# 3.59

store\_prod:*;%rdi: \*dest, %rsi x, %rdx y*

    movq %rdx, %rax*;将y复制到%rax*

    cqto*;符号扩展%rax的64位值[y]为%rdx:%rax的128位值Y:y*

    movq %rsi, %rcx*;把x复制到%rcx*

    sarq $63, %rcx*;x算术右移63位，这样%rcx:%rsi就是全128位值X:x*

*;\*dest=[x]\*[y]=(Xw+x)(Yw+y)=XYww+(Xy+xY)w+xy=(Xy+xY)w+xy*

    imulq %rax, %rcx*;计算Xy，保存至%rcx*

    imulq %rsi, %rdx*;计算xY，保存至%rdx*

    addq %rdx, %rcx*;计算Xy+xY，保存至%rcx*

    mulq %rsi*;无符号计算xy(结果为128位)，保存至%rdx:%rax*

    addq %rcx, %rdx*;将xy和w(Xy+xY)的高64位相加*

    movq %rax, (%rdi)*;把低64位保存至[%rdi]*

    movq %rdx, 8(%rdi)*;高64位保存至[%rdi+8]*

    ret*;返回*

# 3.63

long switch\_prob(long x, long n){

    long result = x;

*switch* (n){

*case* 60:

*case* 62:

        result = 8 \* x;

*break*;

*case* 61:

*default*:

        result = x - 0x4b;

*break*;

*case* 63:

        result = x >> 3;

*break*;

*case* 64:

        x = ((x << 4) - x);

*case* 65:

        result = x \* x + 0x4b;

*break*;

    }

*return* result;

}

# 3.67

## A

|  |  |
| --- | --- |
| 0 | x |
| 4 |
| 8 | y |
| 12 |
| 16 | &z |
| 20 |
| 24 | z |
| 28 |
| 32 |  |
| 36 |  |
| 40 |  |
| 44 |  |
| 48 |  |
| 52 |  |
| 56 |  |
| 60 |  |
| 64 |  |
| 68 |  |
| 72 |  |
| 76 |  |
| 80 |  |
| 84 |  |
| 88 |  |
| 92 |  |
| 96 |  |
| 100 |  |
| 104 |  |

## B

Rbp+64的位置的地址。

## C

通过地址来访问相应的元素

## D

将值写入到栈中给定的位置(rdi+x=rbp+64+x)

## E

|  |  |
| --- | --- |
| 0 | x(s.a[0]) |
| 4 |
| 8 | y(s.a[1]) |
| 12 |
| 16 | &z(s.p) |
| 20 |
| 24 | z |
| 28 |
| 32 |  |
| 36 |  |
| 40 |  |
| 44 |  |
| 48 |  |
| 52 |  |
| 56 |  |
| 60 |  |
| 64 | s.a[1](r.u[0]) |
| 68 |
| 72 | s.a[0](r.u[1]) |
| 76 |
| 80 | \*s.p(r.q) |
| 84 |
| 88 |  |
| 92 |  |
| 96 |  |
| 100 |  |
| 104 |  |

## F

作为参数传递的结构体的内容被顺序放置在栈顶，在汇编中没有显式传递，而储存返回值的结构体的位置由调用者分配，并在汇编中把相应的位置指针显示传递给被调用者。被调用者将内容写入该区域，并返回相同的位置指针。

# 3.71

*#include* <stdio.h>

*#define* BUFFER\_SIZE 1024

void good\_echo(){

    static char buffer[BUFFER\_SIZE];

    int bufferptr;

    int flag = 1;

*while*(flag && fgets(buffer, BUFFER\_SIZE, stdin)) {

        printf("%s", buffer);

*for* (bufferptr = 0; bufferptr < BUFFER\_SIZE; bufferptr++) {

*if* (buffer[bufferptr] == '\n') {

                flag = 0;

*break*;

            }

        }

    }

*return*;

}

int main()

{

    good\_echo();

*return* 0;

}

# 3.75

## A

复参数被放在两个相邻的浮点寄存器中（如：x=xmm0 + j\*xmm1）

## B

负数返回值为x=xmm0 + j\*xmm1