作业4

练习

3.43

A.

08 04 86 43	返回地址
bf ff fc 94	<----%ebp
00 00 00 02	%edi的值
00 00 00 03	%esi的值
00 00 00 01	%ebx的值
	buf[4]-buf[7]
	buf[0]-buf[3]
	<--%esp

B.

08 04 86 00	返回地址
33 32 31 30	<-----%ebp
39 38 37 36	保存的%edi的值
35 34 33 32	保存的%esi的值
31 30 39 38	保存的%ebx的值
37 36 35 34	buf[4]-buf[7]
33 32 31 30	buf[0]-buf[3]
	<-----%esp

C. 程序应该试图返回到0x08048600

D. 寄存器%ebp,%edi,%esi,%ebx保存的值被破坏了

E. 因为字符串结尾是空字符'\0',而strlen()函数统计出的长度不包括空字符，所以调用malloc函数时参数应该是strlen(buf)+1。并且还应检验buf是否为空

3.44

A. 0xffffd754-0xffffb754=2*16^3=2*2^12=2^13

B. 128=2^7,所以需要尝试2^6=64次

3.45

A.

	不带保护者	带保护者
buf	-20(%ebp)	-20(%ebp)
v	-8(%ebp)	-24(%ebp)
金丝雀值	无	-8(%ebp)

B. 在有保护的代码中，一旦buf溢出，不会破坏v的值，但是马上会破坏金丝雀的值，以便于发现溢出行为。

3.47

src_t	dest_t	指令	S	D
long	long	movq	%rdi	%rax
int	long	movslq	%edi	%rax
char	long	movsbq	%dil	%rax
unsigned int	unsigned long	movl	%edi	%eax
unsigned char	unsigned long	movzbq	%dil	%rax
long	int	movl	%edi	%eax

src_t	dest_t	指令	S	D
unsigned long	unsigned	movl	%edi	%eax

3.48

函数原型如下：

```
long arithprob(int a,char b,long c,int d)
```

3.49

A. 填写C代码中缺失的部分

```
long fun_c(unsigned long x){
    long val=0;
    int i;
    for(i=0;i<8;i++){
        val+=x&0x0101010101010101;
        x>>=1;
    }
    val+=(val>>32);
    val+=(val>>16);
    val+=(val>>8);
    return val&0xFF;
}
```

B. 这段代码同时计算8个字节的和来对x中的位求和。然后对val的低32位求和，再是低16位，最后低8位，最终的结果在最低的一个字节中。

作业

3.69

A. 给出一个函数的C版本：

```
long trace(tree_ptr tp){
    long val=0;
    while(tp){
        val=tp->val;
        tp=tp->left;
    }
    return val;
}
```

B. 这段代码跟踪二叉树最左端的分支，并且返回最后一个结点的val值，如果没有就返回0.

3.70

A. 生成函数的C版本：

```
long traverse(tree_ptr tp){
    if(!tp){
        return LONG_MAX;
    }else{
```

```
        long val=tp->val;
        long left_val,right_val;
        left_val=traverse(tp->left);
        if(left_val<val){
            val=left_val;
        }
        right_val=traverse(tp->right);
        if(right_val<val){
            val=right_val;
        }
        return val;
    }
}
```

B. 这段代码用递归调用的方法返回二叉树中的最小值。如果没有则返回LONG_MAX.