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

指令说明

cmpxchg 指令表示比较并交换。
cmpxchg 指令分为 8 位 cmpxchgb、16 位 cmpxchgw、32 位 cmpxchgl、64 位 cmpxchgq。
cmpxchg 指令可以操作寄存器、内存。
cmpxchg 指令的含义为，给定一个变量，输入旧值、新值，比较变量和旧值，如果相等就把新值赋给变量，否则不变动。

xchg 指令表示交换。
xchg 指令分为 8 位 xchgb、16 位 xchgw、32 位 xchgl、64 位 xchgq。
xchg 指令可以操作寄存器、内存。
xchg 指令的含义为，交换 2 个值。

cmpxchg 指令的使用步骤：

```
    movq %r9, %rax       # 旧值写到 rax
    cmpxchgq %r10, (%r8) # 执行 cas
    cmpq %r9, %rax      # 判断 rax == oldValue
    je tmp_yes          # 相等，则 cas 成功
    jne tmp_no          # 不相等，则 cas 失败
```

r8 表示变量的地址，r9 表示旧值，r10 表示新值。执行 cmpxchgq，如果 rax 等于旧值，表示 cas 成功。

cmpxchg 指令

编写代码： cmpxchg.s

```
.data

int64_cas :      # 等待 cas 的变量
    .quad 0x0

int64_old :     # 输入的旧值
    .quad 0x0

int64_new :     # 输入的新值
    .quad 0x0

str_tip :
    .string "Please input like : 0 3 \n"

str_input :     # 输入
    .string "%lld %lld"

str_cas_yes :   # 成功
    .string "CAS = yes . current = %lld  rax = %lld  \n"
```

```
str_cas_no :    # 失败
    .string "CAS = no  . current = %lld  rax = %lld  \n"

.text
.global main

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

    # 提示
    movq $str_tip, %rdi
    callq printf

    movq $0, -8(%rbp)   # 记录循环次数

tmp_loop :      # 循环
    cmpq $3, -8(%rbp)   # 判断循环次数
    jge tmp_loop_out    # 退出循环
    incq -8(%rbp)       # 循环次数+1

    movq $str_input, %rdi
    movq $int64_old, %rsi   # 输入旧值
    movq $int64_new, %rdx   # 输入新值
    callq scanf             # 输入

    movq $int64_cas, %rdi       # 地址
    movq int64_old(%rip), %rsi  # 旧值
    movq int64_new(%rip), %rdx  # 新值
    callq func_cmpxchg          # 执行 cas

    jmp tmp_loop    # 继续循环

tmp_loop_out :

    addq $64, %rsp
    popq %rbp
    retq

# ----------------
func_cmpxchg :      # 函数，执行 cas 操作
    pushq %rbp
    movq %rsp, %rbp
    subq $64, %rsp

    movq %rdi, %r8      # 地址
    movq %rsi, %r9      # 旧值
```

```
    movq %rdx, %r10     # 新值

    movq %r9, %rax       # 旧值写到 rax
    cmpxchgq %r10, (%r8) # 执行 cas

    movq %rax, -8(%rbp) # 保存 rax

    cmpq %r9, %rax      # 判断 rax == oldValue
    je tmp_yes          # 相等，则 cas 成功
    jne tmp_no          # 不相等，则 cas 失败

tmp_yes :
    movq $str_cas_yes, %rdi
    jmp tmp_out

tmp_no :
    movq $str_cas_no, %rdi
    jmp tmp_out

tmp_out :
    movq (%r8) , %rsi   # 取当前的值
    movq -8(%rbp), %rdx # 取 rax
    callq printf

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

编译代码：

```
gcc cmpxchg.s -o cmpxchg
```

运行代码：

```
[root@local cas]# ./cmpxchg
Please input like : 0 3
0 5
CAS = yes . current = 5  rax = 0
5 7
CAS = yes . current = 7  rax = 5
8 6
CAS = no  . current = 7  rax = 7

[root@local cas]# ./cmpxchg
Please input like : 0 3
9 5
CAS = no  . current = 0  rax = 0
0 6
CAS = yes . current = 6  rax = 0
6 7
```

```
CAS = yes . current = 7  rax = 6
```

分析结果：
代码规则为，变量给定初始值 0，输入 2 个数字表示旧值、新值，执行 cas 操作，重复 3 次。
第一次运行。输入 0 5，执行 cas 成功，当前值为 5。输入 5 7，执行 cas 成功，当前值为 7。输入 8 6，执行 cas 失败，当前值为 7。
第二次运行。输入 9 5，执行 cas 失败，当前值为 0。输入 0 6，执行 cas 成功，当前值为 6。输入 6 7，执行 cas 成功，当前值为 7。

xchg 指令

编写代码：xchg.s

```
.data

int32_aa :      # 变量 aa
    .long 0x0

int32_bb :      # 变量 bb
    .long 0x0

str_tip :       # 提示
    .string "Please input like : 100  200 \n"

str_input :     # 输入
    .string "%d %d"

str_before :    # 执行前
    .string "Before xchg :  aa = %d    bb = %d \n"

str_after :     # 执行后
    .string "After  xchg :  aa = %d    bb = %d \n"

.text
.global main

main :
    pushq %rbp
    movq %rsp, %rbp

    # 提示
    movq $str_tip, %rdi
    callq printf

    # 输入
    movq $str_input, %rdi
    movq $int32_aa, %rsi
```

```
    movq $int32_bb, %rdx
    callq scanf

    callq func_xchg     # 执行交换函数

    popq %rbp
    retq

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

    movq $str_before, %rdi
    movl int32_aa(%rip),%esi
    movl int32_bb(%rip),%edx
    callq printf

    movl int32_aa(%rip), %eax   # 读 aa
    xchgl %eax, int32_bb(%rip)  # 交换 aa bb
    movl %eax, int32_aa(%rip)   # 写 aa

    movq $str_after, %rdi
    movl int32_aa(%rip),%esi
    movl int32_bb(%rip),%edx
    callq printf

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

编译代码：

```
gcc xchg.s -o xchg
```

运行代码：

```
[root@local cas]# ./xchg
Please input like : 100  200
22 33
Before xchg :  aa = 22    bb = 33
After  xchg :  aa = 33    bb = 22

[root@local cas]# ./xchg
Please input like : 100  200
-1 5
Before xchg :  aa = -1    bb = 5
After  xchg :  aa = 5    bb = -1
```

分析结果：

代码规则为，输入 2 个数字，交换数字。
第一次运行。输入 22 33，交换后，输出 aa = 33 bb = 22 。
第二次运行。输入-1 5，交换后，输出 aa = 5 bb = -1 。