令向前或向後的32MB的位址空間的跳躍，包括以下4條指令：

      B 跳躍指令
      BL 帶返回的跳躍指令
      BLX 帶返回和狀態切換的跳躍指令
      BX 帶狀態切換的跳躍指令
  - 直接向程式計數器PC寫入跳躍位址值。
    - 通過向程式計數器PC寫入跳躍位址值，可以實現在4GB的位址空間中的任意跳躍，在跳躍之前結合使用：

      MOV LR，PC

      等類似指令，可以保存將來的返回位址值，從而實現在4GB連續的線性位址空間的副程式調用。

# ARM指令集 - 跳躍指令

- B指令
  - B指令的格式為： B{條件} 目標位址
  - B指令是最簡單的跳躍指令。一旦遇到一個 B 指令，ARM 處理器將立即跳躍到給定的目標位址，從那裡繼續執行。注意儲存在跳躍指令中的實際值是相對當前PC值的一個偏移量，而不是一個絕對位址，它的值由組譯器來計算（相對定址）。它是 24 位元有符號數，左移兩位元後有符號擴充為 32 位，表示的有效偏移為 26 位(前後32MB的位址空間)。
  - Ex：
    - B Label ；程式無條件跳躍到標號Label處執行
    - CMP R1，＃0；當CPSR暫存器中的Z條件碼置位元時，程式跳躍
    - BEQ Label ；到標號Label處執行



# ARM指令集 - 跳躍指令

- BL指令
  - BL指令的格式為：BL{條件} 目標位址
  - BL 是另一個跳躍指令，但跳躍之前，會在暫存器R14中保存PC的當前內容，因此，可以通過將R14 的內容重新載入到PC中，來返回到跳躍指令之後的那個指令處執行。該指令是實現副程式調用的一個基本但常用的手段。
  - Ex：
    - BL Label ；當程式無條件跳躍到標號
      ；Label處執行時，同時將當前
      ；的PC值保存到R14中

# ARM指令集 - 跳躍指令

- BX指令
  - BX指令的格式為：BX{條件} 目標位址
  - BX指令跳躍到指令中所指定的目標位址，目標位址處的指令既可以是ARM指令，也可以是Thumb指令。
- BLX指令
  - 其格式為： BLX 目標位址
  - BLX指令從ARM指令集跳躍到指令中所指定的目標位址，並將處理器的工作狀態有ARM狀態切換到Thumb狀態，該指令同時將PC的當前內容保存到暫存器R14中。因此，當副程式使用Thumb指令集，而調用者使用ARM指令集時，可以通過BLX指令實現副程式的調用和處理器工作狀態的切換。同時，副程式的返回可以通過將暫存器R14值複製到PC中來完成



# ARM指令集 - 資料處理指令

- 資料處理指令可分
  - 為資料傳送指令

    暫存器 ←→ 記憶體
  - 算術邏輯運算指令
    - 將運算結果保存在目的暫存器中
    - 更新CPSR中的相應條件標誌位元
  - 比較指令
    - 不保存運算結果，只更新CPSR中相應的條件標誌位元。

# ARM指令集 - 資料處理指令

- MOV指令
  - MOV指令的格式為：MOV{條件}{S}　目的暫存器，來源運算元
  - MOV指令可完成從另一個暫存器、被移位的暫存器或將一個立即常數載入到目的暫存器。其中S選項決定指令的操作是否影響CPSR中條件標誌位元的值，當沒有S時指令不更新CPSR中條件標誌位元的值。
  - Ex：
    - MOV　R1,R0　　；將暫存器R0的值傳送到暫存器R1
    - MOV　PC,R14　　；將暫存器R14的值傳送到PC，常用；於副程式返回
    - MOV　R1,R0，LSL＃3　　；將暫存器R0的值左移3位後；傳送到R1



# ARM指令集 - 資料處理指令

- MVN指令　(negative)
  - MVN指令的格式為：
    MVN{條件}{S}　目的暫存器，來源運算元
  - MVN指令可完成從另一個暫存器、被移位的暫存器、或將一個立即常數載入到目的暫存器。與MOV指令不同之處是在傳送之前按位被取反了，即把一個被取反的值傳送到目的暫存器中。其中S決定指令的操作是否影響CPSR中條件標誌位元的值，當沒有S時指令不更新CPSR中條件標誌位元的值。
  - Ex：
    - MVN　R0,＃0　；將立即常數0取反傳送到暫存；器R0中，完成後R0=-1

# ARM指令集 - 資料處理指令

- CMP指令
  - CMP指令的格式為：CMP{條件} 運算元1，運算元2
  - CMP指令用於把一個暫存器的內容和另一個暫存器的內容或立即常數進行比較，同時更新CPSR中條件標誌位元的值。該指令進行一次減法運算，但不儲存結果，只更改條件標誌位元。標誌位元表示的是運算元1與運算元2的關係(大、小、相等)，例如，當運算元1大於操作運算元2，則此後的有GT 尾碼的指令將可以執行。
  - Ex：
    - CMP R1, R0 ；將暫存器R1的值與暫存器R0的值相減， CPSR ~ R1-R0
    - CMP R1, #100 ；將暫存器R1的值與立即常數100相減， CPSR ~ R1-100



