ARM 汇编指令集

1 跳转指令

1.1 跳转指令 B：

```
B    LABLE          ;跳转到标号 LABEL 处
B    0X1111         ;跳转到绝对地址 0X1111 处
```

1.2 带连接的跳转指令 BL:

```
START   …
        BL    NEXT      ;跳转到标号 NEXT 处，同时保存当前 PC 到 R14 中
        …               ;返回地址
        …
NEXT…                   ;子程序入口
        MOV   PC,R14    ;返回
```

1.3 带状态切换的跳转指令 BX:

```
MOV   R0,  #0X0201
BX          R0          ;程序跳转到 0x0200 处，微处理器切换到 Thumb
```

状态（地址必须是 4 的倍数，否则产生不可预知的后果）

2 算术运算指令

2.1 不带进位加法指令 ADD

ADD R0, R1, R2 ;R0← (R1)+(R2)

ADD R0, R1, #112 ;R0← (R1)+ 112

ADD R0, R1, R2, LSL #1 ;R0←(R1)+(R2<<1) ;将 R2 中的值左移 1 位,再与 R1 值相加，结果送 R0

2.2 带进位加法指令 ADC

ADDS R0, R3, R6 ;加最低位字节，不带进位

ADCS R1, R4, R7 ;加第二个字，带进位

ADCS R2, R5,R8 ;加第三个字，带进位

;三句话实现了 96bit 加法运算，由于 ARM 寄存器宽度只有 32bit 所以分三次相加

2.3 不带进位减法指令 SUB ;S—进位标志

SUB R0, R1, R2 ;R0←(R1)- (R2)

SUB R0, R1, #112 ;R0←(R1)- 112

SUB R0, R1 ,R2 LSL#1 ;R0←(R1)- (R2<<1)

2.4 带进位减法指令 SBC

SUBS R0, R3, R6 ;减最低位字节，不带进位

SBCS R1, R4, R7 ;减第二个字，带进位

SBCS R2, R5, R8 ;减第三个字，带进位

;三句话实现了 96bit 减法运算，由于 ARM 寄存器宽度只有 32bit 所以分三次相减

2.5 不带进位逆向减法指令 RSB

```
RSB     R0, R1, R2                    ;R0←(R2)- (R1)
RSB     R0, R1,   #112                ;R0←  112- (R1)
RSB     R0, R1, R2,   LSL#1           ;R0←(R2<<1)-R1
```

2.6 带进位逆向减法指令 RSC

```
RSBS    R0, R6, R3                    ;减最低字节的字，不带进位
RSCS    R1, R7, R4                    ;减第二个字，带进位
RSCS    R2, R8, R5                    ;减第三个字，带进位
```

;三句话实现了 96bit 减法运算，由于 ARM 寄存器宽度只有 32bit 所以分三次相减

2.7 32 位乘法指令 MUL

```
MUL     R0, R1, R2                    ;R0←(R1) X(R2)
MULS    R0, R1, R2                    ;R0←(R1) X(R2)    ;更新 CPSR 标志位
```

2.8 乘-累加指令 MLA

```
MLA     R0, R1, R2, R3                ;R0←(R1) X(R2)+(R3)
MLAS    R0, R1, R2, R3                ;R0←(R1) X(R2)+(R3)   ;更新 CPSR
```

标志位

2.9 无符号数长乘指令 UMULL

```
MOV     R5, #0X01
MOV     R8, #0X02
UMULL   R0, R1, R5, R8                ;(R1) (R0)←(R5)X(R8)
```

;UMULL 指令实现 64bit 无符号数乘法

2.10 无符号长乘-累加指令 UMLAL

```
MOV     R0, #0X01
MOV     R1, #0X02
MOV     R5, #0X01
MOV     R8, #0X02
UMLAL     R0, R1, R5, R8                ;R0←(R0) +(R5)X(R8)低字节
                                        ;R1←(R1) +(R5) X(R8)高字节
;UMLAL 指令为 64 位无符号乘-累加指令
```

2.11 有符号长乘指令 SMULL

```
MOV     R5, #0X01
MOV     R8, #0X02
SMULL     R0, R1, R5, R8                ;(R1) (R0)←(R5)X(R8)
;SMULL 指令实现 64bit 有符号数乘法
```

2.12 有符号长乘-累加指令 SMLAL

```
MOV     R0, #0X01
MOV     R1, #0X02
MOV     R5, #0X01
MOV     R8, #0X02
SMLAL     R0, R1, R5, R8                ;R0←(R0) +(R5)X(R8)低字节
                                        ;R1←(R1) +(R5) X(R8)高字节
; SMLAL 指令为 64 位有符号乘-累加指令
```

