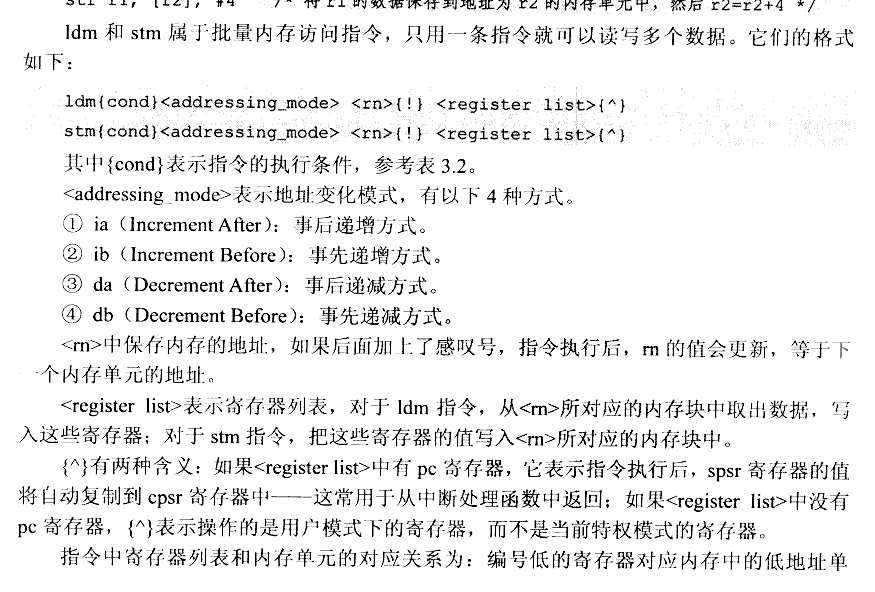
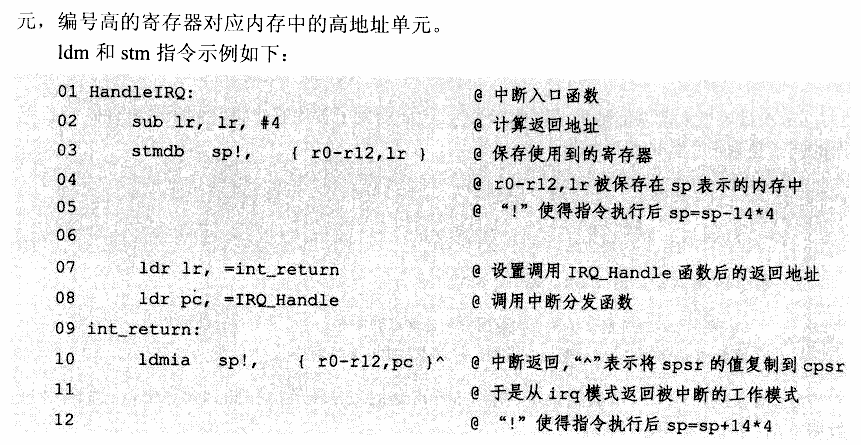
* ldr r0, [r1] 读内存指令，将r1内存地址中的数据读取到r0寄存器中
* str r0 , [r1] 写数据到指定的内存地址空间， 将寄存器r0 中的数据写入到r1 的内存地址当中
* b 跳转指令
* mov r0, r1 r1寄存器中的值复制到r0寄存器中

mov r0, 0x100 将一个立即数装入到r0 寄存器中

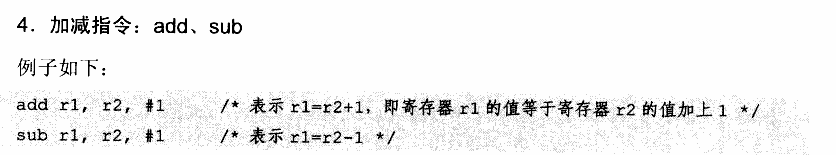
注意：如果mov这样用 mov r0, 0x12345678 用将是错误的！因为32位中肯定有些位数是表示指令本身的，剩下不足 32位数不能表示所有的数字，只能表示一些简单值。

* ldr r0, = 0x12345678 ldr 带等号的为伪指令，伪指令最终会被编译成多条指令； 这条指令的意思是将立即数0x12345678 装入到寄存器r0。
* bl main 跳转到函数main ，并且将返回值保存到lr寄存器
* ldm stm



