二、ARM的寻址方式

1.立即寻址

含义：操作数包含在指令的32位机器编码中;
例如:

```
ADD R0, R0, #5     ;R0 = R0 + 5
AND R1, R2, #0x01 ;R1 = R2 AND 0x01
```

注意：立即数所占位数是12位(不明白请看上一节笔记)所以0x2345是不能作为立即数放入ARM指令中。

问题：这个操作数2和寄存器里面的数据在ARM的CPU逻辑运算单元进行计算的时候这些数据都要转换成32位的数据,那么12位的立即数是如何在ARM硬件当中被转换成32的立即数?

解决：12位的编码中包括8位常数和4位循环右移值,由8位常数循环右移4位值的二倍得到最后的32位立即数。1

例如：`MOV R0,0x0000F200 ;R0 = 0x0000F200`

机器代码：E3A00CF2

由上一次笔记可知晓机器指令格式，E(1110)为条件码其后缀助符为AL，标志位为无条件，定义也为无条件。也就说没有条件可以限制MOV指令是否可以执行也可以将其指令写成MOVAL R0,0x0000F200。同理3A就是MOV指令的机器编码，下面就不——赘述不明白请移步到上一次的笔记中看相关解析。

4位循环右移值：C(十进制12)

8位常数：0xF2

方法：循环右移的位数是12✕2=24，然后就得到了32位的数值

移位前：0000 0000 0000 0000 0000 0000 1111 0010

移位后：0000 0000 0000 0000 1111 0010 0000 0000

注意：并不是所有的32位立即数都可以这样编码！在使用立即数之前需

要做合法性的判断。这是对编程实在是很不友好，值得庆幸的是编译系统提供了伪指令LDR。LDR R1,=0x87654321。即使R1 = 0x87654321

2.寄存器寻址

含义：操作数存放在寄存器中;
基本方式:

```
ADD R0, R1, R2 ;R0 = R1 + R2 因为操作数2占了12位，用12位去描述R0到R1
               ;其实用四位就可以描述R0到R15寄存器所以其他位数就可以进
```

第二操作数寄存器的移位操作:

```
ADD R3,R2,R1,LSR #2 ;R3 = R2 + R1/4
```

3.寄存器间接寻址

含义：利用寄存器的数值作为存储器指针，数据传送类的load/store类指令都使用寄存器间接寻址方式;
例如:

```
LDR R0,[R1] ;即将R1的数值作为内存单元的地址获的该地址的内容放进R0中,
            ;此处R1可以理解成C语言中的指针
```

4.基址加偏移地址

- **前变址**
  例如:

```
LDR R0,[R1,#4] ;此时R1作为基地址往高地址在加一个单元
               ;获得其中内容放进R0中
```

- **自动变址**

  例如:

```
LDR R0,[R1,#4]! ;此时R1作为基地址往高地址在加一个单元获得其中内容
                ;并且将基址寄存器做一次更新即R1 = R1 + 4
```

- **后变址**

  例如:

```
LDR R0,[R1,#4] ;即将R1的数值作为内存单元的地址获的该地址的内容放
               ;并且将基址寄存器做一次更新即R1 = R1 + 4
```

- **寄存器偏移地址**

  例如:

```
LDR R0,[R1,R2] ;此时R1为基地址R2为偏移地址，
               ;将R1+R2的值作为地址获得其地址的内容放入R0中。
LDR R0,[R1,R2,LSL,#2] ;其地址为R1+R2✕4，将其地址的内容放进R0中。
```

5.多寄存器及块拷贝寻址

含义: 一条指令完成多字数据或数据块的传送;
基本指令: LDM/STM
基址寄存器变化方式:
IA: 操作完后地址递增。
IB: 地址先增加后完成操作。
DA: 操作完后地址递减。
DB: 地址先递减后完成操作。
多寄存器语法表示:

多寄存器用"{}"包含,连续寄存器使用"-"间隔,否在用","分隔

例如:

```
LDMIA R0,{R1-R4,R6}    ;R1 = [R0],R2 = [R0 + 4],......,R6 = [R0 + 
                       ;此时R0的数值是不会更新的
LDMIA R0!,{R1-R4,R6}   ;此时R0的数值是会更新的
```

6.堆栈寻址

含义: 存储空间中的数据栈与寄存器组之间的批量数据传输,;

基本指令: LDM/STM;

FD/ED: 满递减/空递减 （满就是sp指针所指的位置是否由压入的数据,空则相反）

FA/EA: 满递增/空递增 （减就是栈的生长方式即往栈中压入数据其地址是由高地址向低地址生长）

例如:

```
STMFD SP!  {R0-R7,LR} ;入栈
LDMFD SP!  {R0-R7,LR} ;弹出堆栈
```

7.相对寻址

含义: 将程序计数器PC作为基址寄存器,指令中的地址标号字段作为偏移量进行寻址,跳转指令采用相对寻址方式。

结尾:

初学ARM汇编将其分段整理成笔记供自己参考也供与大家学习，如有错误请大佬们直言指出，如果感觉有用那就点个赞留个言，谢谢观众老爷们的赏脸。

若想获得上述内容的PDF版本移步到GitHub下载。

地址: https://github.com/QianquanChina/Study-Notes

-----缱绻