看下题目：

Daddy told me I should study arm.
But I prefer to study my leg!

Download : http://pwnable.kr/bin/leg.c
Download : http://pwnable.kr/bin/leg.asm

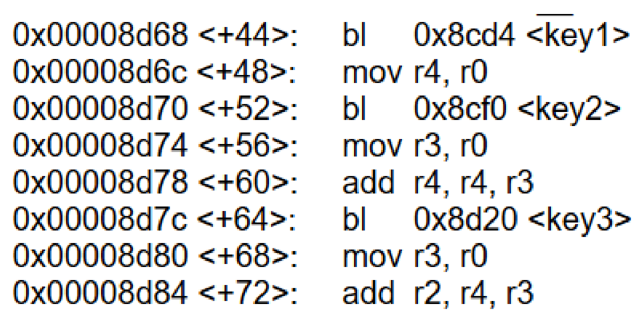
ssh leg@pwnable.kr -p2222 (pw:guest)

先连接上去看看源码：

```c
#include <stdio.h>
#include <fcntl.h>
int key1(){
        asm("mov r3, pc\n");
}
int key2(){
        asm(
        "push   {r6}\n"
        "add    r6, pc, $1\n"
        "bx     r6\n"
        ".code   16\n"
        "mov    r3, pc\n"
        "add    r3, $0x4\n"
        "push   {r3}\n"
        "pop    {pc}\n"
        ".code  32\n"
        "pop    {r6}\n"
        );
}
int key3(){
        asm("mov r3, lr\n");
}
int main(){
        int key=0;
        printf("Daddy has very strong arm! : ");
        scanf("%d", &key);
        if( (key1()+key2()+key3()) == key ){
                printf("Congratz!\n");
                int fd = open("flag", O_RDONLY);
                char buf[100];
                int r = read(fd, buf, 100);
                write(0, buf, r);
        }
        else{
                printf("I have strong leg :P\n");
        }
        return 0;
}
```

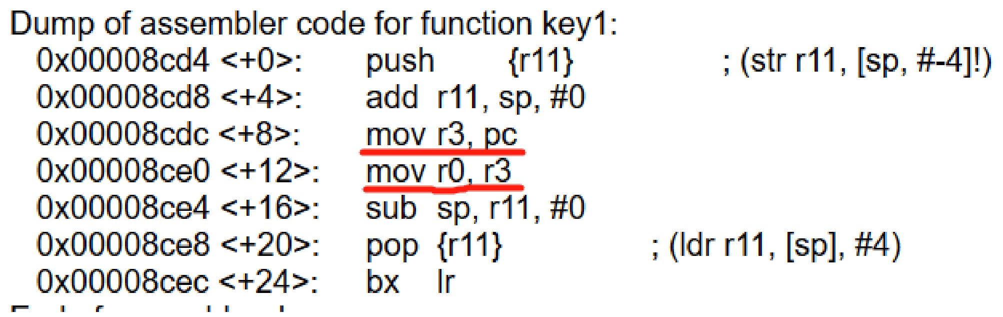
显然通过 key1+key2+key3 即可，再看看汇编源码

main 函数中，key 函数通过 r0 返回



```
0x00008d68 <+44>:   bl     0x8cd4 <key1>
0x00008d6c <+48>:   mov r4, r0
0x00008d70 <+52>:   bl     0x8cf0 <key2>
0x00008d74 <+56>:   mov r3, r0
0x00008d78 <+60>:   add  r4, r4, r3
0x00008d7c <+64>:   bl     0x8d20 <key3>
0x00008d80 <+68>:   mov r3, r0
0x00008d84 <+72>:   add  r2, r4, r3
```

key1：



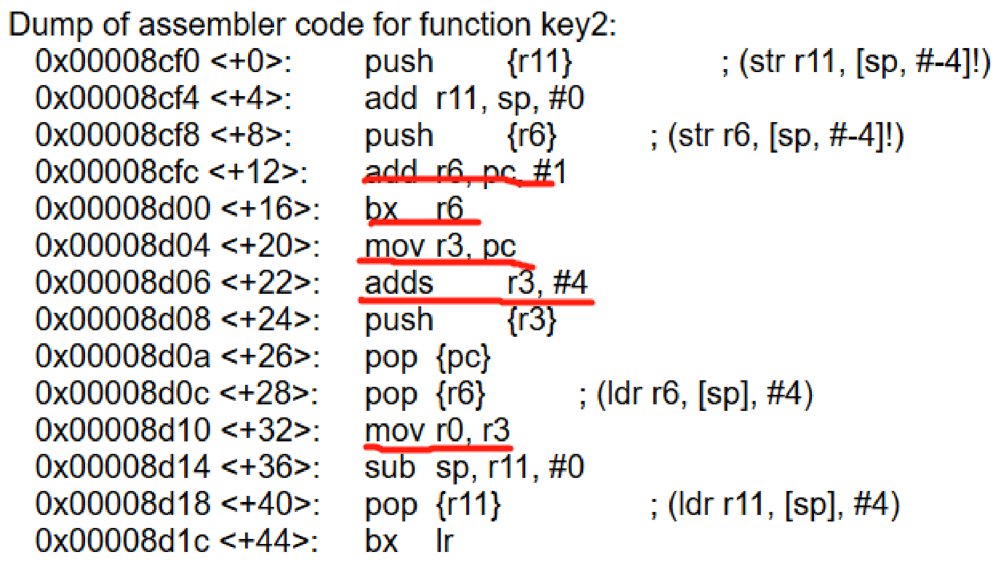
```
Dump of assembler code for function key1:
   0x00008cd4 <+0>:       push        {r11}              ; (str r11, [sp, #-4]!)
   0x00008cd8 <+4>:       add  r11, sp, #0
   0x00008cdc <+8>:       mov r3, pc
   0x00008ce0 <+12>:      mov r0, r3
   0x00008ce4 <+16>:      sub  sp, r11, #0
   0x00008ce8 <+20>:      pop  {r11}             ; (ldr r11, [sp], #4)
   0x00008cec <+24>:      bx    lr
```