# ARM指令集 - 資料處理指令

- CMN指令
  - CMN指令的格式為：CMN{條件} 運算元1，運算元2
  - CMN指令用於把一個暫存器的內容和另一個暫存器的內容或立即常數取反後進行比較，同時更新CPSR中條件標誌位元的值。該指令實際完成運算元1和運算元2相加，並根據結果更改條件標誌位元。
  - Ex：
    - CMN R1，R0 ；將暫存器R1的值與暫存器R0的值相加，並根據結果設置CPSR的標誌位元
    - CMN R1，#100 ；將暫存器R1的值與立即常數100相加，並根據結果設置CPSR的標誌位元

# ARM指令集 - 資料處理指令

- TST指令
  - TST指令的格式為：TST{條件} 運算元1，運算元2
  - TST指令用於把一個暫存器的內容和另一個暫存器的內容或立即常數進行按位的AND運算，並根據運算結果更新CPSR中條件標誌位元的值。運算元1是要測試的資料，而運算元2是一個位遮罩，該指令一般用來檢測是否設置了特定的位。
  - Ex：
    - TST R1，＃%1　　；用於測試在暫存器R1中是否設置了
      ；最低位（%表示二進位數字）
    - TST R1，＃0xffe　　；將暫存器R1的值與立即常數0xffe按位元
      ；AND，並根據結果設置CPSR的標誌位元



# ARM指令集 - 資料處理指令

- TEQ指令
  - TEQ指令的格式為：TEQ{條件} 運算元1，運算元2
  - TEQ指令用於把一個暫存器的內容和另一個暫存器的內容或立即常數進行按位的XOR運算，並根據運算結果更新CPSR中條件標誌位元的值。該指令通常用於比較運算元1和運算元2是否相等。
  - Ex：
    - TEQ R1，R2　　；將暫存器R1的值與暫存器R2的值按位元XOR
      ；並根據結果設置CPSR的標誌位元

# ARM指令集 - 資料處理指令

- ADD指令
  - ADD指令的格式為：

    ADD{條件}{S} 目的暫存器，運算元1，運算元2
  - ADD指令用於把兩個運算元相加，並將結果存放到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。
  - Ex：
    - ADD R0，R1，R2 ； R0 = R1 + R2
    - ADD R0，R1，#256 ； R0 = R1 + 256
    - ADD R0，R2，R3，LSL#1 ； R0 = R2 + (R3 << 1)



# ARM指令集 - 資料處理指令

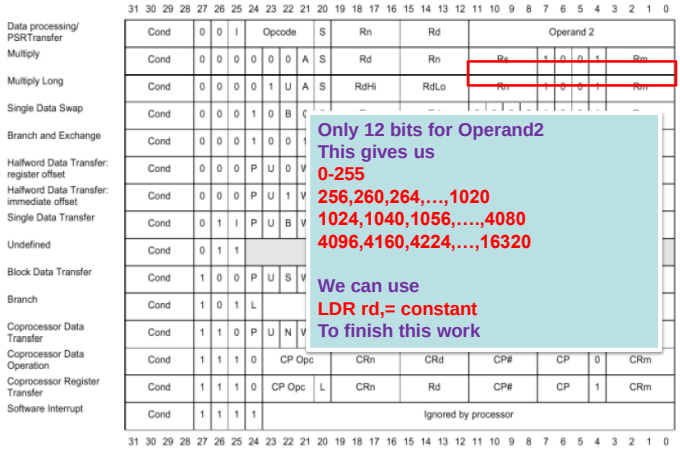
- ADD指令
  - ADD指令的格式為：

    ADD{條件}{S} 目的暫存器，運算元1，運算元2
  - ADD指令用於把兩個運算元相加，並將結果存放到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。
  - Ex：
    - ADD R0，R1，R2 ； R0 = R1 + R2
    - ADD R0，R1，#256 ； R0 = R1 + 256
    - ADD R0，R2，R3，LSL#1 ； R0 = R2 + (R3 << 1)

Compiler…Error!...Why?

