ARM GNU汇编学习笔记

# Linux下一个ARM GNU汇编Demo工程

新建三个文件demo.S demo.lds makefile,文件内容如下：

demo.S:

1. .text
2. .global \_start
3. \_start:
5. mov     r0,#1
7. loop:
9. b       loop

demo.lds:

1. OUTPUT\_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-littlearm")
2. OUTPUT\_ARCH(arm)
3. ENTRY(\_start)
4. SECTIONS
5. {
6. . = 0x00000000;
7. .text :   { \*(.text) }
8. .data :   { \*(.data) }
9. .bss  :   { \*(.bss)  }
10. }

Makefile:

1. demo.bin : demo.S
2. arm-linux-gcc -g -c -o demo.o demo.S
3. arm-linux-ld  -g -Tdemo.lds demo.o  -o demo\_elf
4. arm-linux-objcopy -O binary -S demo\_elf demo.bin
5. clean:
6. rm -f   demo.bin demo\_elf \*.o

# 查看反汇编代码arm-linux-objdump

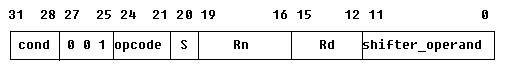
1. arm-linux-objdump -D demo\_elf
2. arm-linux-objdump -D -b binary -m arm demo.bin

# ARM汇编语法格式及编码格式

一条典型的ARM汇编语法格式：

**<opcode>{cond}{S} <Rd>,<Rn>,<shifter\_operand>**

一条典型的ARM汇编编码格式：



|  |  |
| --- | --- |
| 项目 | 说明 |
| opcode | 指令助记符，如ADD、SUB、MOV等 |
| cond | 条件码助记符，如EQ(0000)、NE(0001)、AL(1110)等 |
| S | 如果指令有S后缀，则该指令的操作会影响CPSR的值 |
| Rd | 目标寄存器 |
| Rn | 包含第一个源操作数的寄存器 |
| shifter\_operand | 表示第二个源操作数，可以为立即数 |

# ARM汇编的立即数

操作数的语法格式：

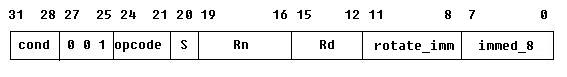
#<immediate>

其中， **<immediate>=immed\_8循环右移(2\*rotate\_imm)**

immediate  :  立即数

immed\_8    :  8位的常数

rotate\_imm :  4位的循环右移值



意思是每个立即数都是由一个8位的常循环右移偶数位得到。

ROR 循环右移：即将操作数循环按指定的数量向右循环移位，左边用右边移出的位来填充；

例如：有效的立即数：0x104、0xff0

immediate    :    0x104            //  0001 0000 0100

immed\_8    :    0x41                //  0100 0001

左边补0   :    0x00000041 //  0000 0000 0000 0000 0000 0000 0100 0001

rotate\_imm  :    15                   //  循环右移(2\*15)位

                                                     //  0000 0000 0000 0000 0000 0001 0000 0100

                                                     //  0x0000 0104

immediate    :    0xff0               //  1111 1111 0000

immed\_8    :    0xff                  //  1111 1111

 左边补0       :    0x000000ff    //  0000 0000 0000 0000 0000 0000 1111 1111

rotate\_imm  :    14                    //  循环右移(2\*14)位

                                                     //  0000 0000 0000 0000 0000 1111 1111 0000

                                                    //  0x0000 0ff0

无效的立即数不能通过上面的构造方法得到：0x101、0x102、0xff1

0x101 : 0001 0000 0001 (不能通过一个八位常数获得)

0x102 : 0001 0000 0010 (不能通过移动偶数位获得)

0xff1 : 1111 1111 0001 (不能通过一个八位常数获得)

但是有时候用出现这种情况，如：0x3f0   // 0011 1111 0000

可以为：#0x3f,ror (2\*14)                          // 0000 0011 1111

或：    #0xfc,ror (2\*15)                             // 0000 1111 1100

面对这种情况，ARM有如下规则：

1. 当立即数的值0-0xFF时，immed\_8=<immediate>,rotate\_imm=0;

2. 其它情况下，汇编编译器选择使 rotate\_imm 的数值最小的编码方式.

所以0x3f0 是通过 0x3f>>(2\*14) 的方式获得。

LDR伪指令: 装入32位立即数或地址到寄存器

语法： LDR{<cond>} <register>,=[<expr>|label-expr]

expr表示32位常数

eg:

    ldr r3,=0xFFF

    ldr r1, =12345678