key 是将 PC 寄存器的值赋给 r3，r3 再赋给 r0 返回出去

key1=PC

而 asm 指令在 asm 体系下是流水线　　即 PC=8cdc+8

key2：



```
Dump of assembler code for function key2:
   0x00008cf0 <+0>:       push        {r11}              ; (str r11, [sp, #-4]!)
   0x00008cf4 <+4>:       add  r11, sp, #0
   0x00008cf8 <+8>:       push        {r6}          ; (str r6, [sp, #-4]!)
   0x00008cfc <+12>:      add  r6, pc, #1
   0x00008d00 <+16>:      bx    r6
   0x00008d04 <+20>:      mov r3, pc
   0x00008d06 <+22>:      adds         r3, #4
   0x00008d08 <+24>:      push         {r3}
   0x00008d0a <+26>:      pop  {pc}
   0x00008d0c <+28>:      pop  {r6}          ; (ldr r6, [sp], #4)
   0x00008d10 <+32>:      mov r0, r3
   0x00008d14 <+36>:      sub  sp, r11, #0
   0x00008d18 <+40>:      pop  {r11}             ; (ldr r11, [sp], #4)
   0x00008d1c <+44>:      bx    lr
```

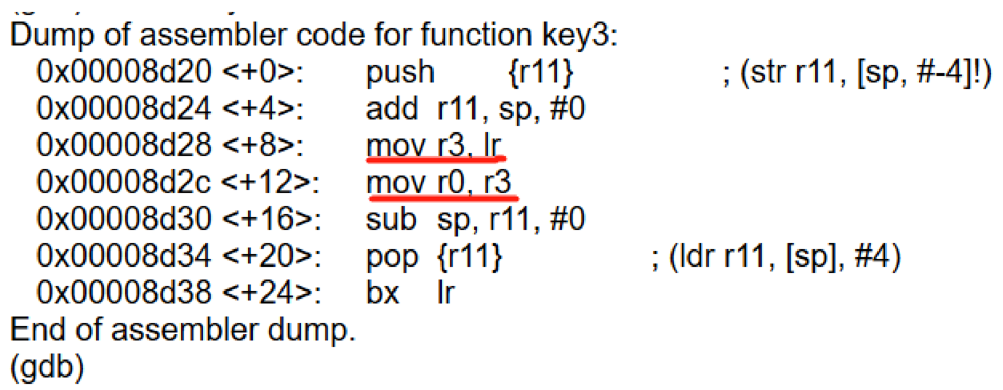
从 add r6，pc，#1 这句可知 r6=8cfc+8+1

Bx r6 跳转到 r6 的地址中，而跳转时检查地址最低为是否为 1，是则切换为 thumb 模式

即跳转到 8d04：mov r3，pc 即 r3=8d04+4（thumb 模式）

故 key2=r0=8d04+4+4

key3：



```
Dump of assembler code for function key3:
   0x00008d20 <+0>:     push    {r11}           ; (str r11, [sp, #-4]!)
   0x00008d24 <+4>:     add  r11, sp, #0
   0x00008d28 <+8>:     mov r3, lr
   0x00008d2c <+12>:    mov r0, r3
   0x00008d30 <+16>:    sub  sp, r11, #0
   0x00008d34 <+20>:    pop  {r11}              ; (ldr r11, [sp], #4)
   0x00008d38 <+24>:    bx   lr
End of assembler dump.
(gdb)
```

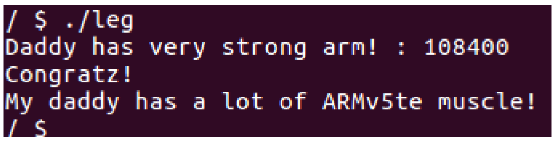
key3=lr

lr 是寄存器存储的是子函数的返回地址

lr=8d80

key=key1+key2+key3=108400

验证一下，得到 flag：



```
/ $ ./leg
Daddy has very strong arm! : 108400
Congratz!
My daddy has a lot of ARMv5te muscle!
/ $
```