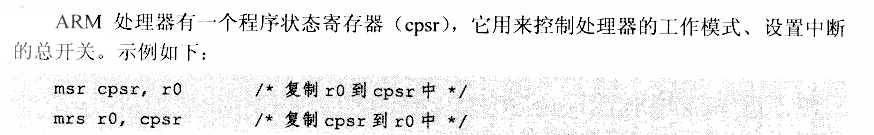


* add sub



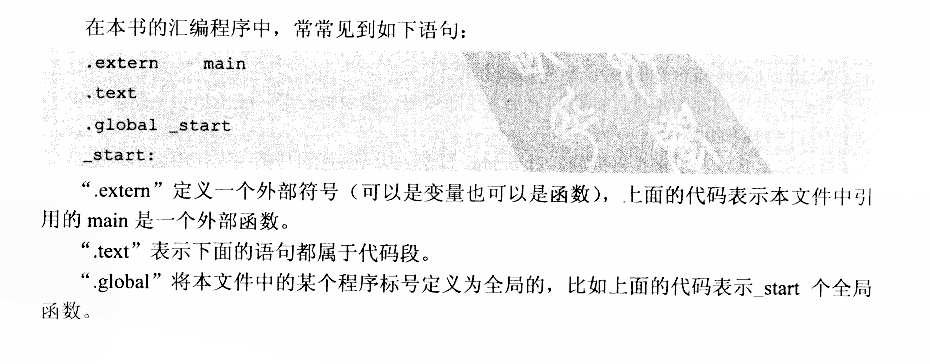
* msr mrs

程序状态寄存器的访问指令



* 一些伪指令

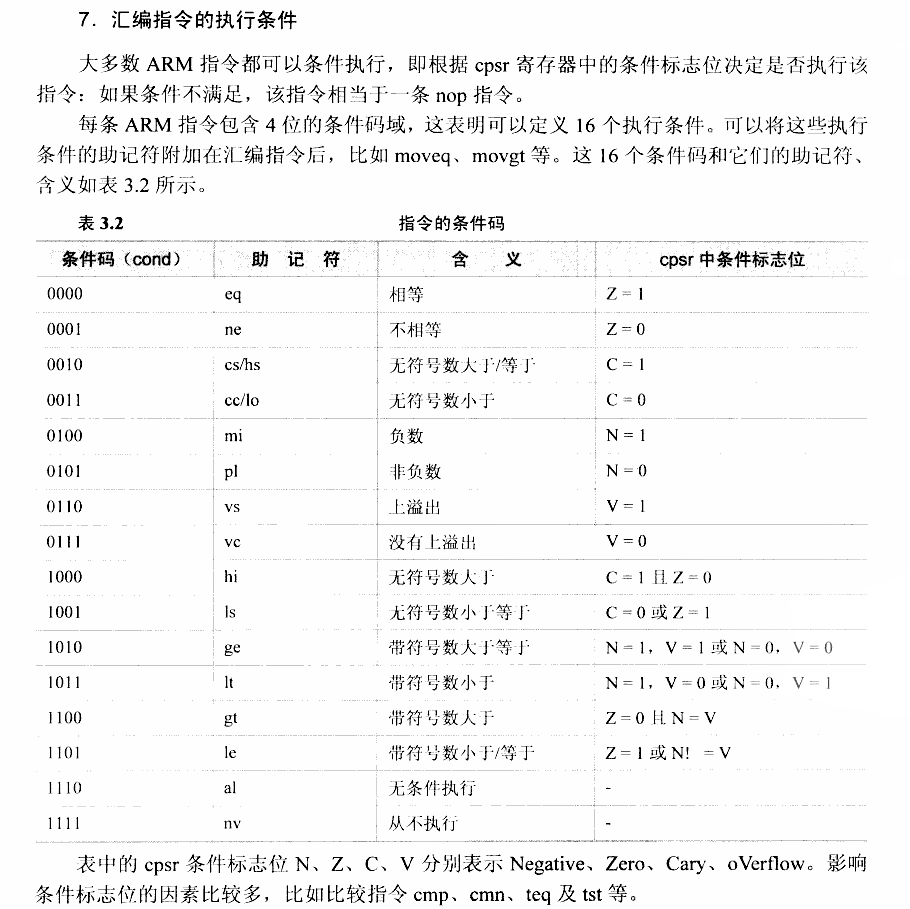
.extern .global .text



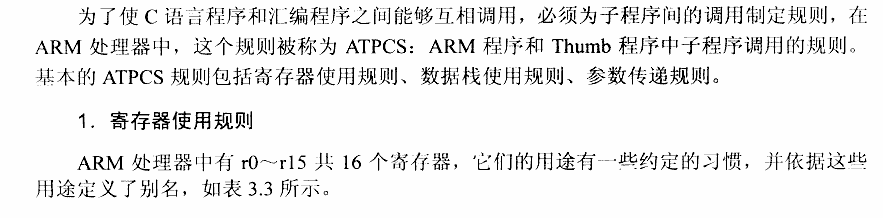
.extern : 可以理解为汇编文件的声明，相当于C语言头文件的 extern，可以声明变量和函数。

