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

寻址指令，使用 movq、leaq。
绝对寻址，取地址使用 movq $param。比如， `movq $int64_arr, %r8` 。
相对寻址，取地址使用 leaq param(%rip)。比如， `leaq int64_arr+8(%rip), %r8` ，其中 8 表示变量 int64_arr 的地址偏移 8 个字节。
绝对寻址，取值使用 movq (%reg)。比如， `movq (%r8), %r9` ，其中 r8 存的某个地址，(%r8)表示取地址的值。
相对寻址，取值使用 movq param(%rip)。比如， `movq int64_arr+8(%rip)` ，其中 8 表示变量 int64_arr 的地址偏移 8 个字节。

编写代码：mem_addr.s

```
.data

int64_arr :     # 整数数组。多个元素。
    .quad 0xA1A2A3A4A5A6A7A8    # 元素 1
    .quad 0xB1B2B3B4B5B6B7B8    # 元素 2
    .quad 0xC1C2C3C4C5C6C7C8    # 元素 3

str_addr :  # 数组的地址
    .string " Array   addr = %#llX  \n"

str_element :   # 元素的地址和值
    .string " Element   addr = %#llX   value = %#llX  \n"

.text
.global main

main :
    pushq %rbp
    movq %rsp, %rbp

    # 打印数组的地址
    movq $int64_arr, %r8        # 绝对寻址。取符号的地址
    movq %r8, %rdi
    callq func_print_addr

    # 打印数组的地址
    leaq int64_arr(%rip), %r9   # 相对寻址。取符号的地址
    movq %r9, %rdi
    callq func_print_addr

    # 打印第 2 个元素
    movq $int64_arr, %r8        # 绝对寻址
    addq $8, %r8                # 地址加上偏移 8
    movq (%r8), %r9             # 取值
    movq %r8, %rdi
    movq %r9, %rsi
    callq func_print_element
```

```
    # 打印第 2 个元素
    leaq int64_arr+8(%rip), %r8    # 相对寻址
    movq int64_arr+8(%rip), %r9    # 取值
    movq %r8, %rdi
    movq %r9, %rsi
    callq func_print_element

    popq %rbp
    retq

# -----------------
func_print_addr :   # 函数，打印数组的地址
    pushq %rbp
    movq %rsp, %rbp

    movq %rdi, %rsi     # 入参。数组的地址
    movq $str_addr, %rdi
    callq printf

    popq %rbp
    retq

# -----------------
func_print_element :    # 函数，打印某个元素的地址和值
    pushq %rbp
    movq %rsp, %rbp

    movq %rdi, %r8      # 入参。某个元素的地址
    movq %rsi, %r9      # 入参。某个元素的值

    movq $str_element, %rdi
    movq %r8, %rsi
    movq %r9, %rdx
    callq printf

    popq %rbp
    retq
```

编译代码：

```
gcc mem_addr.s -o mem_addr
```

运行代码：

```
[root@local addr]# ./mem_addr
 Array   addr = 0X601034
 Array   addr = 0X601034
 Element   addr = 0X60103C   value = 0XB1B2B3B4B5B6B7B8
 Element   addr = 0X60103C   value = 0XB1B2B3B4B5B6B7B8
```

分析结果：

汇编代码	结果和分析
`movq $int64_arr, %r8 # 绝对寻址。取符号的地址`	`Array addr = 0X601034` 绝对寻址。 把符号的地址写到 r8。结果为 0X601034。
`leaq int64_arr(%rip), %r9 # 相对寻址。取符号的地址`	`Array addr = 0X601034` 相对寻址。 把符号的地址写到 r9。结果为 0X601034。

`# 打印第 2 个元素`
`movq $int64_arr, %r8 # 绝对寻址`
`addq $8, %r8 # 地址加上偏移 8`
`movq (%r8), %r9 # 取值`
	`Element addr = 0X60103C value = 0XB1B2B3B4B5B6B7B8`
	
	绝对寻址。
	首先，把符号的地址写到 r8。
	然后，加上元素的偏移 8 个字节，即为元素的地址。
	之后，使用 movq 取元素的值。

`# 打印第 2 个元素`
`leaq int64_arr+8(%rip), %r8 # 相对寻址`
`movq int64_arr+8(%rip), %r9 # 取值`
	`Element addr = 0X60103C value = 0XB1B2B3B4B5B6B7B8`
	
	相对寻址。
	使用 leaq 取元素的地址。使用 8 个字节的偏移。
	使用 movq 取元素的值。使用 8 个字节的偏移。

取地址，绝对寻址 `movq $int64_arr, %r8` ，相对寻址 `leaq int64_arr(%rip), %r9` ，结果都为 0X601034，说明功能等价。
取值，绝对寻址 `movq (%r8), %r9` ，相对寻址 `movq int64_arr+8(%rip), %r9` ，结果都为 0XB1B2B3B4B5B6B7B8，说明功能等价。