2.13 比较指令 CMP

```
CMP     R1, #0X10                       ;比较
```

BGT TAG ;R1> #0X10 转到 TAG 标号处

……

2.14 负数比较指令 CMN

CMN R0, #1 ;判断 R0 中的值是否为 1 的补码，是则置标志位 Z 为 1

3 逻辑运算指令

3.1 “与”指令 AND

MOV R0, 0XFF

AND R0, R0, #3 ;取出 R0 的最低 2bit

3.2 “或”指令 ORR

MOV R0, 0XFF

ORR R0, R0, #3

3.3 “异或”指令 EOR

MOV R0, 0XFF

EOR R0, R0, #3 ;R0←(R0)^(0X03)

3.4 位清除指令 BIC

MOV R0, 0XFF

BIC R0, R0, #B11 ;寄存器 R0 的低 2bit 被清零

3.5 测试比较指令 TST

TST R1, #b11 ;测试寄存器 R1 中的第 0 位和第 1 位，更新 CPSR 中

标志位，应用中会在 TST 指令后加一条跳转指令。

3.6 异或测试指令 TEQ

```
TEQ    R0, R1          ;R1 和 R0 中的值按位异或，更新 CPSR，实际应用中用 TEQ
```

指令测试两个寄存器中的数值是否相等。

4 存储器访问指令

4.1 字加载指令 LDR

```
LDR    R1, [R0, #0X08]          ;读取 R0+0X08 地址处的数据，保存到 R1
                                ;完成后 R0 中的数据保持不变
LDR    R1, [R0]
LDR    R1, [R0, R2]
LDR    R1, [R0, R2,  LSL#2]     ;读取 R0+ (R2<<2)地址处的数据，保存到
R1
LDR    R1, LABEL                ;LABEL 为程序标号 ,必须是当前指令的±4KB
LDR    R1, [R0], #0X04
LDR    R1, [R0], #0X04          ;读取 R0 地址处的数据，保存到 R1
                                ;指令执行后 R0 中值变为 R0+0X04
```

注意：使用 LDR 指令时字节地址是 4 的倍数

4.2 字存储指令 STR

STR R0, [R1, #4] ;将 R0 中的数据保存到内存地址（R0）+4 中

4.3 字节加在指令 LDRB (低 8 位)

LDRB R0, [R1, #4] ;将地址（R1）+4 处的 1 字节存储到寄存器 R0 的低

;位，并将 R0 的高 24 位清零

4.4 字节存储指令 STRB

STRB R1, [R0, #0X04] ;将 R1 的低 8 位存到地址 R0+0X04 处

;执行后 R0 中数据不变

4.5 半字加载指令 LDRH (低 16 位)

LDRH R0, [R1, #8] ;将 R1+8 地址处的 16 位数据送 R0 中

LDRH R0, [R1, R2] ;将 R1+R2 地址处的数据送寄存器 R0 中

4.6 半字存储指令 STRH

STRH R1, [R0, #0X04] ;将 R1 的低 16bit 数据存到地址 R0+0X04 处

4.7 用户模式数据加载存储指令

用户模式数据加载存储指令	功能描述
LDRT	功能和 LDR 指令相同，当微处理器在特权模式下使用此指令时，内存系统将该操作当做一般用户模式下的内存访问指令
STRT	同上
LDRBT	同上

STRBT	同上

4.8 有符号字节加在指令 LDRSB (带符号 &低 8 位)

```
LDRSB   R0, [R1, R2]          ;将地址 R1+R2 上的 8 位数值送 R0
                              ;R0 的高 24 位用符号位扩展
```

4.9 有符号半字加载指令 LDRSH (带符号 &低 16 位)

```
LDRSH   R0, [R1]              ;将地址 R1 上的低 16 位数据送 R0
                              ;R0 的高 16 位用符号位扩展
```

4.10 批量数据加载/存储指令 LDM/STM

LDM/STM 指令地址模式选择

指令	操作	数据传送起始地址
IA	先传送数据，后基址加 4	(Rn)
IB	先基址加 4，后传送数据	(Rn) +4
DA	先传送数据，后基址减 4	(Rn)
DB	先基址减 4，后传送数据	(Rn)-4

基址	地址	数据
	0X0000002C	0X0000002C
	0X00000028	0X00000028
	0X00000024	0X00000024
	0X00000020	0X00000020
R0 0X0000001C →	0X0000001C	0X0000001C
	0X00000018	0X00000018
	0X00000014	0X00000014
	0X00000010	0X00000010

```
LDMIA   R0!, {R2-R4}    ;R0=0X00000028  R2=0X0000001C
                        ;R3=0X0000020   R4=0X00000024
LDMIB   R0!, {R2-R4}    ;R0=0X00000028  R2=0X00000020
                        ;R3=0X00000024  R4=0X00000028
LDMDA   R0!, {R2-R4}    ;R0=0X00000010  R2=0X0000001C
                        ;R3=0X00000018  R4=0X00000014
LDMDB   R0!, {R2-R4}    ;R0=0X00000010  R2=0X00000018
                        ;R3=0X00000014  R4=0X00000010
```

