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

指令说明

移动指令，包括整数移动指令、浮点数移动指令等。
整数移动指令，分为 8 位 movb、16 位 movw、32 位 movl、64 位 movq。
浮点数移动指令，分为 32 位 movss、64 位 movsd 等。(打包指令 movps 等，这里不讨论。)
整数移动指令，可以操作立即数、寄存器、内存。
浮点数移动指令，可以操作寄存器、内存。

读写变量，使用 rip 相对寻址。
`movq $0x7172, int64_aa(%rip)` 表示把值 0x7172 写到变量 int64_aa。
`movq int64_aa(%rip), %rax` 表示把变量 int64_aa 的值写到变量 rax。
`movl %ebx, int32_bb+4(%rip)` 表示把 ebx 的值写到变量 int32_bb 的地址偏移 4 字节对应的内存。

读写栈内存，使用 rbp、rsp。
`movq $0x7172, -16(%rbp)` 表示把值 0x7172 写到内存(rbp-16)。
`movq -16(%rbp), %rax` 表示把内存(rbp-16)的值写到 rax。

整数移动指令

编写代码： int_mov.s

```
.data

int64_aa :      # 64 位整数
    .quad 0x1

int32_bb :      # 32 位整数数组
    .long 0x2   # 元素 1
    .long 0x3   # 元素 2

.text
.global main

main :
    pushq %rbp
    movq %rsp, %rbp
    subq $64, %rsp

    # 立即数与寄存器、内存
    movq $0x7172, %rax              # 64 位，把 0x7172 写到 rax
    movq $0x7172, int64_aa(%rip)    # 64 位，把 0x7172 写到变量 int64_aa
    movq $0x7172, -16(%rbp)         # 64 位，把 0x7172 写到内存(rbp-16)
    movl $0x6162, int32_bb+4(%rip)  # 32 位，把 0x6162 写到变量(int32_bb+4)
    movl $0x6162, %ecx      # 32 位，把 0x6162 写到 ecx
    movw $0x6162, %cx       # 16 位，把 0x6162 写到 cx
```

```
    movw $0x6162, -8(%rbp)  # 16 位，把 0x6162 写到内存(rbp-8)
    movb $0x61, %cl         # 8 位，把 0x61 写到 cl
    movb $0x61, -8(%rbp)    # 8 位，把 0x61 写到内存(rbp-8)

    # 寄存器与寄存器
    movq %rcx, %rax     # 64 位，把 rcx 写到 rax
    movl %ecx, %eax     # 32 位，把 ecx 写到 eax
    movw %cx, %ax       # 16 位，把 cx 写到 ax
    movb %cl, %al       # 8 位，把 cl 写到 al
    movb %ch, %ah       # 8 位，把 ch 写到 ah

    # 寄存器与变量内存
    movq %rax, int64_aa(%rip)   # 64 位，把 rax 写到变量 int64_aa
    movq int64_aa(%rip), %rax   # 64 位，把变量 int64_aa 写到 rax
    movl %ebx, int32_bb+4(%rip) # 32 位，把 ebx 写到变量(int32_bb+4)
    movl int32_bb+4(%rip), %ebx # 32 位，把变量(int32_bb+4)写到 ebx
    movw %bx, int32_bb(%rip)    # 16 位，把 bx 写到变量 int32_bb
    movw int32_bb(%rip), %bx    # 16 位，把变量 int32_bb 写到 bx
    movb %bl, int32_bb(%rip)    # 8 位，把 bl 写到变量 int32_bb
    movb int32_bb(%rip), %bl    # 8 位，把变量 int32_bb 写到 bl

    # 寄存器与栈内存
    movq %rax, -16(%rbp)    # 64 位，把 rax 写到内存(rbp-16)
    movq -16(%rbp), %rax    # 64 位，把内存(rbp-16)写到 rax
    movl %eax, -16(%rbp)    # 32 位，把 eax 写到内存(rbp-16)
    movl -16(%rbp), %eax    # 32 位，把内存(rbp-16)写到 eax
    movw %ax, -16(%rbp)     # 16 位，把 ax 写到内存(rbp-16)
    movw -16(%rbp), %ax     # 16 位，把内存(rbp-16)写到 ax
    movb %al, -16(%rbp)     # 8 位，把 al 写到内存(rbp-16)
    movb -16(%rbp), %al     # 8 位，把内存(rbp-16)写到 al

    addq $64, %rsp
    popq %rbp
    retq
```

