3.59

store\_prod

movq %rdx, %rax #rax=yl

cqto #rdx位扩展, rdx=yh

movq %rsi, %rcx #rcx=xl

sarq $63, %rcx #rcx位扩展, rcx=xh

imulq %rax, %rcx #rcx = yl\*xh，计算yl\*xh

imulq %rsi, %rdx #rdx = xl\*yh，计算xl\*yh

addq %rdx, %rcx #rcx = yl\*xh + xl\*yh，计算yl\*xh + xl\*yh

mulq %rsi #rax = yl\*xl，计算yl\*xl

addq %rcx, %rdx #rdx = yl\*xh + xl\*yh + rdx, 加上进位

movq %rax, (%rdi) #rax即为结果的低64位

movq %rdx, 8(%rdi) #rdx即为结果的高64位

ret

3.63

long switch\_prob(long x, long n)

{

long result = x;

switch(n)

{

case 60:

case 62:

result = 8\*x;

break;

case 63:

result=x/8;

break;

case 64:

result = 15\*x;

case 65:

x\*=x;

break;

default:

result = x+75;

}

return result;

}

3.67

A.

|  |  |
| --- | --- |
| 相对地址 | 内容 |
| -104(rsp) | x |
| -96 | y |
| -88 | &z rcx |
| -80 | Z rdx |
| ... | ... |
| 0 | eavl原栈顶 |

B. 传了一个相对于rsp位置为64的指针

C. 直接使用相对于rsp的偏移量访问

D. 通过设置相对于rsp的偏移量

E.

|  |  |
| --- | --- |
| 相对地址 | 内容 |
| -112(rsp) | 返回地址 |
| -104 | x |
| -96 | y |
| -88 | &z |
| -80 | z |
| ... | ... |
| -40 | u[0]=y |
| -32 | u[1]=x |
| -24 | q=Z |
| ... | ... |
| 0 | eavl原栈顶 |

eval直接使用相对于rsp的偏移量访问r的元素

F. 如果把结构作为参数，那么实际传递的会是一个空的位置指针，函数把数据存储在这个位置上，同时返回值也是这个指针。

3.71

#include <stdio.h>

#define size 10

void good\_echo()

{

char buf[size];

while(1)

{

char \*fp;

fp=fgets(buf, size, stdin);

if(fp==NULL) break;

printf("%s", fp);

}

}

int main()

{

good\_echo();

return 0;

}