3.59:

把参数 x 扩展到 128 位时，可以写成 $x = 2^{64} \cdot x_h + x_l$，同样的 y 可以写成 $y = 2^{64} \cdot y_h + y_l$，那么 $xy = 2^{128} \cdot x_h \cdot y_h + 2^{64} \cdot (x_h \cdot y_l + x_l \cdot y_h) + x_l \cdot y_l$，如果取 128 位的乘积结果，$2^{128} \cdot x_h \cdot y_h$ 在 128 位内全是 0，可以删去。$x_l \cdot y_l$ 是两个 64 位数，相乘可能超过 64 位，因此 $x_l \cdot y_l$ 可以写为 $x_l \cdot y_l = 2^{64} \cdot z_h + z_l$，则 xy 的 128 位乘积可以写成 $2^{64} \cdot (x_h \cdot y_l + x_l \cdot y_h + z_h) + z_l$。

下面分析汇编代码:

```
void store_prod(int128_t *dest, int64_t x, int64_t y){
    *dest = x * (int128_t) y;
}
//%rsi存的是x_low,%rdx存的是y_low
//下面是汇编代码部分:
store_prod:
    movq %rdx, %rax    //%rax = %rdx = y_low
    cqto               //将%rax中的符号位复制到%rdx的所有位中
                       //即%rdx = y_high
    movq %rsi, %rcx    //%rcx = %rsi = x_low
    sarq $63, %rcx     //将%rcx中的值算术右移63位,即%rcx = x_high
    imulq %rax, %rcx   //%rcx = %rax * %rcx = y_low*x_high
    imulq %rsi, %rdx   //%rdx = %rsi * %rdx = x_low*y_high
    addq %rdx, %rcx    //%rcx = %rdx + %rcx = x_low*y_high+x_high*y_low
    mulq %rsi          //将%rax和%rsi相乘,即x_low*y_low
                       //高64位z_high存入%rdx,低64位z_low存入%rax
                       //即%rdx = z_high, %rax = z_low
    addq %rcx, %rdx    //%rdx = %rdx + %rcx
```

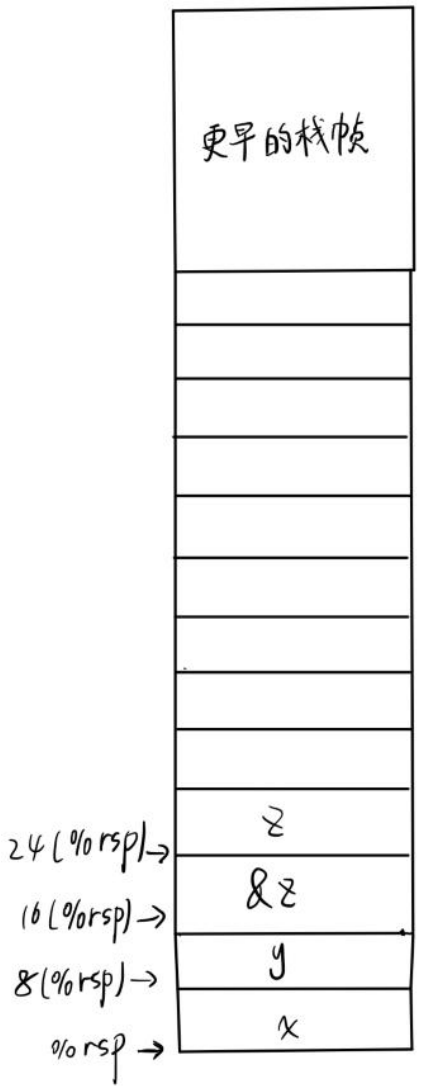
3.63:

```
long switch_prob(long x, long n){
        long result = x;
        switch(n){
                case 60:
                        result = 8*x;
                        break;
                case 61:
                        result = x + 75;
                        break;
                case 62:
                        result = 8*x;
                        break;
                case 63:
                        result = x>>3;
                        break;
```

```
        case 64:
                result = (x<<4-x)*(x<<4-x)+75;
                break;
        case 65:
                result = x*x + 75;
                break;
        default:
                result = x + 75;
                break;
    }
    return result;
}
```

3.67:

A.

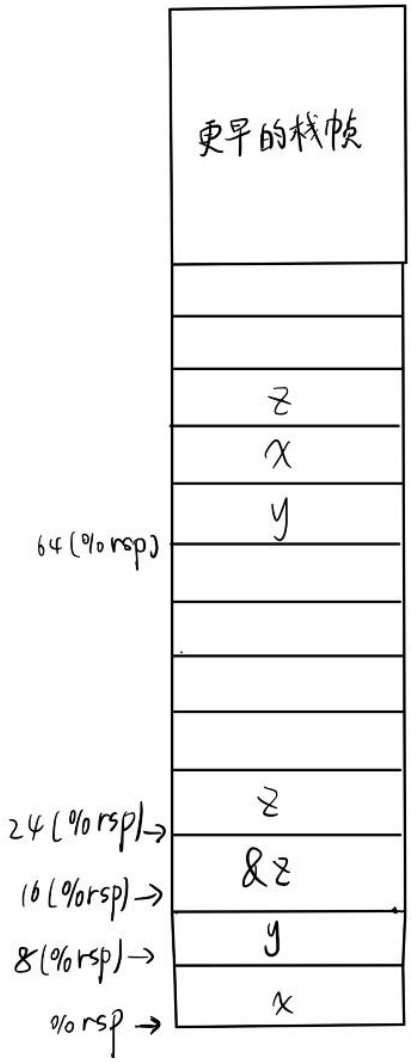


B. eval 调用 process 时传递了 64(%rsp)这个地址

C. process 的代码是通过%rbp+偏移量的方式访问结构参数 s 的元素

D. process 的代码是通过%rdi+偏移量的方式来设置结构 r 的字段

E.



地址 64(%rsp)中存了 r.u[0]，地址 72(%rsp)中存了 r.u[1]，地址 80(%rsp)中存了 r.q

F.

函数可以将结构体中的各元素存储在栈中，通过%rsp+偏移量的方式来访问结构体中的元素，从而将结构体作为参数传递或者返回。

3.71:

```
#define bufSize 12

void good_echo()
{
        char buf[bufSize];
        while (1)
                {
                        char* p=fgets(buf,bufSize,stdin);
                        if (p==NULL && ferror(stdin)!=0) break;
                        printf("%s\n",p);
                }
        return;
}
```

3.75:

A. 第 n 个参数的实部存在%xmm(2n-2)中，第 n 个参数的虚部存在%xmm(2n-1)中

B. 返回值的实部存在%xmm0，虚部存在%xmm1 中