```
long rfun(unsigned long x) {
    if (x == 0)
        return 0;
    unsigned long nx = x>>2;
    long rv = rfun(nx);
    return x + rv;
}
```

3.36　这个练习测试你对数据大小和数组索引的理解。注意，任何类型的指针都是 8 个字节长。short 数据类型需要 2 个字节，而 int 需要 4 个。

数组	元素大小	总大小	起始地址	元素 i
S	2	14	x_S	x_S+2i
T	8	24	x_T	x_T+8i
U	8	48	x_U	x_U+8i
V	4	32	x_V	x_V+4i
W	8	32	x_W	x_W+8i

3.37　这个练习是关于整数数组 E 的练习的一个变形。理解指针与指针指向的对象之间的区别是很重要的。因为数据类型 short 需要 2 个字节，所以所有的数组索引都将乘以因子 2。前面我们用的是 movl，现在用的则是 movw。

表达式	类型	值	汇编语句
S+1	short*	x_S+2	leal2(%rdx),%rax
S[3]	short	M[x_S+6]	movw6(%rdx),%ax
&S[i]	short*	x_S+2i	leal(%rdx,%rcx,2),%rax
S[4*i+1]	short	M[x_S+8i+2]	movw2(%rdx,%rcx,8),%ax
S+i-5	short*	$x_S+2i-10$	leal-10(%rdx,%rcx,2),%rax

3.38　这个练习要求你完成缩放操作，来确定地址的计算，并且应用行优先索引的公式(3.1)。第一步是注释汇编代码，来确定如何计算地址引用：

```
    long sum_element(long i, long j)
    i in %rdi, j in %rsi
1   sum_element:
2     leaq    0(,%rdi,8), %rdx        Compute 8i
3     subq    %rdi, %rdx              Compute 7i
4     addq    %rsi, %rdx              Compute 7i + j
5     leaq    (%rsi,%rsi,4), %rax     Compute 5j
6     addq    %rax, %rdi              Compute i + 5j
7     movq    Q(,%rdi,8), %rax        Retrieve M[xQ + 8 (5j + i)]
8     addq    P(,%rdx,8), %rax        Add M[xP + 8 (7i + j)]
9     ret
```

我们可以看出，对矩阵 P 的引用是在字节偏移 $8\times(7i+j)$ 的地方，而对矩阵 Q 的引用是在字节偏移 $8\times(5j+i)$ 的地方。由此我们可以确定 P 有 7 列，而 Q 有 5 列，得到 $M=5$ 和 $N=7$。

3.39　这些计算是公式(3.1)的直接应用：

- 对于 $L=4$，$C=16$ 和 $j=0$，指针 Aptr 等于 $x_A+4\times(16i+0)=x_A+64i$。
- 对于 $L=4$，$C=16$，$i=0$ 和 $j=k$，指针 Bptr 等于 $x_B+4\times(16\times0+k)=x_B+4k$。
- 对于 $L=4$，$C=16$，$i=16$ 和 $j=k$，Bend 等于 $x_B+4\times(16\times16+k)=x_B+1024+4k$。

3.40　这个练习要求你能够研究编译产生的汇编代码，了解执行了哪些优化。在这个情况中，编译器做一些聪明的优化。

让我们先来研究一下 C 代码，然后看看如何从为原始函数产生的汇编代码推导出这个 C 代码。