作业3

练习

3.32

从第3行可以看出，%eax中存放的是p,%edx中存放的是d，即p在栈中的位置是16(%ebp),d在栈中的位置是12(%ebp)。又从第6行看出%edx中是x,%eax中是c,即x在栈中的位置是20(%ebp),c在栈中的位置是8(%ebp)，所以入栈的顺序是c,d,p,x,所以函数fun的原型是：

```
int fun(short c,char d,int *p,int x);
```

3.33

A. 第3行%ebp的值被设置成0x80003C(pushl把栈指针减4)

B. 第4行%esp的值被设置成0x800014

C. 局部变量x的存放地址是0x800038,局部变量y的存放地址是0x800034

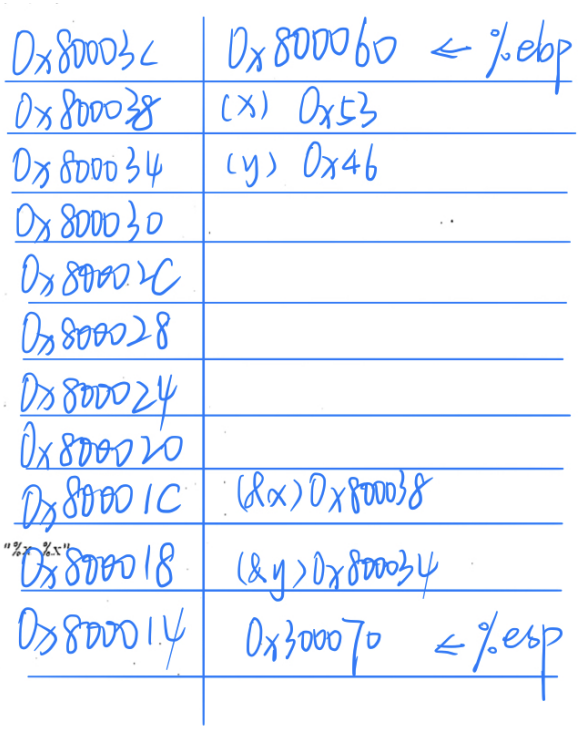
D.图如下：

练习题 3.33 给定 C 函数如下：

```
1   int proc(void)
2   {
3       int x,y;
4       scanf("%x %x", &y, &x);
5       return x-y;
6   }
```

GCC 产生以下汇编代码：

```
1   proc:
2     pushl   %ebp
3     movl    %esp, %ebp
4     subl    $40, %esp
5     leal    -4(%ebp), %eax
6     movl    %eax, 8(%esp)
7     leal    -8(%ebp), %eax
8     movl    %eax, 4(%esp)
9     movl    $.LC0, (%esp)     Pointer to string "%x %x"
10    call    scanf
      Diagram stack frame at this point
11    movl    -4(%ebp), %eax
12    subl    -8(%ebp), %eax
13    leave
14    ret
```



E. 未使用的栈帧区域为0x800020~0x800033

3.34

A. rfun存储在被调用者保存寄存器%ebx中的值是x的值

B. 填写代码中缺失的表达式：

```
int rfun(unsigned x){
    if(x==0)
        return 0;
    unsigned nx=x>>1;
    int rv=rfun(nx);
    return (x&0x1)+rv;
}
```

C. 这段代码的功能是用递归的方法计算x中所有位的和

3.36

表达式	类型	值	汇编代码
s+1	short*	xs+2	leal 2(%edx),%eax
s[3]	short	M[xs+6]	movw 6(%edx),%ax
&s[i]	short*	xs+2*i	leal (%edx,%ecx,2),%eax
s[4*i+1]	short	M[xs+8*i+2]	movw 2(%edx,%ecx,8),%ax
s+i-5	short*	xs+2*i-10	leal -10(%edx,%ecx,2),%eax

3.37

由汇编代码可以推出，计算mat1[i][j]所用的式子是mat1[0][0]+4*(7i+j),说明mat1有7列，则N=7，同理可推出M=5

3.39

A.字段偏移量：

字段	p	s.x	s.y	next
偏移量	0	4	8	12

B.这个结构总共需要16字节

C. 补全代码中缺失的表达式：

```
void sp_init(struct prob *sp){
    sp->s.x=sp->s.y;
    sp->p=&(sp->s.x);
    sp->next=sp;
}
```

3.40

EXPR	TYPE	Code
up->t1.s	int	movl 4(%eax),%eax movl %eax,(%edx)
up->t1.v	short	movw (%eax),%ax movw %ax,(%edx)
&up->t1.d	short*	leal 2(%eax), %eax movl %eax, (%edx)
up->t2.a	int *	movl %eax, (%edx)

up->t2.a[up->t1.s]	int	movl 4(%eax), %ecx
		movl (%eax,%ecx,4), %eax
		movl %eax, (%edx)

*up->t2.p	char	movl 8(%eax), %eax
		movb (%eax), %al
		movb %al, (%edx)

3.41

A.

i	c	j	d	总大小	对齐要求
0	4	8	12	16	4

B.

i	c	d	j	总大小	对齐要求
0	4	5	8	12	4

C.

w	c	总大小	对齐要求
0	6	10	2

D.

w	c	总大小	对齐要求
0	8	20	4

E.

a	p	总大小	对齐要求
0	32	36	4

作业

3.60

A. 数组元素A[i][j][k]的位置是xA+4*(k+T*(i*S+j))

B. 第3行计算9j, 4-6行计算63i, 第7行计算(63i+9j)=9(7i+j),第8行计算9(7i+j)+k,由第9行知道地址为xA+4(9(7i+j)+k),所以S=7，T=9，因为sizeofA=2772,4RST=2772,所以R=11

3.62

A. 从第7行可以看出，M=76/4=19

B. 根据第8行的cmpl, %edi保存i, %ecx保存j

C. C代码版本：

```
void transpose_improve(int A[M][M]){
    int i,j;
    for(i=0;i<M;i++){
        int *hori=A[i][0];
        int *colu=A[0][i];
        for(j=0;j<i;j++){
            int x=hori[j];
            int y=*colu;
            *colu=x;
            hori[j]=y;
            colu+=M;
        }
    }
}
```

3.63

从第18行可以看出E1(n)放在寄存器%esi中，从1-4行可以看出E1(n)=3*n

由第17行和第8行可以推出，E2(n)=2*n-1

所以最终结果是

```
#define E1(n) (3*n)
#define E2(n) (2*n-1)
```

3.64

A. 8(%ebp)是函数返回的结构的首地址，16(%ebp)是s1.v,12(%ebp)是s1.p

B. 从%esp往上，第一个字段是返回的结构s2的首地址，第二个字段是s1.p,第三个字段是s1.v,第四个字段是s2.prod,第五个字段是s2.sum

C. 向函数传递结构参数的通用策略是通过栈来传递，通过地址来访问结构中的元素

D. 首先函数在栈上为作为结果的返回结构分配栈空间，然后再创建一个指向返回结构的指针来访问它。

3.66

A.可以推出a_struct的大小是28（第7行），又根据第11行bp->right是0xc8(%ecx)，CNT=(200-4)/28=7

B.从第13行结合之前的计算可以得出数组x的偏移量是4。

```
typedef struct{
    int idx;
    int x[6];
}a_struct;
```

3.67

A.

e1.p的偏移量是0，e1.x的偏移量是4

e2.y的偏移量是0，e2.next的偏移量是4

B. 这个结构总共需要8个字节

C. 填写代码中缺失的表达式：

```
void proc(union ele *up){
    up->e2.next->e1.x=*(up->e2.next->e1.p)-up->e2.y;
}
```