汇编

\_add\_a\_and\_b:

push %ebx 4 3-

mov %eax, [%esp+8] 从第8位地址找出来，是2

mov %ebx, [%esp+12 ] 从第12位地址找出来，是3，栈区压栈

add %eax, %ebx

pop %ebx

ret

\_main:

push 3 4 1-

push 2 4 2-

call \_add\_a\_and\_b

add %esp,8

ret

一、cpu指令

1、第一行、push 指令

push 3

\_main开始执行，并且从栈上面为main建立一帧，且指向的地址写入esp寄存器。push 3 将运算子3放入栈，将3写入main帧

push存在一个前置操作，先取出esp里面地址，将其减去4字节，将新地址写入esp寄存器，使用减法是因为栈从高到底发展，4个字节是因为3类型是int，占用4字节，得到新地址，3就会写入这个地址开始的4个字节。

2、第二行、

push 2

第2行一样，将2写入main帧，位置紧贴着1行的3，esp再减去4个字节。（esp-8）

3、第三行、call指令

call \_add\_a\_and\_b

表示 调用\_add\_a\_and\_b函数，程序会去找标签，并且建立帧

4、\_add\_a\_and\_b的第二行

push %ebx

表示将ebx写入\_add\_a\_and\_b帧，push再将esp-4地址。

5、\_add\_a\_and\_b的第三行

mov指令

mov指令用于将一个值写入某个寄存器

**mov %eax, [%esp+8]**

表示esp寄存器地址加上8个字节，得到新地址，在这个地址堆取出数据，之前换算到压栈第八个地址是目前是2，所以将2取出来写入eax

6、\_add\_a\_and\_b的第四行

mov %ebx， [%esp+12]

将esp寄存器第12位地址取出来，是3，放到ebx里面

7、\_add\_a\_and\_b的第五行

add指令，将两个算子相加，并且将结果写入第一个算子。

add %eax, %ebx

将eax寄存器值加上ebx值写入eax的寄存器（第一个算子）

8、\_add\_a\_and\_b的第六行

pop指令

pop用于取栈区最近写入的一个值（最低位地址的值），将这个值写入运算子指定位置

pop %ebx

取出stack最近写入值，ebx最原始的值，在将值写回ebx。（加法做完，可以还回去）

注：pop会将esp里面地址加4，回收4个字节。

9、\_add\_a\_and\_b的第七行

ret指令

ret用于终止当前函数执行，将运行权限交还给上层函数，（函数帧回收）

10、main的第四行

add %esp，8

将esp寄存器地址，手动加上8字节，写回esp里面。（因为esp是栈写入开始地址，前面pop操作回收4个字节，这里回收8字节，等于全部回收）。

11、main的第五行

ret

退出程序执行

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sub指令

sub ax，bx将ax中值减去bx，把结果放进去算子ax

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*arm汇编\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CPSR以及SPSR中各个位的含义以及相关技巧

N：负数标志位，如果目标寄存器中有符号数为负，则N=1，否则N=0

Z：零位标志。如果目标寄存器中数为0，则N=1，否则N=0；

C：进位标志。

1、无符号加法运算和cmn指令，如果产生进位，则C=1，否则C=0；

2、无符号减法运算和cmp指令，如果产生借位，则C=0，否则C=1；

3、进行移位操作的时候，C中保存最后一位移出

说明：当一条指令中同时含有算术运算指令时，影响C值云萨un而不是位移操作。

V：溢出标志位。进位有符号运算时如果发生错误，则V=1，否则 V=0；

注意：一些指令cmn、teq会无条件刷新cpsr中的条件标志位，其他指令必须要在指令后面加s才能改变cpsr的条件标志位

I：IRQ中断禁止位。I=1代表禁止中断，I=0代表允许

F：FIQ中断禁止位。F=1代表FIQ中断，F=0代表允许

T：ARM v4t指令集版以上才有效，armv4下不支持Thumb指令集。在支持Thumb指令集的处理器中，T=0代表ARM状态，T=1代表Thumb状态。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