But…..
ADD R0,R1,#257

# Data Processing Instructions (cont.)






# ARM指令集 - 資料處理指令

- ADC指令
  - ADC指令的格式為：ADC{條件}{S} 目的暫存器，運算元1，運算元2
  - ADC指令用於把兩個運算元相加，再加上CPSR中的C條件標誌位元的值，並將結果存放到目的暫存器中。它使用一個進位元標誌位元，這樣就可以做比32位大的數的加法，注意不要忘記設置S尾碼來更改進位元標誌。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。
  - 以下指令序列完成兩個128位數的加法，第一個數由高到低存放在暫存器R7～R4，第二個數由高到低存放在暫存器R11～R8，運算結果由高到低存放在暫存器R3～R0：
    - ADDS R0，R4，R8 ；加低端的字
    - ADCS R1，R5，R9 ；加第二個字，帶進位
    - ADCS R2，R6，R10 ；加第三個字，帶進位
    - ADC R3，R7，R11 ；加第四個字，帶進位

# ARM指令集 - 資料處理指令

- SUB指令
  - SUB指令的格式為：

    SUB{條件}{S} 目的暫存器，運算元1，運算元2
  - SUB指令用於把運算元1減去運算元2，並將結果存放到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。該指令可用於有符號數或無符號數的減法運算。
  - Ex：
    - SUB R0，R1，R2 ； R0 = R1 - R2
    - SUB R0，R1，#256 ； R0 = R1 - 256
    - SUB R0，R2，R3，LSL#1 ； R0 = R2 - (R3 << 1)



# ARM指令集 - 資料處理指令

- SBC指令
  - SBC指令的格式為：

    SBC{條件}{S} 目的暫存器，運算元1，運算元2
  - SBC指令用於把運算元1減去運算元2，再減去CPSR中的C條件標誌位元的反碼，並將結果存放到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。該指令使用進位元標誌來表示借位，這樣就可以做大於32位的減法，注意不要忘記設置S尾碼來更改進位元標誌。該指令可用於有符號數或無符號數的減法運算。
  - Ex：
    - SBCS R0，R1，R2 ； R0 = R1 - R2 - ！C
      ；並根據結果設置CPSR
      ；的進位元標誌位元

# ARM指令集 - 資料處理指令

- RSB指令
  - RSB指令的格式為：

    RSB{條件}{S} 目的暫存器，運算元1，運算元2
  - RSB指令稱為逆向減法指令，用於把運算元2減去運算元1，並將結果存放到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。該指令可用於有符號數或無符號數的減法運算。
  - Ex：
    - RSB R0，R1，R2 ； R0 = R2 – R1
    - RSB R0，R1，#256 ； R0 = 256 – R1
    - RSB R0，R2，R3，LSL#1 ； R0 = (R3 << 1) - R2



# ARM指令集 - 資料處理指令

- RSC指令
  - RSC指令的格式為：

    RSC{條件}{S} 目的暫存器，運算元1，運算元2
  - RSC指令用於把運算元2減去運算元1，再減去CPSR中的C條件標誌位元的反碼，並將結果存放到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。該指令使用進位元標誌來表示借位，這樣就可以做大於32位的減法，注意不要忘記設置S尾碼來更改進位元標誌。該指令可用於有符號數或無符號數的減法運算。
  - Ex：
    - RSC R0，R1，R2 ； R0 = R2 – R1 - ！C

# ARM指令集 - 資料處理指令

- AND指令
  - AND指令的格式為：

    AND{條件}{S} 目的暫存器，運算元1，運算元2
  - AND指令用於在兩個運算元上進行邏輯AND運算，並把結果放置到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。該指令常用於遮罩運算元1的某些位。
  - Ex：
    - AND R0，R0，#3 ；該指令保持R0的0、1位，其餘位清零。



# ARM指令集 - 資料處理指令

- ORR指令
  - ORR指令的格式為：

    ORR{條件}{S} 目的暫存器，運算元1，運算元2
  - ORR指令用於在兩個運算元上進行邏輯OR運算，並把結果放置到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。該指令常用於設置運算元1的某些位。
  - Ex：
    - ORR R0，R0，#3 ；該指令設置R0的0、1位，其餘位保持不變。

# ARM指令集 - 資料處理指令

- EOR指令
  - EOR指令的格式為：

    EOR{條件}{S} 目的暫存器，運算元1，運算元2
  - EOR指令用於在兩個運算元上進行邏輯異或運算，並把結果放置到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。該指令常用於反轉運算元1的某些位。
  - Ex：
    - EOR R0，R0，＃3 ；該指令反轉R0的0、1位，其餘位；保持不變。



# ARM指令集 - 資料處理指令

- BIC指令
  - BIC指令的格式為：

    BIC{條件}{S} 目的暫存器，運算元1，運算元2
  - BIC指令用於清除運算元1的某些位，並把結果放置到目的暫存器中。運算元1應是一個暫存器，運算元2可以是一個暫存器，被移位的暫存器，或一個立即常數。運算元2為32位的遮罩，如果在遮罩中設置了某一位，則清除這一位。未設置的遮罩位保持不變。
  - Ex：
    - BIC R0，R0，＃%1011 ；該指令清除 R0 中的位 0、；1、和 3，其餘的位保持不變。

