

作业二

1190201215 冯开来

3.59

(*dest in %rdi, x in %rsi, y in %rdx)

```
movq    %rdx, %rax    // 将 y 传给 %rax
cqto                      // 将 y 有符号扩展到 128 位, 并将低位给 %rax, 符号位高位存放在 %rdx
movq    %rsi, %rcx    // 将 x 传给 %rcx
sarq    $63, %rcx    // 将 x 算数右移 63 位, 若 x 为正, %rcx 为 0, 若 x 为负, %rcx 为 -1
imulq   %rax, %rcx    // 将 y 的低位与 x 的符号位相乘, 即 %rcx = xh * yl
imulq   %rsi, %rdx    // 类似上一步, %rdx = xl * yh
addq    %rdx, %rcx    // %rcx = xl * yh + xh * yl
mulq    %rsi          // 将 %rax 与 %rsi 相乘送到 %rdx 和 %rax 中, 即 yl * xl
addq    %rcx, %rdx    // 将 %rcx 累加到 %rdx 中, 可以理解为 (xl*yh+xh*yl) * 264 + yl*xl 的高位
movq    %rax, (%rdi)  // 将 yl*xl 低 64 位送到 (%rdi) 即 dest 的低位
movq    %rdx, 8(%rdi) // 将 xl*yh+xh*yl 送到 dest 的高位
ret                                // 返回
```

因为 $x = 2^{64} * xh + xl$, $y = 2^{64} * yh + yl$, 则 $x * y = (2^{64} * xh + xl) * (2^{64} * yh + yl)$
 $= xhyh * 2^{128} + (xh*yl + xl*yh) * 2^{64} + xl*yl$
这里面 $xhyh * 2^{128}$ 必定溢出, 截断之后全是 0 不用管。

所以我们最后要算的就是 $(xh*yl + xl*yh) * 2^{64} + xl*yl$, 而这里 2^{64} 的意思就是这个二进制数的高 64 位, 存在 %rdx 中, 整个值的高 64 位是由 $xh*yl + xl*yh + xl*yl >> 64$ 决定的。所以整个值的低位 $pl = xl * xh$ 的低 64 位, $ph = xh*yl + xl*yh + xl*yl >> 64$ ($xl*yl$ 高出 64 位的部分)。

3.63

```
long switch_prob(long x, long n)
{
    long result = x;
    switch(n){
        case 60:
        case 62:
            result = x * 8;
            break;
        case 63:
            result = x / 8;
            break;
        case 64:
            result = 16 * x - x;
            x = result;
        case 65:
            x = x * x;
        default:
            result = x + 75;
    }
}
```

3.67

A:

| | | |
|--------|----|----------|
| | | |
| | | |
| | | 80(%rsp) |
| | | 72(%rsp) |
| 第9个8字节 | | 64(%rsp) |
| | | |
| | | |
| | | |
| 第4个8字节 | z | |
| | &z | |
| | y | |
| | x | %rsp |

B:

传递了 64 (%rsp)，即栈指针 rsp 开始的第 9 个 8 字节的开始地址

C:

因为这是通过堆栈传参，因为 call 指令导致返回地址压栈，rsp-8 个字节，所以 process 是通过栈指针%rsp+8+偏移量来访问的

D:

process 过程通过 rdi 寻址返回值结构体 r。rdi 寄存器中存放了返回值结构体 r 的起始地址，通过 rdi 寄存器加上偏移即可寻址返回值结构体的各个元素。

E: 调用 process 后的栈帧

| | | |
|--------|-------------------|----------|
| | | |
| | | |
| | r.q = z | 80(%rsp) |
| | r.u[1] = x | 72(%rsp) |
| 第9个8字节 | r.u[0] = y | 64(%rsp) |
| | | |
| | | |
| | | |
| 第4个8字节 | z | |
| | s.p = &z = rsp+24 | |
| | s.a[1] = y | |
| | s.a[0] = x | %rsp |

Process 的返回值结构体 r 放在了 rsp+64 开始的内存处，通过这个地址+偏移量访存。

F:

结构体无论是作为函数参数还是作为返回值，都是通过内存传参，而不是通过寄存器。作为函数参数时，在子过程中通过 rsp+偏移访问结构体参数的各个元素，作为返回值时，在调用子过程时通过 rdi 寄存器将返回值结构体在主过程中的存放的起始地址传给子过程。

3.71

```
#define SIZE 10
void good_echo()
{
    char str[SIZE];
    while (1)
    {
        fgets(str, SIZE, stdin);
        printf("%s", str);
        if (str[strlen(str) - 1] == '\n')
        {
            break;
        }
    }
}
```

3.75

A.

对于第 n 个参数,则 imag 部分传 $\text{\%xmm}(2n-1)$, real 部分传 $\text{\%xmm}(2n-2)$

B:

imag 部分返回值 \%xmm1 , real 部分返回值 \%xmm0 .