讲，因为它需要理解数组和对齐。

图 3-43a 的代码给出了一个包含变长数组的例子。该函数声明了 n 个指针的局部数组 p，这里 n 由第一个参数给出。这要求在栈上分配 $8n$ 个字节，这里 n 的值每次调用该函数时都会不同。因此编译器无法确定要给该函数的栈帧分配多少空间。此外，该程序还产生一个对局部变量 i 的地址引用，因此该变量必须存储在栈中。在执行工程中，程序必须能够访问局部变量 i 和数组 p 中的元素。返回时，该函数必须释放这个栈帧，并将栈指针设置为存储返回地址的位置。

```
long vframe(long n, long idx, long *q)  {
    long i;
    long *p[n];
    p[0] = &i;
    for (i = 1; i < n; i++)
        p[i] = q;
    return *p[idx];
}
```

a）C代码

```
    long vframe(long n, long idx, long *q)
    n in %rdi, idx in %rsi, q in %rdx
    Only portions of code shown
1   vframe:
2     pushq   %rbp                    Save old %rbp
3     movq    %rsp, %rbp              Set frame pointer
4     subq    $16, %rsp               Allocate space for i (%rsp = s1)
5     leaq    22(,%rdi,8), %rax
6     andq    $-16, %rax
7     subq    %rax, %rsp              Allocate space for array p (%rsp = s2)
8     leaq    7(%rsp), %rax
9     shrq    $3, %rax
10    leaq    0(,%rax,8), %r8         Set %r8 to &p[0]
11    movq    %r8, %rcx               Set %rcx to &p[0] (%rcx = p)
          . . .
    Code for initialization loop
    i in %rax and on stack, n in %rdi, p in %rcx, q in %rdx
12  .L3:                            loop:
13    movq    %rdx, (%rcx,%rax,8)     Set p[i] to q
14    addq    $1, %rax                Increment i
15    movq    %rax, -8(%rbp)          Store on stack
16  .L2:
17    movq    -8(%rbp), %rax          Retrieve i from stack
18    cmpq    %rdi, %rax              Compare i:n
19    jl      .L3                     If <, goto loop
          . . .
    Code for function exit
20    leave                           Restore %rbp and %rsp
21    ret                             Return
```

b）生成的部分汇编代码

图 3-43　需要使用帧指针的函数。变长数组意味着在编译时无法确定栈帧的大小