# ARM指令集 -乘法指令與乘加指令

- ARM微處理器支援的乘法指令與乘加指令共有6條

| | | |
|---|---|---|
| 32位元 | MUL | 32位元乘法指令 |
| | MLA | 32位元乘加指令 |
| 64位元 | SMULL | 64位元有號數乘法指令 |
| | SMLAL | 64位元有號數乘加指令 |
| | UMULL | 64位元無號數乘法指令 |
| | UMLAL | 64位元無號數乘加指令 |

- 限制條件：
  - 指令中的所有運算元、目的暫存器必須為通用暫存器
  - 不能對運算元使用立即常數或被移位的暫存器
  - 目的暫存器和運算元1必須是不同的暫存器。



# ARM指令集 -乘法指令與乘加指令

- MUL指令
  - MUL指令的格式為：

    MUL{條件}{S}　　目的暫存器，運算元1，運算元2
  - MUL指令完成將運算元1與運算元2的乘法運算，並把結果放置到目的暫存器中，同時可以根據運算結果設置CPSR中相應的條件標誌位元。其中，運算元1和運算元2均為32位元的有符號數或無符號數。
  - Ex：

```
MUL     R0，R1，R2    ；R0 = R1 x R2
MULS    R0，R1，R2    ；R0 = R1 x R2，同時設置
                      ；CPSR中的相關條件標誌位
                      ；元
```

# ARM指令集 -乘法指令與乘加指令

- MLA指令
  - MLA指令的格式為：

    MLA{條件}{S}　　目的暫存器，運算元1，運算元2，運算元3
  - MLA指令完成將運算元1與運算元2的乘法運算，再將乘積加上操作數3，並把結果放置到目的暫存器中，同時可以根據運算結果設置CPSR中相應的條件標誌位元。其中，運算元1和運算元2均為32位元的有符號數或無符號數。
  - Ex：
    - MLA　R0，R1，R2，R3　　；R0 = R1 × R2 + R3
    - MLAS　R0，R1，R2，R3　　；R0 = R1 × R2 + R3，同時設置CPSR中的相關條件標誌位元



# ARM指令集 -乘法指令與乘加指令

- SMULL指令
  - SMULL指令的格式為：

    SMULL{條件}{S}　　目的暫存器Low，目的暫存器High，運算元1，運算元2
  - SMULL指令完成將運算元1與運算元2的乘法運算，並把結果的低32位元放置到目的暫存器Low中，結果的高32位元放置到目的暫存器High中，同時可以根據運算結果設置CPSR中相應的條件標誌位元。其中，運算元1和運算元2均為32位元的有號數。
  - Ex：
    - SMULL　R0，R1，R2，R3；R0 =（R2 × R3）的低32位
      ；R1 =（R2 × R3）的高32位

# ARM指令集 -乘法指令與乘加指令

- SMLAL指令
  - SMLAL指令的格式為：

    SMLAL{條件}{S}　目的暫存器Low，目的暫存器低High，運算元1，運算元2
  - SMLAL指令完成將運算元1與運算元2的乘法運算，並把結果的低32位元同目的暫存器Low中的值相加後又放置到目的暫存器Low中，結果的高32位元同目的暫存器High中的值相加後又放置到目的暫存器High中，同時可以根據運算結果設置CPSR中相應的條件標誌位元。其中，運算元1和運算元2均為32位元的有符號數。
  - 對於目的暫存器Low，在指令執行前存放64位元加數的低32位元，指令執行後存放結果的低32位。
  - 對於目的暫存器High，在指令執行前存放64位元加數的高32位元，指令執行後存放結果的高32位。
  - Ex：
    - SMLAL　R0，R1，R2，R3　　；R0 =（R2 x R3）的低32位 + R0
      ；R1 =（R2 x R3）的高32位 + R1



# ARM指令集 -乘法指令與乘加指令

- UMULL指令
  - UMULL指令的格式為：

    UMULL{條件}{S}　目的暫存器Low，目的暫存器低High，運算元1，運算元2
  - UMULL指令完成將運算元1與運算元2的乘法運算，並把結果的低32位元放置到目的暫存器Low中，結果的高32位元放置到目的暫存器High中，同時可以根據運算結果設置CPSR中相應的條件標誌位元。其中，運算元1和運算元2均為32位元的無符號數。
  - Ex：
    - UMULL　R0，R1，R2，R3　　；R0 =（R2 x R3）的低32位
      ；R1 =（R2 x R3）的高32位

