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

浮点数计算器的功能

浮点数计算器，输入 2 个浮点数和 1 个运算符，执行浮点数运算，输出结果。
功能包括输入、输出、加法指令、减法指令、乘法指令、除法指令、函数指令、寻址指令、比较指令、跳转指令等。

浮点数计算器的实现

编写代码： float_computer.s

```
.data

float64_aa :  # 64 位浮点数
    .double 0x1

float64_bb :  # 64 位浮点数
    .double 0x2

float64_xx :  # 运算的结果
    .double 0x3

input_op :  # 运算符
    .byte 0x0,0x0,0x0,0x0,0x0

str_tips :  # 提示
    .string ">> Please input like  1.2 * 2.3 \n"

str_read :  # 输入
    .string "%lf %s %lf"

str_tmp :
    .string "operator :   %s \n"

str_out : # 输出
    .string "output   :   %f  %s  %f  =  %f \n\n"

str_error :  # 错误
    .string " wrong params :   %f  %s  %f \n\n"

# 运算符
op_add : .string "+"    # 加法
op_sub : .string "-"    # 减法
op_mul : .string "*"    # 乘法
op_div : .string "/"    # 除法
```

```
.text
.global main

main :
    pushq %rbp
    movq %rsp, %rbp

    # 提示
    movq $str_tips, %rdi
    callq printf

    # 输入
    movq $str_read , %rdi
    movq $float64_aa, %rsi    # 变量 aa
    movq $input_op, %rdx    # 运算符
    movq $float64_bb, %rcx    # 变量 bb
    callq scanf

    # 加法
    movq $input_op, %rdi
    movq $op_add, %rsi
    callq strcmp        # 比较运算符
    cmpq $0, %rax
    je mark_add

    # 减法
    movq $input_op, %rdi
    movq $op_sub, %rsi
    callq strcmp        # 比较运算符
    cmpq $0, %rax
    je mark_sub

    # 乘法
    movq $input_op, %rdi
    movq $op_mul, %rsi
    callq strcmp        # 比较运算符
    cmpq $0, %rax
    je mark_mul

    # 除法
    movq $input_op, %rdi
    movq $op_div, %rsi
    callq strcmp        # 比较运算符
    cmpq $0, %rax
    je mark_div

    # 参数错误
    jmp mark_error
```

```
mark_add :
    movsd float64_aa(%rip), %xmm0    # 变量 aa
    movsd float64_bb(%rip), %xmm1    # 变量 bb
    addsd %xmm1, %xmm0               # xmm0 = xmm0 + xmm1
    movsd %xmm0, float64_xx(%rip)    # 保存结果
    jmp mark_out

mark_sub :
    movsd float64_aa(%rip), %xmm0    # 变量 aa
    movsd float64_bb(%rip), %xmm1    # 变量 bb
    subsd %xmm1, %xmm0               # xmm0 = xmm0 - xmm1
    movsd %xmm0, float64_xx(%rip)    # 保存结果
    jmp mark_out

mark_mul :
    movsd float64_aa(%rip), %xmm0    # 变量 aa
    movsd float64_bb(%rip), %xmm1    # 变量 bb
    mulsd %xmm1, %xmm0               # xmm0 = xmm0 * xmm1
    movsd %xmm0, float64_xx(%rip)    # 保存结果
    jmp mark_out

mark_div :
    movsd float64_aa(%rip), %xmm0    # 变量 aa
    movsd float64_bb(%rip), %xmm1    # 变量 bb
    divsd %xmm1, %xmm0               # xmm0 = xmm0 / xmm1
    movsd %xmm0, float64_xx(%rip)    # 保存结果
    jmp mark_out

mark_error :  # 错误
    movq $str_error, %rdi
    movsd float64_aa(%rip), %xmm0
    movq $input_op, %rsi
    movsd float64_bb(%rip), %xmm1
    callq printf
    jmp mark_last

mark_out : # 输出
    callq func_print_out  # 调用函数

mark_last :

    popq %rbp
    retq

# ---------   输出函数
func_print_out :
    pushq %rbp
    movq %rsp, %rbp
```

```
    movq $str_tmp, %rdi
    movq $input_op, %rsi
    callq printf

    movq $str_out, %rdi
    movsd float64_aa(%rip), %xmm0   # 变量 aa
    movq $input_op, %rsi            # 运算符
    movsd float64_bb(%rip), %xmm1   # 变量 bb
    movsd float64_xx(%rip), %xmm2   # 结果
    callq printf

    popq %rbp
    retq
```

编译代码：

```
gcc float_computer.s -o float_computer
```

运行代码：

```
[root@local zong]# ./float_computer
>> Please input like  1.2 * 2.3
22.2 + 33.3
operator :   +
output   :   22.200000  +  33.300000  =  55.500000

[root@local zong]# ./float_computer
>> Please input like  1.2 * 2.3
22.2 - 33.3
operator :   -
output   :   22.200000  -  33.300000  =  -11.100000

[root@local zong]# ./float_computer
>> Please input like  1.2 * 2.3
22.2 * -6
operator :   *
output   :   22.200000  *  -6.000000  =  -133.200000

[root@local zong]# ./float_computer
>> Please input like  1.2 * 2.3
22.2 / 3
operator :   /
output   :   22.200000  /  3.000000  =  7.400000
```

分析结果：

加法。输入 `22.2 + 33.3` ，输出 `22.200000  +  33.300000  =  55.500000` 。
减法。输入 `22.2 - 33.3` ，输出 `22.200000  -  33.300000  =  -11.100000` 。
乘法。输入 `22.2 * -6` ，输出 `22.200000  *  -6.000000  =  -133.200000` 。
除法。输入 `22.2 / 3` ，输出 `22.200000  /  3.000000  =  7.400000` 。

汇编代码	结果和分析
float64_aa : # 64 位浮点数 .double 0x1 float64_bb : # 64 位浮点数 .double 0x2 float64_xx : # 运算的结果 .double 0x3 input_op : # 运算符 .byte 0x0,0x0,0x0,0x0,0x0	用变量承接 2 个输入浮点数、1 个运算符、1 个输出浮点数。 运算符，用字节数组变量。
# 运算符 op_add : .string "+" # 加法 op_sub : .string "-" # 减法 op_mul : .string "*" # 乘法 op_div : .string "/" # 除法	运算符包括加法、减法、乘法、除法。 运算符格式为字符串。

# 输入 	
movq $str_read , %rdi
movq $float64_aa, %rsi # 变量 aa
movq $input_op, %rdx # 运算符
movq $float64_bb, %rcx # 变量 bb
callq scanf
	使用 scanf 函数，输入变量。

# 加法 	
movq $input_op, %rdi
movq $op_add, %rsi
callq strcmp # 比较运算符
cmpq $0, %rax
je mark_add
	使用 strcmp 函数，比较运算符和加法字符串。
	如果返回值等于 0，说明字符串匹配，跳转到加法语句块。
	其他运算符，也是类似逻辑。

mark_add :
movsd float64_aa(%rip), %xmm0 # 变量 aa
movsd float64_bb(%rip), %xmm1 # 变量 bb
addsd %xmm1, %xmm0 # xmm0 = xmm0 + xmm1
movsd %xmm0, float64_xx(%rip) # 保存结果
jmp mark_out
	加法语句块。
	把 2 个浮点数相加,结果写到变量 float64_xx。
	跳转到输出语句块。
	其他运算符，也是类似逻辑。