# 家庭作业

3.59

根据等式：x=264·xh + xl，y=264·yh + yl，以及p= 264·ph + pl。

可以得出：p = （xh·yl + xl·yh）·264 + xl·yl。

其中，xl·yl 也可能超出64位，设它的高64位为Z1，Z0，则xl·yl = Z1·264 + Z0。

综上，p = （xh·yl + xl·yh + Z1）·264 + Z0 。

**#x in %rsi ， y in %rdx**

**movq %rdx, %rax** **# 用%rax中储存y**

**cqto**  **# 将%rax有符号扩展为%rdx: %rax，%rdx为%rax的符号**

**位相同**

**movq %rsi, %rcx** **# 用%rcx中储存x**

**sarq $63, %rcx**  **# 将x算术右移63位存入%rcx，即将x符号扩展到128**

**位，其中高64位储存在%rcx中。**

**# 此时，寄存器%rsi储存x的低64位，%rcx储存x的高64位。**

**%rax储存y的低64位，%rdx储存y的高64位。**

**imulq %rax, %rcx # %rcx中存放xh·yl**

**imulq %rsi, %rdx # %rdx中存放xl·yh**

**addq %rdx, %rcx # %rcx存放xh·yl + xl·yh**

**mulq %rsi # 128位无符号全乘法，计算xl·yl ,结果的高64位存储**

**在：%rdx，低64位存储在:%rax**

**即%rdx中存放Z1，%rax中存放Z0**

**addq %rcx, %rdx # %rdx中存放xh·yl + xl·yh + Z1**

**movq %rax, (%rdi) # 将%rax中的值，即Z0存储在低64位**

**movq %rdx, 8(%rdi) # 将%rdx中的值，即xh·yl + xl·yh + Z1存储在高64位**

**ret**

3.63

答案：

long switch\_prob(long x,long n)

{

    long result = x;

    switch(n) {

    case 60:

    case 62:

        result = 8\*x;

        break;

    case 63:

        result = x>>3;

        break;

    case 64:

        x=225\*x; //[(x<<4)-x]\*[(x<<4)-x]

        result = x+75;

        break;

    case 65:

        x=x\*x;

        result = x+75;

        break;

    default:

        result = x+75;

        break;

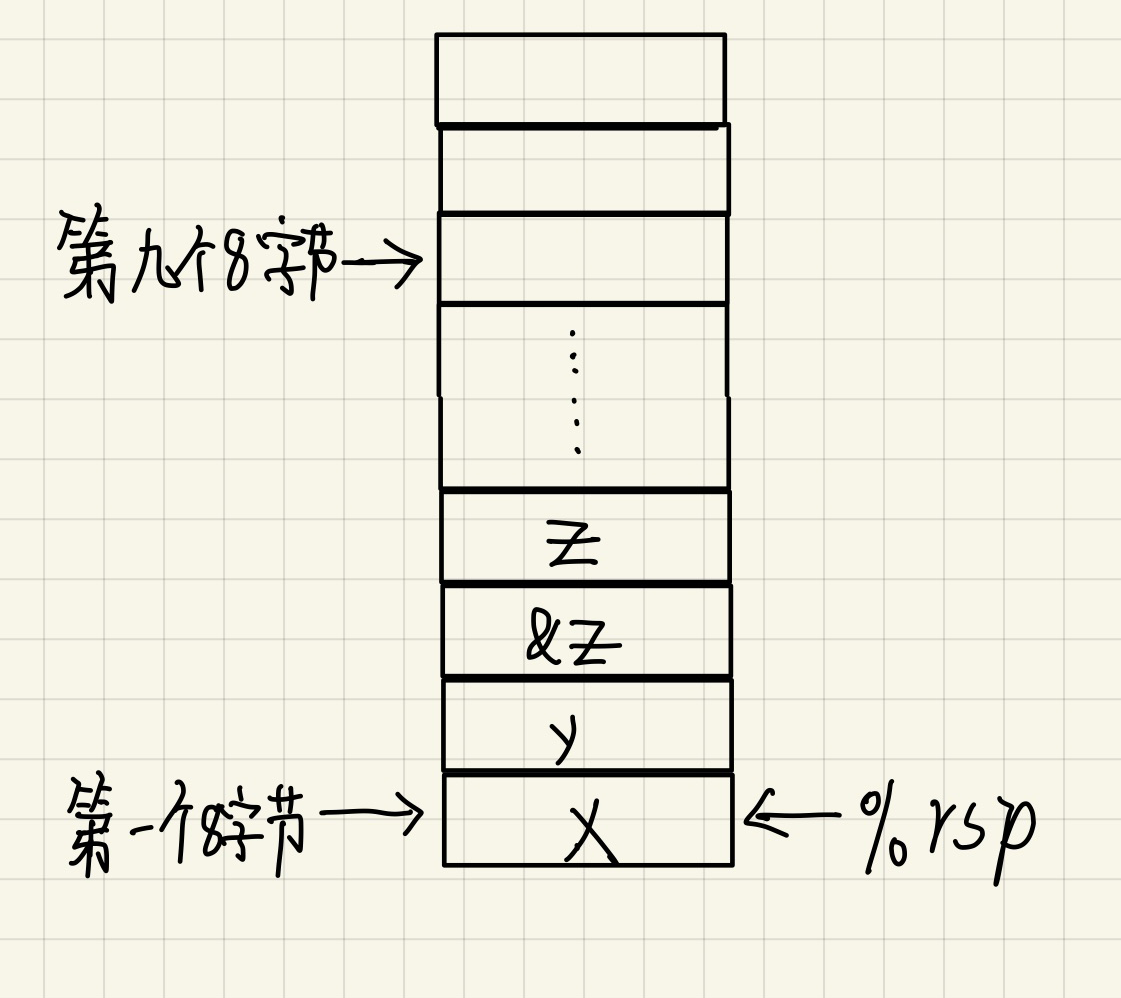
    }

    return result;