# ARM指令集 -乘法指令與乘加指令

- UMLAL指令
  - UMLAL指令的格式為：

    UMLAL{條件}{S} 目的暫存器Low，目的暫存器低High，運算元1，運算元2
  - UMLAL指令完成將運算元1與運算元2的乘法運算，並把結果的低32位元同目的暫存器Low中的值相加後又放置到目的暫存器Low中，結果的高32位元同目的暫存器High中的值相加後又放置到目的暫存器High中，同時可以根據運算結果設置CPSR中相應的條件標誌位元。其中，運算元1和運算元2均為32位元的無符號數。
  - 對於目的暫存器Low，在指令執行前存放64位元加數的低32位元，指令執行後存放結果的低32位。
  - 對於目的暫存器High，在指令執行前存放64位元加數的高32位元，指令執行後存放結果的高32位。
  - Ex：
    - UMLAL R0，R1，R2，R3 ；R0 =（R2 x R3）的低32位 + R0
      ；R1 =（R2 x R3）的高32位 + R1



# ARM指令集 - 程式狀態暫存器存取指令

- ARM微處理器支援程式狀態暫存器存取指令，用於在程式狀態暫存器和通用暫存器之間傳送資料，程式狀態暫存器存取指令包括以下兩條：
  - MRS 程式狀態暫存器到通用暫存器的資料傳送指令
  - MSR 通用暫存器到程式狀態暫存器的資料傳送指令

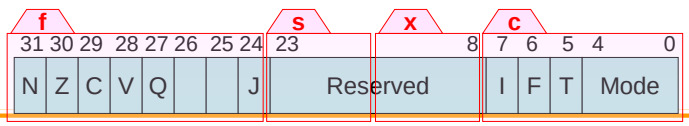
# ARM指令集 - 程式狀態暫存器存取指令

- MRS指令
  - MRS指令的格式為：
    MRS{條件}　通用暫存器，程式狀態暫存器（CPSR或SPSR）
  - MRS指令用於將程式狀態暫存器的內容傳送到通用暫存器中。該指令一般用在以下幾種情況：
    - 當需要改變程式狀態暫存器的內容時，可用MRS將程式狀態暫存器的內容讀入通用暫存器，修改後再寫回程式狀態暫存器。
    - 當在異常處理或進程切換時，需要保存程式狀態暫存器的值，可先用該指令讀出程式狀態暫存器的值，然後保存。
  - Ex：
    - MRS R0，CPSR　；傳送CPSR的內容到R0
    - MRS R0，SPSR　；傳送SPSR的內容到R0



# ARM指令集 - 程式狀態暫存器存取指令

- MSR指令
  - MSR指令的格式為：
    MSR{條件}　程式狀態暫存器（CPSR或SPSR）_<域>，運算元
  - MSR指令用於將運算元的內容傳送到程式狀態暫存器的特定域中。其中，運算元可以為通用暫存器或立即常數。<域>用於設置程式狀態暫存器中需要操作的位元，32位元的程式狀態暫存器可分為4個域：
    - [31：24]為條件標誌位元域，用f表示；
    - [23：16]為狀態位元域，用s表示；
    - [15：8]為擴展位域，用x表示；
    - [7：0]為控制位域，用c表示；
  - 該指令通常用於恢復或改變程式狀態暫存器的內容，在使用時，一般要在MSR指令中指明將要操作的域。
  - Ex：
    - MSR CPSR，R0　；傳送R0的內容到CPSR
    - MSR SPSR，R0　；傳送R0的內容到SPSR
    - MSR CPSR_c，R0　；傳送R0的內容到SPSR，但僅僅修改 ；CPSR中的控制位域



# ARM指令集 - 載入/儲存指令

- ARM微處理器支援載入/儲存指令用於在暫存器和記憶體之間傳送資料，載入指令用於將記憶體中的資料傳送到暫存器，儲存指令則完成相反的操作。常用的載入儲存指令如下：
  - LDR　字資料載入指令
  - LDRB　位元組資料載入指令
  - LDRH　半字資料載入指令
  - STR　字資料儲存指令
  - STRB　位元組資料儲存指令
  - STRH　半字資料儲存指令



# ARM指令集 - 載入/儲存指令

- LDR指令
  - LDR指令的格式為：
    LDR{條件} 目的暫存器，<記憶體位址>
  - LDR指令用於從記憶體中將一個32位元的資料傳送到目的暫存器中。該指令通常用於從記憶體中讀取32位元的資料到通用暫存器，然後對資料進行處理。當程式計數器PC作為目的暫存器時，指令從記憶體中讀取的字資料被當作目的地址，從而可以實現程式流程的跳躍。該指令在程式設計中比較常用，且定址方式靈活多樣。
  - Ex：
    - LDR　R0，[R1]　；將記憶體位址為R1的資料讀入暫存器R0。
    - LDR　R0，[R1，R2]　；將記憶體位址為R1+R2的資料讀入暫存器R0。
    - LDR　R0，[R1，＃8]　；將記憶體位址為R1+8的資料讀入暫存器R0。
    - LDR　R0，[R1，R2]！　；將記憶體位址為R1+R2的資料讀入暫存器R0，並將新位址；R1＋R2寫入R1。
    - LDR　R0，[R1，＃8]！　；將記憶體位址為R1+8的資料讀入暫存器R0，並將新位址；R1＋8寫入R1。
    - LDR　R0，[R1]，R2　；將記憶體位址為R1的資料讀入暫存器R0，並將新位址；R1＋R2寫入R1。
    - LDR　R0，[R1，R2，LSL＃2]！；將記憶體位址為R1＋R2x4的資料讀入暫存器R0，；並將新位址R1＋R2x4寫入R1。
    - LDR　R0，[R1]，R2，LSL＃2　；將記憶體位址為R1的資料讀入暫存器R0，並將新位址；R1＋R2x4寫入R1。

