# 3.59

x\*y = (2^64\*x\_h + x\_l) \* (2^64\*y\_h +y\_l)

=2^128\*x\_h\*y\_h + 2^64\*(x\_h\*y\_l + y\_h\*x\_l) + x\_l\*y\_l

第一项2^128\*x\_h\*y\_h溢出，舍去

第二项2^64\*(x\_h\*y\_l + y\_h\*x\_l)为高64位结果

第三项x\_l\*y\_l为低64位结果

分别计算出2^64\*(x\_h\*y\_l + y\_h\*x\_l)和x\_l\*y\_l，相加，即为最终结果

dest in %rdi, x in %rdx, y in %rsi

1 store\_prod:

2 movq %rdx, %rax （rdx为x）

3 cqto （rdx为x\_h，rax为x\_l）

4 movq %rsi, %rcx （rsi为y）

5 sarq $63, %rcx （rcx为y\_h）

6 imulq %rax, %rcx （rcx=y\_h \* x\_l）

7 imulq %rsi, %rdx （rdx=x\_h \* y\_l）

8 addq %rdx, %rcx （rcx=x\_h \* y \_l + y\_h \*x\_l）

9 mulq %rsi （rax=rsi\*rax=x\_l \* y\_l，则rax为低64位结果）

10 addq %rcx, %rdx （rdx=y\_h \* x\_l + x\_h \* y\_l，rdx为高64位结果）

11 movq %rax, (%rdi) （低64位结果赋值给dest）

12 movq %rdx, 8(%rdi)（高64位结果赋值给dest地址+8）

# 3.63

long switch\_prob(long x, long n)

{

long result = x;

switch(n)

{

case 60:

case 62:

result \*= 8;

break;

case 63:

result >>= 3;

break;

case 64:

result <<= 4;

result -= x;

x = result;

case 65:

x \*= x;

case 61:

default:

result = x + 0x4b;

break;

}

return result;

}

# 3.67

A.表格中的地址指的是相对于 eval 的原始栈顶的地址，自上向下递减的

|  |  |  |
| --- | --- | --- |
| **相对地址** | **栈值** | **当前栈顶** |
| -104 | x | 🡨 |
| -96 | y |  |
| -88 | &z |  |
| -80 | z |  |
| … | 垃圾值 |  |
| 0 | eval 的原始栈顶 |  |

B.传递了整个结构体 strA，包括 x y &z

C.使用 rsp 加偏移访问

|  |  |  |
| --- | --- | --- |
| **相对地址** | **栈值** | **当前栈顶** |
| -112 | 返回地址 | 🡨 |
| -104 | x |  |
| -96 | y |  |
| -88 | &z |  |
| -80 | z |  |
| … | 垃圾值 |  |
| -40 | u[0] |  |
| -32 | u[1] |  |
| -24 | q |  |
| … | 垃圾值 |  |
| 0 | eval 的原始栈顶 |  |

D.使用 rdi 加偏移访问

E.使用 rsp 加偏移访问

|  |  |  |
| --- | --- | --- |
| **相对地址** | **栈值** | **当前栈顶** |
| -104 | x | <----- |
| -96 | y |  |
| -88 | &z |  |
| -80 | z |  |
| … | 垃圾值 |  |
| -40 | u[0] |  |
| -32 | u[1] |  |
| -24 | q |  |
| … | 垃圾值 |  |
| 0 | eval 的原始栈顶 |  |

F.由于结构体的复杂性，是通过存储在栈上访问的

# 3.71

#include<stdio.h>

#include<string.h>

int main()

{

char s[1000];

while (fgets(s, 1000, stdin) != NULL)

{

printf("%s", s);

if (ferror(stdin))

{

printf("\nError\n");

return;

}

}

return 0;

}