}

3.67

A：

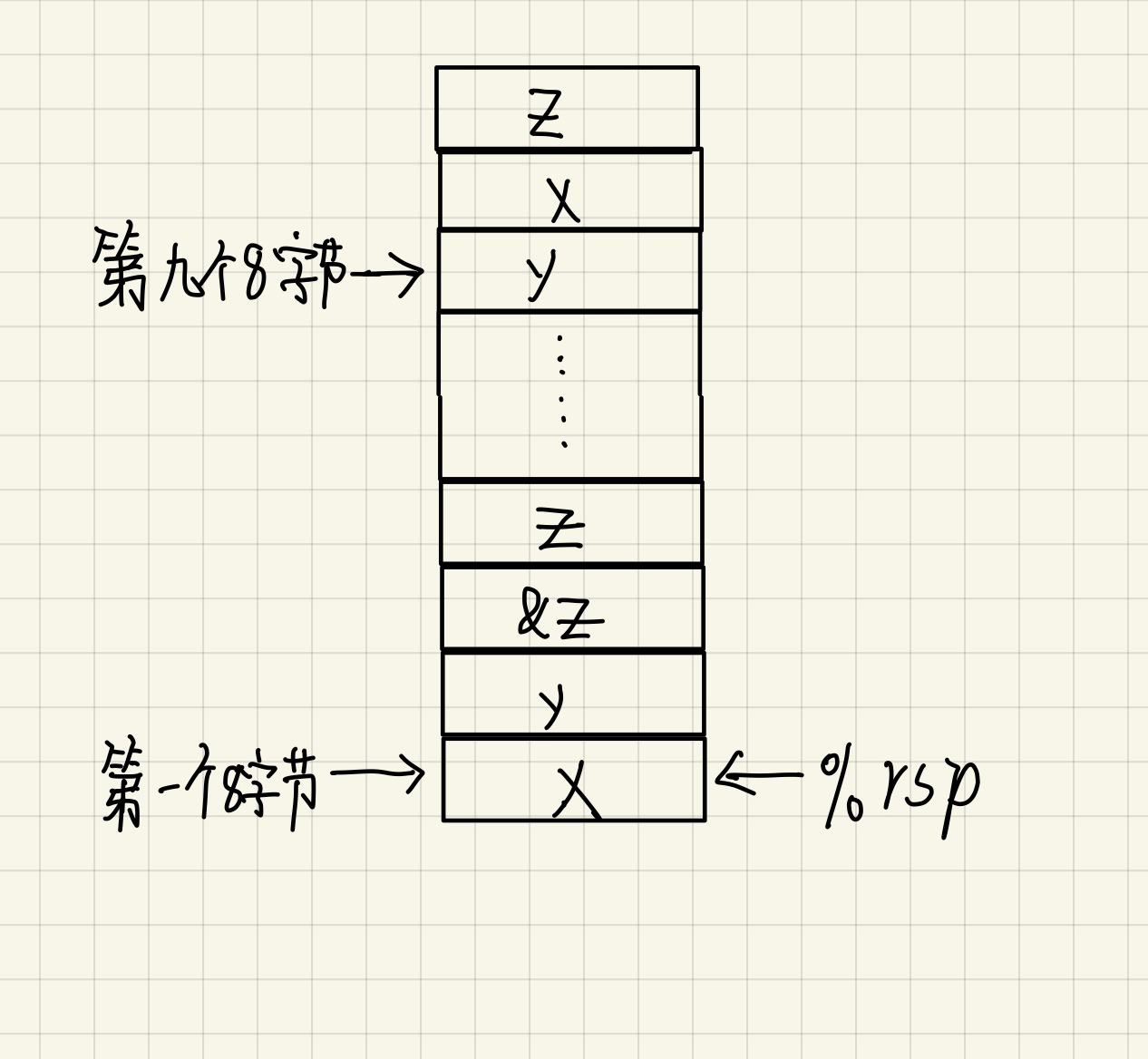


B：%rsp+64

C：s保存在栈里，用%rsp+偏移量来访问的

D：r分配在栈里，也是%rsp+偏移量来设置的。

E：



F：调用者给被调用者传递地址和需要的空间大小，被调用者存储数据后，返回传入的地址值。

3.71

#include <stdio.h>

#define BUFFER\_SIZE 114

void good\_echo(void){

    char buf[BUFFER\_SIZE];

    while(1){

        char\* p = fgets(buf,BUFFER\_SIZE,stdin);

        if(p==NULL){

            break;

        }

        printf("%s",p);

    }

    return;

}

int main(){

    good\_echo();

    return 0;

}