# ARM指令集 - 載入/儲存指令

- LDRB指令
  - LDRB指令的格式為：LDR{條件}B 目的暫存器，<記憶體位址>
  - LDRB指令用於從記憶體中將一個8位元的位元組資料傳送到目的暫存器中，同時將暫存器的高24位清零。該指令通常用於從記憶體中讀取8位元的位元組資料到通用暫存器，然後對資料進行處理。當程式計數器PC作為目的暫存器時，指令從記憶體中讀取的資料被當作目的地址，從而可以實現程式流程的跳躍。
  - Ex：
    - LDRB R0，[R1] ；將記憶體位址為R1的位元組資料讀入暫存器R0，並將R0的高24位清零。
    - LDRB R0，[R1，＃8] ；將記憶體位址為R1＋8的位元組資料讀入暫存器R0，並將R0的高24位清零。



# ARM指令集 - 載入/儲存指令

- LDRH指令
  - LDRH指令的格式為：LDR{條件}H 目的暫存器，<記憶體位址>
  - LDRH指令用於從記憶體中將一個16位元的HalfWord資料傳送到目的暫存器中，同時將暫存器的高16位清零。該指令通常用於從記憶體中讀取16位元的HalfWord資料到通用暫存器，然後對資料進行處理。當程式計數器PC作為目的暫存器時，指令從記憶體中讀取的Word資料被當作目的地址，從而可以實現程式流程的跳躍。
  - Ex：
    - LDRH R0，[R1] ；將記憶體位址為R1的HalfWord資料讀入暫存器R0，並將R0的高16位清零。
    - LDRH R0，[R1，＃8] ；將記憶體位址為R1＋8的HalfWord資料讀入暫存器R0，並將R0的高16位清零。
    - LDRH R0，[R1，R2] ；將記憶體位址為R1＋R2的HalfWord資料讀入暫存器R0，並將R0的高16位清零。

# ARM指令集 - 載入/儲存指令

- STR指令
  - STR指令的格式為：STR{條件} 源暫存器，<記憶體位址>
  - STR指令用於從源暫存器中將一個32位元的資料傳送到記憶體中。該指令在程式設計中比較常用，且定址方式靈活多樣，使用方式可參考指令LDR。
  - Ex：
    - STR R0，[R1]，＃8 ；將R0中的資料寫入以R1為位址的記憶
      ；體中，並將新位址R1＋8寫入R1。
    - STR R0，[R1，＃8] ；將R0中的資料寫入以R1＋8為位址的記
      ；憶體中。



# ARM指令集 - 載入/儲存指令

- STRB指令
  - STRB指令的格式為：
    STR{條件}B 源暫存器，<記憶體位址>
  - STRB指令用於從源暫存器中將一個8位元的位元組資料傳送到記憶體中。該位元組資料為源暫存器中的低8位。
  - Ex：
    - STRB R0，[R1] ；將暫存器R0中的位元組資料寫入以
      ；R1為位址的記憶體中。
    - STRB R0，[R1，＃8] ；將暫存器R0中的位元組資料寫入
      ；以R1＋8為位址的記憶體中。

# ARM指令集 - 載入/儲存指令

- STRH指令
  - STRH指令的格式為：

    STR{條件}H 源暫存器，<記憶體位址>
  - STRH指令用於從源暫存器中將一個16位元的半字資料傳送到記憶體中。該半字資料為源暫存器中的低16位。
  - Ex：
    - STRH　R0，[R1]　　；將暫存器R0中的HalfWord資料寫入；以R1為位址的記憶體中。
    - STRH　R0，[R1，＃8]　；將暫存器R0中的HalfWord資料；寫入以R1＋8為位址的記憶體中。



# ARM指令集 - 多重資料載入/儲存指令

- ARM微處理器所支援多重資料載入/儲存指令可以一次在一個連續的記憶體單元和多個暫存器之間傳送資料，多重載入指令用於將一個連續的記憶體中的資料傳送到多個暫存器，多重資料儲存指令則完成相反的操作。常用的載入儲存指令如下：
  - LDM　　多重資料載入指令
  - STM　　多重資料儲存指令