数据指令：

MOV、ADD、ADDS、ADC、SUB、SUBS、SBC、RSB、MUL、AND、ORR、EOR、BIC、CMP、TST、TEQ、LSL、LSR、ASR、RORV

语法：

操作 cond s rd ，rn ，operand2

操作码 目标寄存器rd 第一操作寄存器rn 第二操作数oper

1、数据发送指令mov

mov r1, #0x1(立即数）

mov r2, r1 将 r2=r1

mov r1， 0x0000000ff 不是立即数，编译器阶段进行替换

数据指令

需要有#号才能进行进行赋值立即数

2、加法指令 add

add r2,1 r2=r2+1;

如果没有进位cpsr‘c’位置0

3、数据操作堆cpsr的影响

默认情况下，数据处理指令不影响条件码标志位，但可以添加“s”来影响标志位。

Adds r3,r1,#imm imm范围0-7且必须位lo寄存器

4、带进位的加法指令 ADC

adc r5,r1,r3 r5=r1+r3+’C’ cpsr的进位标志

5、减法指令SUB

sub r2,r0,r1 r2=r0-r1;

6、带借位的减法指令SBC

sbc R5，R1，R3

7、逆向减法指令rsb

rsb r1，r0，#5 r1=5-r0

8、乘法指令MUL

mul不可以使用立即数

9、乘法——累加指令MLA

MLA r3，r0，r1，r2 r3=(r0\*r1)+r2

10、逻辑与指令AND

and r2,r0,r1 r2=r0&r1

11、逻辑或指令ORR

orr r2,r0,#0xd3 r2=r0|0xd3

12、逻辑异或运算指令EOR

eor r2,r1,r0 r2=r1^r0

13、位清零指令BIC

bic{cond}{S} Rd,Rn,operand2

BIC指令将Rn的值与操作数operand2的反码按位逻辑“与”，结果存放到目的寄存器Rd中，

BIC R0，R0#0X0F；将第四位清零

14、比较指令CMP

cmp r1，r2 是r1-r2，但不影响两个数的值，但是会影响flag的值CF、ZF、OF、AF、PF

（1）、ZF=1，两个数相等，zero=1结果位0

（2）、CF=1，说明存在进位或者借位，cmp时进行的减法操作，可以看出是借位，所以r1<r2

CF=0，说明无借位，但要看此时ZF是否为0，如果不为0，则说明结果不为0，故此时r1>r2。

如果SF=0，OF=0说明此时值为正数，且没有一处，r1>r2

如果SF=1，0F=0说明此时值为负数，且没有溢出，r1<r2

如果SF=0，OF=1说明此时值为正数，有溢出，r2<r1

如果SF=1，OF=1说明此时值为负数，有溢出，r1>r2

最后作出判断是因为溢出本质，

两数为正，加，值为负数，说明溢出

两数为负，加，值为正，说明溢出

**正正得负，负负得正都为溢出，两数相减，同号不溢出，两数异号，结果与第一算子相同，则溢出。**

15、位测试指令TST

tst r0, #0x3 测试第三位是否位0，通过CPSR的’Z’判断

16、相等测试指令TEQ

teq r0,r1 测试是否相等，相等时异或结果为0，通过CPSR ‘z’

17、移位指令LSL、LSR、ASR、ROR

需要配合mov使用

mov r0, #0xff

mov r1,r0, lsl #4,进行将r0逻辑左移4位放入r1中

LSL 逻辑左移，高位移出，低位补零 (1<=n<=31)

LSR 逻辑右移，低位移出，高位补零 (1<=n<=32)

ASR 算是右移，低位移出，高位补符号位 （1<=n<=32)

ROR 循环右移，低位移出，高位补低位移出位 (1<=n<=31)

RRX 循环右移一位，带扩展。

18、LDR加载指定地址的字数据到某寄存器中

Rn不允许是15，

LDR Rd，addr

LDR B Rd,addr

LDR R0，[R1]