4.11 寄存器 存储器字交换指令 SWP

```
MOV    R0, #0X10
MOV    R1, #0X11
MOV    R2, #0X12
SWP    R0, R1, [R2]
```

设地址 0X12 处值为 0XFF,执行指令后 R0=0XFF, R1=0X11, R2=0X12

内存地址 0X12 处值为 0X11

4.12 寄存器 存储器字节交换指令 SWPB (高 24 位填 0 低 8 位数据)

```
MOV    R0, #0X10
MOV    R1, #0XFF11
MOV    R2, #0X12
SWPB   R0, R1, [R2]
;设地址 0X12 处值为 0XFF,执行指令后 R0=0XFF,  R1=0XFF11,  R2=0X12
;地址 0X12 处值为 0XFF11
```

5 数据传送指令

5.1 数据传送指令 MOV

```
MOV    R0, #1
MOV    R0, R1                ;R0← (R1)
MOV    R0, R1, LSL#3         ;R0← (R1<<3)
```

5.2 反向传送指令 MVN

MVN R0, #8 ;R0=-9??

;指令执行后，R0 中值为立即数 8 按位取反的值？？

5.3 程序状态字内容送通用寄存器指令 MRS

MRS R1, CPSR ;将 CPSR 的值送 R1

;MRS 指令是唯一可以直接读取 CPSR 和 SPSR 寄存器的指令

5.4 写状态寄存器指令 MSR

;状态寄存器分 4 个域 [31:24]为条件标志域用 f 表示，[23:16]为状态位域用 s 表示，

[15:8]为扩展位域用 x 表示，[7:0]为控制域用 c 表示。

MSR CPSR_c, #0X11 ;CPSR[7:0]=0X11

MSR CPSR_cxsf， R0 ;CPSR=R0

6 移位指令

6.1 逻辑左移操作 LSL (低位填 0)

MOV R1, #0X01

MOV R0, R1, LSL#2

MOV R2, #2

MOV R0, R1, LSL R2

6.2 算数左移操作 ASL (低位填 0)

MOV R1, #0X01

MOV R0, R1, ASL #2

6.3 逻辑右移操作 LSR (高位填 0)

MOV R1, #0X04

MOV R0, R1, LSR#2

6.4 算术右移操作 ASR （？？？）

MOV R1, #0X80000004

MOV R0, R1, ASR# 2

6.5 循环右移操作 R0R (高位由低位填充)

MOV R1, #0X01

MOV R0, R1, ROR#2

6.6 带扩展的循环右移操作 RRX （低位→ C → 高位）

MOV R3, #0X7FFFFFFF

MOV R4, #0X7FFFFFFF

MOV R1, #0X04

ADDS R3, R3, R4

MOV R0, R1, RRX#2

7.0 异常产生指令

7.1 软中断指令 SWI

SWI 12 ;将产生中断号为 12 的软中断

;通过该条指令可以用软件的方法实现异常

7.2 断点中断指令 BKPT

BKPT [immediate_16] ;16 位立即数

;断点中断指令 BKPT 常被用来设置软件断点，在调试时非常有用

8 协处理指令

8.1 协处理器数据操作指令 CDP

CPD P1, 1, C2, C3, C4

;指令要操作的协处理器为 p1 ，1 为协处理器 p1 的指令操作代码。c2、c3、c4 为协处理器 p1 的寄存器。CPD 指令的使用并不影响 ARM 微处理器中 CPSR 寄存器的状态位。

8.2 协处理器数据读取指令 LDC

LDC p5, c2, [R0, #4] ;将内存地址为 R0+4 处的数据送到协处理器 p5 的寄存器 c2 中。

LDC p5, c1, [R0,R1] ;将内存地址为 R0+R1 处的数据送到协处理器 p5 的寄存器 c1 中。

8.3 协处理器数据写入指令 STC

STC P1, P2, c2, [R0, #-0X4] ;将协处理器 p1 的寄存器 c2 中的数据读取到内存地址 R0-0X4 处。

8.4 ARM 微处理器到协处理器的数据传送指令 MCR

MCR{[code]} [coproc], [opcode_1], [Rd],[CRn], [CRm],{[opcode_2]}

8.5 协处理器到 ARM 寄存器的数据传送指令 MRC

MRC{[code]} [coproc], [opcode_1], [Rd],[CRn], [CRm],{[opcode_2]}