编译代码：

```
gcc int_mov.s -o int_mov
```

分析结果：

汇编代码	分析

movq $0x7172, %rax
movq $0x7172, int64_aa(%rip)
movl $0x6162, int32_bb+4(%rip)
movw $0x6162, %cx
movw $0x6162, -8(%rbp)
	把立即数移动到寄存器。
	把立即数移动到变量内存。
	把立即数移动到栈内存。

movq %rcx, %rax
movl %ecx, %eax
movw %cx, %ax
	把值从寄存器移动到寄存器。

movb %cl, %al	

movq %rax, int64_aa(%rip)
movl int32_bb+4(%rip), %ebx
movw int32_bb(%rip), %bx
movb %bl, int32_bb(%rip)
	把值从寄存器移动到变量内存。
	把值从变量内存移动到寄存器。

movq %rax, -16(%rbp)
movl -16(%rbp), %eax
movw %ax, -16(%rbp)
movb -16(%rbp), %al
	把值从寄存器移动到栈内存。
	把值从栈内存移动到寄存器。

整数的通用寄存器为64位。
8位movb指令，操作低8位、次低8位。比如，`movb %cl, %al` ，`movb %ch, %ah` 。
16位movw指令，操作低16位。比如，`movw $0x6162, %cx` 。
32位movl指令，操作低32位。比如，`movl %ebx, int32_bb+4(%rip)` 。
64位movq指令，操作64位。比如，`movq %rax, -16(%rbp)` 。

浮点数移动指令

编写代码：float_mov.s

```
.data

float64_aa :        # 64位浮点数数组
    .double 22.22   # 元素1
    .double 33.33   # 元素2

float32_bb :        # 32位浮点数
    .float 55.55

.text
.global main

main :
    pushq %rbp
    movq %rsp, %rbp
    subq $64, %rsp

    # 寄存器与寄存器
    movsd %xmm2, %xmm3      # 64位，把xmm2写到xmm3
    movss %xmm5, %xmm6      # 32位，把xmm5写到xmm6

    # 寄存器与变量内存
    movsd float64_aa(%rip), %xmm3   # 64位，把变量float64_aa写到xmm3
    movsd %xmm3, float64_aa(%rip)   # 64位，把xmm3写到变量float64_aa
    movsd float64_aa+8(%rip), %xmm3 # 64位，把变量(float64_aa+8)写到xmm3
    movsd %xmm3, float64_aa+8(%rip) # 64位，把xmm3写到变量(float64_aa+8)
    movss float32_bb(%rip), %xmm5   # 32位，把变量float32_bb写到xmm5
```

```
    movss %xmm5, float32_bb(%rip)    # 32 位，把 xmm5 写到变量 float32_bb

    # 寄存器与栈内存
    movsd -8(%rbp), %xmm5    # 64 位，把内存(rbp-8)写到 xmm5
    movsd %xmm5, -8(%rbp)    # 64 位，把 xmm5 写到内存(rbp-8)
    movss -16(%rbp), %xmm6   # 32 位，把内存(rbp-16)写到 xmm6
    movss %xmm6, -16(%rbp)   # 32 位，把 xmm6 写到内存(rbp-16)

    addq $64, %rsp
    popq %rbp
    retq
```

编译代码：

```
gcc float_mov.s -o float_mov
```

分析结果：

汇编代码	分析
`movsd %xmm2, %xmm3` `movss %xmm5, %xmm6`	把值从寄存器移动到寄存器。 `movsd %xmm2, %xmm3` 表示把 xmm2 的低 64 位，移动到 xmm3 的低 64 位。 `movss %xmm5, %xmm6` 表示把 xmm5 的低 32 位，移动到 xmm6 的低 32 位。

`movsd float64_aa(%rip), %xmm3`
`movsd %xmm3, float64_aa(%rip)`
`movss float32_bb(%rip), %xmm5`
`movss %xmm5, float32_bb(%rip)`
	把值从寄存器移动到变量内存。
	把值从变量内存移动到寄存器。
	
	`movsd float64_aa(%rip), %xmm3` 表示把 float64_aa 的值，移动到 xmm3 的低 64 位。
	`movss %xmm5, float32_bb(%rip)` 表示把 xmm5 的低 32 位，移动到 float32_bb。

`movsd -8(%rbp), %xmm5`
`movsd %xmm5, -8(%rbp)`
`movss -16(%rbp), %xmm6`
`movss %xmm6, -16(%rbp)`
	把值从寄存器移动到栈内存。
	把值从栈内存移动到寄存器。
	
	`movsd %xmm5, -8(%rbp)` 表示把 xmm5 的低 64 位，移动到(rbp-8)的栈内存。
	`movss -16(%rbp), %xmm6` 表示把(rbp-16)的栈内存，移动到 xmm6 的低 32 位。

浮点数寄存器 xmm 为 128 位。

32 位 movss 指令，操作 xmm 的低 32 位。比如，`movss %xmm5, float32_bb(%rip)` 。

64 位 movsd 指令，操作 xmm 的低 64 位。比如，`movsd %xmm5, -8(%rbp)` 。