19、STR指令存储 寄存器 最低字节数据到指定地址单元中

20、ADDS 带进位add

adds rd,rn,rm

rd,rn,rm必须时lo寄存器

（1）、Lo寄存器：

概要：mips拥有32个通用寄存器，一个pc寄存器，一个HI寄存器，一个LO寄存器，另外协处理器也有自己的寄存器也有自己的寄存器，如CP0也有32个单独寄存器，（不同厂家可实现自己的独特寄存器MTC0 rt，rd，sel， sel进行区分），浮点协处理单元也有自己独立的寄存器。

（2）、32个通用寄存器：

0 zero returns 0

1 at 保留被asm使用

2-3 v0,v1 通过子程序返回变量

4-7 a0-a3 参数，第一次小量参数给子程序

8-15 t0-t7 临时，子程序使用保存

24-25 t8，t9

16-23 s0-s7 子程序寄存器变量，一个子程序连接一个旧变量和在退出之前恢复他，这样调用例程可以看到保存值。

26-27 k0,k1 保留被中断和异常处理使用，也许改变你的场管。

28 gp 全局指针，一些程序会给予维护，方便访问。

29 sp 栈指针

30 s8/fp 第九个寄存器变量，子程序需要一个可以使用框架子指针

31 ra 返回子程序地址

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

（3）、CP0寄存器

0 index 索引进入TLB 数组，4k核心，这个寄存器保留在4kp和4km核

1 random 随机通用索引，在tlb数组，这个寄存器保留在4km 4kp核心

2 entrylo0 低位指针关于tlb进入偶数虚拟页（4KC核心）。这个寄存器保留在4KP，4KM核心

3 entrylo1 低位指针关于tlb进入奇数虚拟页（4KC核心）。这个寄存器保留在4KP，4KM核心

4 context 指向内存页表条目的指针

5 pagemask 控制变量页大小在tlb表项，这个寄存器保留在4KP，4KM内核里面

6 wired 控制大量修复连线tlb页表，这个寄存器保留在4KP，4KM。

7 reserved reserved

8 badVaddr 报告地址最新地址关系异常

9 count 处理循环计数

10 entryhi 高位指针关于tlb页表，这个寄存器是保留在4KP 4KM里面。

11 compare 定时器中断控制

12 status 处理状态和控制

13 cause 最后一次异常

14 epc 最后的程序计数器异常

15 prid 程序处理识别和修订

16 config/ 配置寄存器

17 Lladdr 加载链接地址

18 watchlo 低地址观察点

19 watchhi 高地址观察点

20-22 reserved 保留

23 debug 调试控制和异常状态

24 depc 程序计数器在最后一次异常

25 reserved 保留

26 errctl 控制访问数据和sram数组给缓存区

27 reserved 保留

28 taglo/datalo 缓存区标签接口的低地址部分

29 reserved 保留

30 errorepc 最后一次程序计数器

31 desave 调试处理器暂存寄存器

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

mov 移动数据

mvn 移动和否定

add 加法

sub 减法

mul 乘法

lsl 逻辑左移

lsr 逻辑右移

asr 算术右移

ror 向右旋转

cmp 比较

and 按位和

orr 按位或

eor 按位异或

ldr 加载 ldr r2,[r0] 源地址是r0中找到的值,加载内存内容到寄存器里面

str 保存 str r2,[r1] 目标地址是一个从r1找到的值，保存一些东西到寄存器地址里面

ldm 加载多个

stm 保存多个

push 推入栈

pop 弹出栈

b 分支

bl 分支链接

bx 分支和交换

blx 分支链接和交换

swi/svc 系统呼叫

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*data struct\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BYTE 8位无符号整数，B代表字节

SBYTE 8位有符号整数，S代表符号

WORD 16位无符号整数

SWORD 16位有符号整数

DWORD 32位无符号整数，D代表双字节

SDWORD 32位有符号整数，SD代表有符号双字节

FWORD 48位整数（保护模式中远指针）

QWORD 64位整数，Q代表四字节

TBYTE 80位（10字节）整数，T代表10字节

REAL4 32位（4字节）IEEE短实数

REAL8 64位（8字节）IEEE长实数

REAL10 80位（10字节）IEEE扩展实数