```
3       long *p;
4   } strA;
5
6   typedef struct {
7       long u[2];
8       long q;
9   } strB;
10
11  strB process(strA s) {
12      strB r;
13      r.u[0] = s.a[1];
14      r.u[1] = s.a[0];
15      r.q =    *s.p;
16      return r;
17  }
18
19  long eval(long x, long y, long z) {
20      strA s;
21      s.a[0] = x;
22      s.a[1] = y;
23      s.p = &z;
24      strB r = process(s);
25      return r.u[0] + r.u[1] + r.q;
26  }
```

GCC 为这两个函数产生下面的代码：

```
    strB process(strA s)
1   process:
2     movq    %rdi, %rax
3     movq    24(%rsp), %rdx
4     movq    (%rdx), %rdx
5     movq    16(%rsp), %rcx
6     movq    %rcx, (%rdi)
7     movq    8(%rsp), %rcx
8     movq    %rcx, 8(%rdi)
9     movq    %rdx, 16(%rdi)
10    ret

    long eval(long x, long y, long z)
    x in %rdi, y in %rsi, z in %rdx
1   eval:
2     subq    $104, %rsp
3     movq    %rdx, 24(%rsp)
4     leaq    24(%rsp), %rax
5     movq    %rdi, (%rsp)
6     movq    %rsi, 8(%rsp)
7     movq    %rax, 16(%rsp)
8     leaq    64(%rsp), %rdi
9     call    process
10    movq    72(%rsp), %rax
11    addq    64(%rsp), %rax
12    addq    80(%rsp), %rax
13    addq    $104, %rsp
14    ret
```

A. 从 eval 函数的第 2 行我们可以看到，它在栈上分配了 104 个字节。画出 eval 的栈帧，给出它在调用 process 前存储在栈上的值。

B. eval 调用 process 时传递了什么值？

C. process 的代码是如何访问结构参数 s 的元素的？

D. process 的代码是如何设置结果结构 r 的字段的？