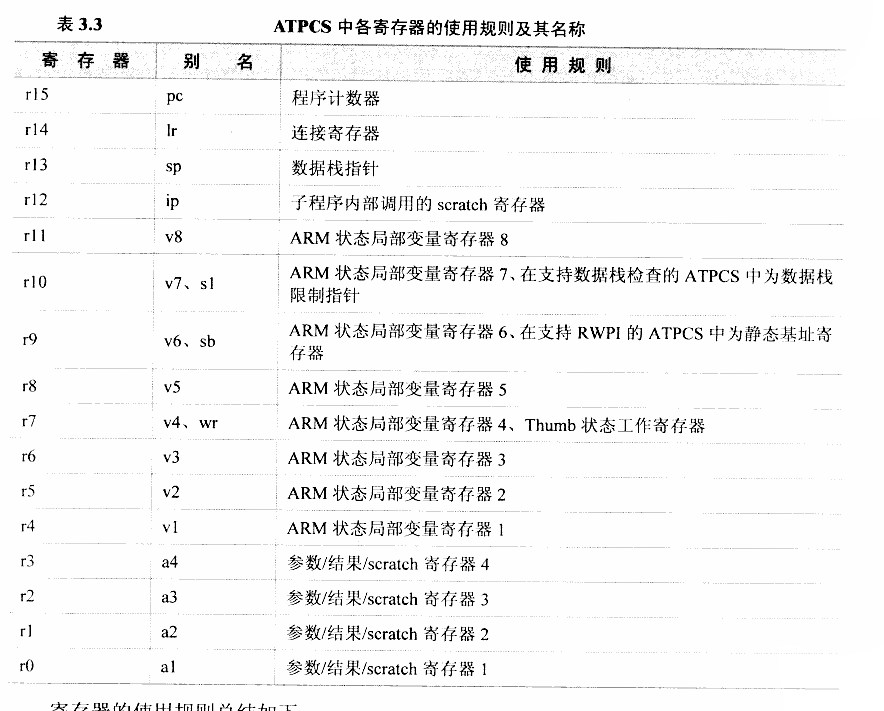
.global: 可以认为汇编文件定义的全局函数或变量，可以理解为C语言源文件定义的函数和变量没有加static 是全局可以见的。