# ARM指令集 - 多重資料載入/儲存指令

- LDM（或STM）指令
  - LDM（或STM）指令的格式為：LDM（或STM）{條件}{類型} 基址暫存器{!}，暫存器列表{^}
  - LDM（或STM）指令用於從由基址暫存器所指示的一個連續記憶體到暫存器列表所指示的多個暫存器之間傳送資料，該指令的常見用途是將多個暫存器的內容輸入或輸出。其中，{類型}為以下幾種情況：
    - IA　每次傳送後位址加1；　　FD　滿遞減堆疊；
    - IB　每次傳送前位址加1；　　ED　空遞減堆疊；
    - DA　每次傳送後位址減1；　　FA　滿遞增堆疊；
    - DB　每次傳送前位址減1；　　EA　空遞增堆疊；
  - {!}為可選尾碼，若選用該尾碼，則當資料傳送完畢之後，將最後的位址寫入基址暫存器，否則基址暫存器的內容不改變。(基址暫存器不允許為R15，暫存器列表可以為R0～R15的任意組合。)
  - {^}為可選尾碼，當指令為LDM且暫存器列表中包含R15，選用該尾碼時表示：除了正常的資料傳送之外，還將SPSR複製到CPSR。同時，該尾碼還表示傳入或傳出的是用戶模式下的暫存器，而不是當前模式下的暫存器。
  - Ex：
    - STMFD　R13!，{R0，R4-R12，LR}　　；將暫存器列表中的暫存器（R0，R4到R12，LR）存入堆疊。
    - LDMFD　R13!，{R0，R4-R12，PC}　　；將堆疊內容恢復到暫存器（R0，R4到R12，LR）。



# ARM指令集 - 資料交換指令

- ARM微處理器所支援資料交換指令能在記憶體和暫存器之間交換資料。資料交換指令有如下兩條：
  - SWP　　Word資料交換指令
  - SWPB　　Byte資料交換指令

# ARM指令集 - 資料交換指令

- SWP指令
  - SWP指令的格式為：
    SWP{條件} 目的暫存器，源暫存器1，[源暫存器2]
  - SWP指令用於將源暫存器2所指向的記憶體中的字資料傳送到目的暫存器中，同時將源暫存器1中的字資料傳送到源暫存器2所指向的記憶體中。顯然，當源暫存器1和目的暫存器為同一個暫存器時，指令交換該暫存器和記憶體的內容。
  - Ex：

```
SWP   R0，R1，[R2]    ；將R2所指向的記憶體中的Word資料傳送
                      ；到R0，同時將R1中的Word資料傳送到
                      ；R2所指向的儲存單元。
SWP   R0，R0，[R1]    ；該指令完成將R1所指向的記憶體中的
                      ；Word資料與R0中的字資料交換。
```



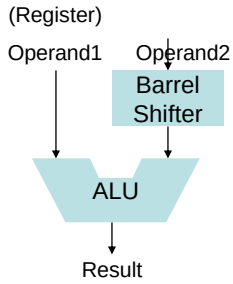
# ARM指令集 - 資料交換指令

- SWPB指令
  - SWPB指令的格式為：
    SWP{條件}B 目的暫存器，源暫存器1，[源暫存器2]
  - SWPB指令用於將源暫存器2所指向的記憶體中的位元組資料傳送到目的暫存器中，目的暫存器的高24清零，同時將源暫存器1中的位元組資料傳送到源暫存器2所指向的記憶體中。顯然，當源暫存器1和目的暫存器為同一個暫存器時，指令交換該暫存器和記憶體的內容。
  - Ex：

```
SWPB        R0，R1，[R2]    ；將R2所指向的記憶體中的位元
                            ；組資料傳送到R0，R0的高24位
                            ；清零，同時將R1中的低8位元資
                            ；料傳送到R2所指向的儲存單元
SWPB        R0，R0，[R1]    ；該指令完成將R1所指向的記憶
                            ；體中的位元組資料與R0中的低8
                            ；位元資料交換。
```

# ARM指令集 - 移位元指令

- ARM微處理器內嵌的桶型移位器（Barrel Shifter），支援資料的各種移位元操作，移位元操作在ARM指令集中不作為單獨的指令使用，它只能作為指令格式中是一個欄位，在組合語言中表示為指令中的選項。例如，資料處理指令的第二個運算元為暫存器時，就可以加入移位元操作選項對它進行各種移位操作。移位操作包括如下6種類型，ASL和LSL是等價的，可以自由互換：
  - LSL 邏輯左移
  - ASL 算術左移
  - LSR 邏輯右移
  - ASR 算術右移
  - ROR 迴圈右移
  - RRX 帶擴展的迴圈右移






# ARM指令集 - 移位元指令

- LSL（或ASL）操作
  - LSL（或ASL）操作的格式為：

    通用暫存器，LSL（或ASL）運算元
  - LSL（或ASL）可完成對通用暫存器中的內容進行邏輯（或算術）的左移操作，按運算元所指定的數量向左移位，低位用零來填充。其中，運算元可以是通用暫存器，也可以是立即常數（0～31）。
  - Ex：
    - MOV R0, R1, LSL#2 ；將R1中的內容左移兩位元後；傳送到R0中。

