作者：代兴　邮箱：503268771@qq.com　Vx 公众号：东方架构师

指令说明

函数指令包括函数调用指令 callq、函数退出指令 retq、栈帧操作指令 pushq 和 popq。
callq 指令，把 rip 入栈，然后跳转到指定的函数地址。比如，`callq func_sum`。
retq 指令，把 rip 出栈，然后跳转到 rip 指定的地址。比如，`retq`。
pushq 指令，把某值入栈。比如，`pushq %rbp`。
popq 指令，把某值出栈。比如，`popq %rbp`。

函数定义的基本格式：

```
func_name :           # 函数名称
    pushq %rbp          # rbp 入栈
    movq %rsp, %rbp     # rsp 赋给 rbp

    # 其他指令。这里省略。

    popq %rbp       # rbp 出栈
    retq
```

用汇编代码分析

编写代码： sum_func.s

```
.data

int64_aa :      # 64 位整数
    .quad 0x0

float64_bb :    # 64 位浮点数
    .double 0

str_tip :       # 输入 1 个整数、1 个浮点数
    .string "Please input like : 300  55.66 \n"

str_input :     # 输入 1 个整数、1 个浮点数
    .string "%lld %lf"

str_sum :       # 结果
    .string "Sum = %lf \n"

.text
.global main

main :
    pushq %rbp
```

```
    movq %rsp, %rbp

    # 提示
    movq $str_tip, %rdi     # 参数 1
    callq printf            # 调用输出函数

    # 输入
    movq $str_input, %rdi   # 参数 1
    movq $int64_aa, %rsi    # 参数 2，整数变量的地址
    movq $float64_bb, %rdx  # 参数 3，浮点数变量的地址
    callq scanf             # 调用输入函数

    movq int64_aa(%rip), %rdi       # 参数 1，整数
    movsd float64_bb(%rip), %xmm0   # 参数 2，浮点数
    callq func_sum                  # 调用函数

    popq %rbp
    retq

func_sum :          # 自定义函数。sum = 整数 + 浮点数
    pushq %rbp          # rbp 入栈
    movq %rsp, %rbp     # rsp 赋给 rbp

    movq %rdi, %r8          # 参数 1，整数
    movsd %xmm0, %xmm6      # 参数 2，浮点数

    cvtsi2sd %r8, %xmm5     # 整数转浮点数
    addsd %xmm5, %xmm6      # 浮点数加法

    movq $str_sum, %rdi     # 参数 1，字符串
    movsd %xmm6, %xmm0      # 参数 2，浮点数
    callq printf            # 调用输出函数

    popq %rbp       # rbp 出栈
    retq            # 退出函数。
```

编译代码：

```
gcc sum_func.s -o sum_func
```

运行代码：

```
[root@local func]# ./sum_func
Please input like : 300  55.66
100 22.33
Sum = 122.330000

[root@local func]# ./sum_func
Please input like : 300  55.66
-500 -66.77
```

```
Sum = -566.770000
```

分析结果：
代码规则为，定义 1 个函数，输入 1 个整数、1 个浮点数，输出两数之和。
输入 100 22.33，输出 122.330000。
输入-500 -66.77，输出-566.770000。