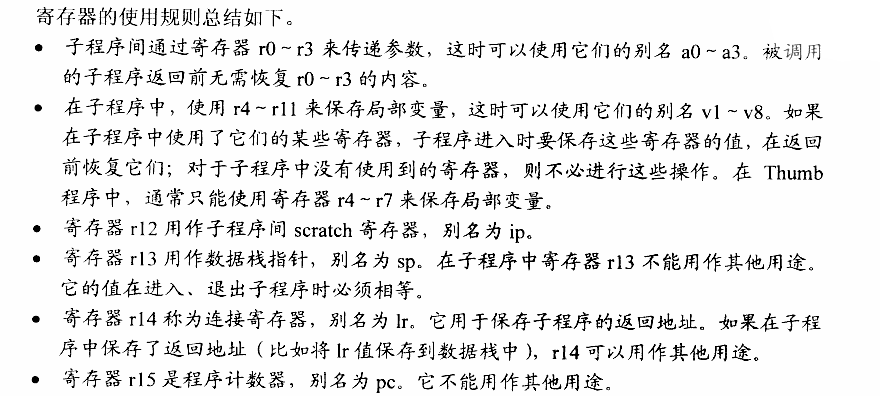
* 汇编指令的一些执行条件



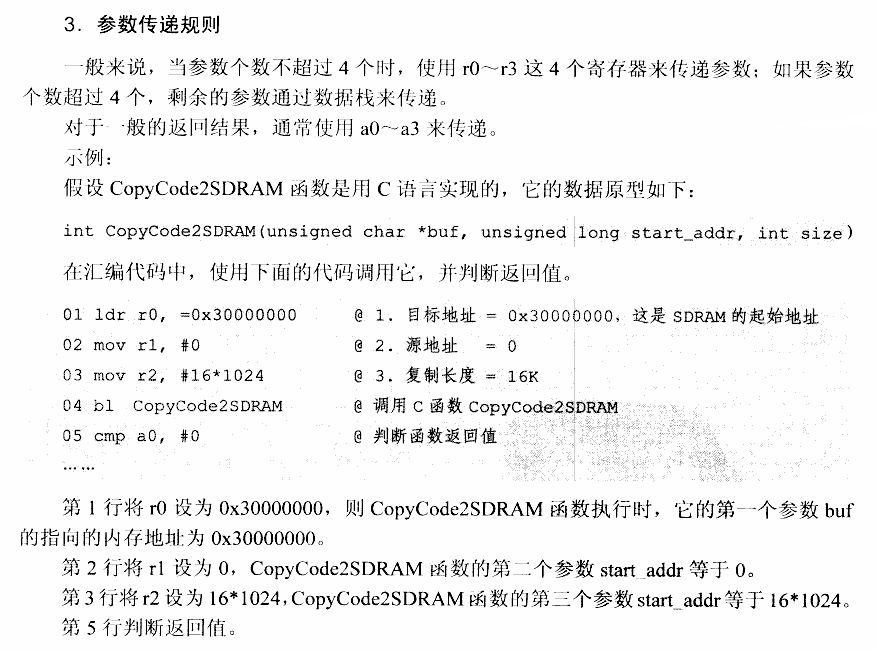
* ARM-THUMB 子程序调用规则ATPCS



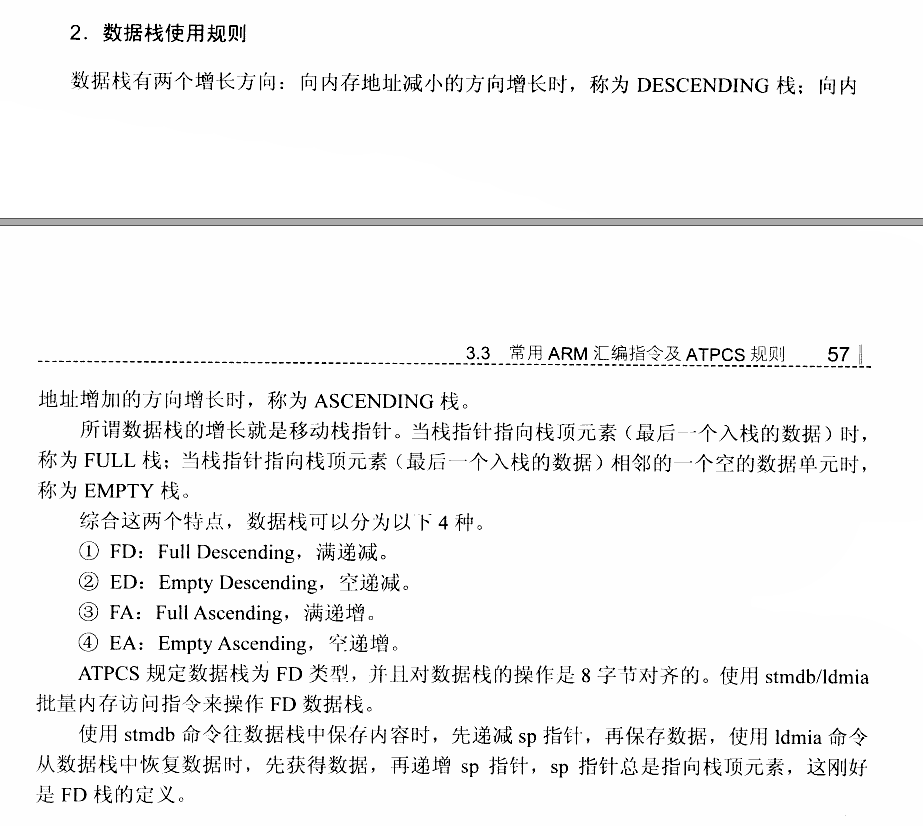




\*可以简单理解为R0-R3 是用来传递参数的，同时R0 还可以作为返回值，如果参数大于四个的时候就会使用栈来传递参数。



* ATPCS数据栈的使用规则



* 地址无关码和地址相关码

b/bl 是地址无关的，是相对跳转；ldr pc, \_start\_address 是绝对跳转，是地址相关的。