# ARM指令集 - 移位元指令

- LSR操作
  - LSR操作的格式為：通用暫存器，LSR 運算元
  - LSR可完成對通用暫存器中的內容進行右移的操作，按運算元所指定的數量向右移位，左端用零來填充。其中，運算元可以是通用暫存器，也可以是立即常數（0～31）。
  - Ex：
    - MOV R0, R1, LSR#2 ；將R1中的內容右移兩位元後；傳送到R0中，左端用零來填；充。



# ARM指令集 - 移位元指令

- ASR操作
  - ASR操作的格式為：通用暫存器，ASR 運算元
  - ASR可完成對通用暫存器中的內容進行右移的操作，按運算元所指定的數量向右移位，左端用第31位的值來填充。其中，運算元可以是通用暫存器，也可以是立即常數（0～31）。
  - Ex：
    - MOV R0, R1, ASR#2 ；將R1中的內容右移兩位元後；傳送到R0中，左端用第31位；的值來填充。

# ARM指令集 - 移位元指令

- RRX操作
  - RRX操作的格式為：通用暫存器，RRX 運算元
  - RRX可完成對通用暫存器中的內容進行帶擴展的迴圈右移的操作，按運算元所指定的數量向右迴圈移位，左端用進位元標誌位元C來填充。其中，運算元可以是通用暫存器，也可以是立即常數（0～31）。
  - Ex：
    - MOV R0, R1, RRX#2 ；將R1中的內容進行帶擴展的；迴圈右移兩位後傳送到R0中

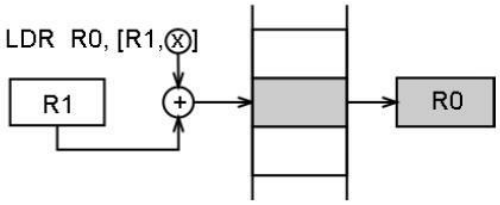


# ARM指令集 - 輔助運算器指令&異常產生指令

- ARM微處理器可支援多達16個輔助運算器，用於各種協處理操作，在程式執行的過程中，每個輔助運算器只執行針對自身的協處理指令，忽略ARM處理器和其他輔助運算器的指令。
- ARM的輔助運算器指令主要用於ARM處理器初始化ARM輔助運算器的資料處理操作，以及在ARM處理器的暫存器和輔助運算器的暫存器之間傳送資料，和在ARM輔助運算器的暫存器和記憶體之間傳送資料。ARM輔助運算器指令包括以下5條：
  - CDP 輔助運算器數操作指令
  - LDC 輔助運算器資料載入指令
  - STC 輔助運算器資料儲存指令
  - MCR ARM處理器暫存器到輔助運算器暫存器的資料傳送指令
  - MRC 輔助運算器暫存器到ARM處理器暫存器的資料傳送指令
- ARM微處理器所支援的異常指令有如下兩條：
  - SWI 軟體中斷指令
  - BKPT 中斷點中斷指令

# 5 Addressing Modes

## Offset Addressing

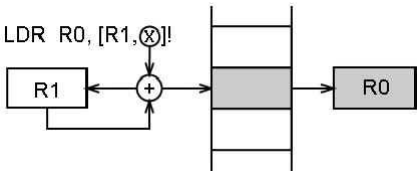



- LDR R0, [R1] ; R0 ← [R1], R1 not changed
- LDR R0, [R1,#4] ; R0 ← [R1+4], R1 not changed
- LDR R0, [R1, R2] ; R0 ← [R1+R2], R1,R2 not changed



# 5 Addressing Modes

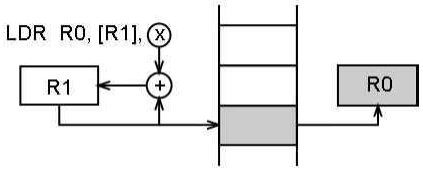
## Pre-index Addressing




- LDR R0, [R1, #4] ! ; R0 ← [R1+4], R1 ← R1+4
- LDR R0, [R1, R2] ! ; R0 ← [R1+R2], R2 not changed
  ; R1 ← R1+R2
- LDR R0, [R1, R2, LSL #2]! ; R0 ← [R1+(R2 shift left by 2 bits)],
  ; R1 ← R1+(R2 shift left by 2 bits),
  ; R2 not changed

# 5 Addressing Modes

## Post-index Addressing




- LDR R0, [R1], #4 ; R0 ← [R1], R1 ← R1+4
- LDR R0, [R1], R2 ; R0 ← [R1], R2 not changed
  ; R1 ← R1+R2
- LDR R0, [R1], R2, LSL #2 ; R0 ← [R1], R2 not changed,
  ; R1 ← R1+(R2 shift left by 2 bits)