

因为字段 i 的偏移量为 0，所以这个字段的地址就是 r 的值。为了存储到字段 j ，代码要将 r 的地址加上偏移量 4。

要产生一个指向结构内部对象的指针，我们只需将结构的地址加上该字段的偏移量。例如，只用加上偏移量 $8+4\times 1=12$ ，就可以得到指针 $\&(r\rightarrow a[1])$ 。对于在寄存器 $\%rdi$ 中的指针 r 和在寄存器 $\%rsi$ 中的长整数变量 i ，我们可以用一条指令产生指针 $\&(r\rightarrow a[i])$ 的值：

```
Registers: r in %rdi, i %rsi
1    leaq    8(%rdi,%rsi,4), %rax    Set %rax to &r->a[i]
```


最后举一个例子，下面的代码实现的是语句：

```
r->p = &r->a[r->i + r->j];
```

开始时 r 在寄存器 $\%rdi$ 中：

```
Registers: r in %rdi
1    movl    4(%rdi), %eax           Get r->j
2    addl    (%rdi), %eax           Add r->i
3    cltq                               Extend to 8 bytes
4    leaq    8(%rdi,%rax,4), %rax    Compute &r->a[r->i + r->j]
5    movq    %rax, 16(%rdi)         Store in r->p
```

综上所述，结构的各个字段的选取完全是在编译时处理的。机器代码不包含关于字段声明或字段名字的信息。

 **练习题 3.41** 考虑下面的结构声明：

```
struct prob {
    int *p;
    struct {
        int x;
        int y;
    } s;
    struct prob *next;
};
```

这个声明说明一个结构可以嵌套在另一个结构中，就像数组可以嵌套在结构中、数组可以嵌套在数组中一样。

下面的过程(省略了某些表达式)对这个结构进行操作：

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

A. 下列字段的偏移量是多少(以字节为单位)?

```
p:      _____
s.x:    _____
s.y:    _____
next:   _____
```

B. 这个结构总共需要多少字节?

C. 编译器为 `sp_init` 的主体产生的汇编代码如下：