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

指令说明

cmp 指令表示整数比较。
cmp 指令分为 8 位 cmpb、16 位 cmpw、32 位 cmpl、64 位 cmpq。
cmp 指令可以操作立即数、寄存器、内存。

语法格式 `cmp bb, aa` 表示 `result = aa - bb` 。

查看多种比较指令

编写代码： many_int.s

```
.data

int64 :
    .quad 333

int32 :
    .long 222

.text
.global main

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

    # 比较整数。64 位
    cmpq %rcx, %rdx          # 比较寄存器与寄存器
    cmpq %rcx, int64(%rip)   # 比较寄存器与变量
    cmpq int64(%rip), %rcx   # 比较寄存器与变量
    cmpq %rcx, -8(%rbp)      # 比较寄存器与栈内存
    cmpq -8(%rbp), %rcx      # 比较寄存器与栈内存
    cmpq $0x77, %rdx         # 比较立即数与寄存器
    cmpq $0x77, int64(%rip) # 比较立即数与变量

    # 比较整数。32 位
    cmpl %ecx, %edx         # 比较寄存器与寄存器
    cmpl %ecx, -4(%rbp)     # 比较寄存器与栈内存
    cmpl %ecx, int32(%rip) # 比较寄存器与变量
    cmpl $0x77, %edx        # 比较立即数与寄存器

    # 比较整数。16 位
    cmpw %cx, %dx          # 比较寄存器与寄存器
```

```
    cmpw %cx, -2(%rbp)    # 比较寄存器与栈内存
    cmpw %cx, int32(%rip) # 比较寄存器与变量
    cmpw $0x77, %dx       # 比较立即数与寄存器

    # 比较整数。8 位
    cmpb %cl, %dl         # 比较寄存器与寄存器
    cmpb %cl, -1(%rbp)    # 比较寄存器与栈内存
    cmpb %cl, int32(%rip) # 比较寄存器与变量
    cmpb $0x77, %dl       # 比较立即数与寄存器

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

编译代码：

```
gcc many_int.s -o many_int
```

64 位比较指令 cmpq。比如 `cmpq %rcx, int64(%rip)` 。
32 位比较指令 cmpl。比如 `cmpl %ecx, %edx` 。
16 位比较指令 cmpw。比如 `cmpw %cx, -2(%rbp)` 。
8 位比较指令 cmpb。比如 `cmpb $0x77, %dl` 。

有符号整数的比较

编写代码：signed_int.s

```
.data

int64_aa:
    .quad 0x0

int64_bb:
    .quad 0x0

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

str_big :   # 大于
    .string "Result :  %lld %#llX   >   %lld %#llX \n\n"

str_small :  # 小于
    .string "Result :  %lld %#llX   <   %lld %#llX \n\n"

str_equal :  # 等于
    .string "Result :  %lld %#llX   ==   %lld %#llX \n\n"

.text
```

```
.global main

main :
    pushq %rbp
    movq %rsp, %rbp

    # 输入 2 个变量
    movq $str_read, %rdi
    movq $int64_aa, %rsi        # 变量 aa
    movq $int64_bb, %rdx        # 变量 bb
    callq scanf

    # 比较 2 个变量
    movq int64_aa(%rip), %rax   # 变量 aa
    movq int64_bb(%rip), %rbx   # 变量 bb
    cmpq %rbx, %rax             # 比较  result = rax - rbx

    # 分支。大于、小于、等于
    jg tmp_big              # aa > bb
    jl tmp_small            # aa < bb
    je tmp_eq               # aa == bb

tmp_big :       # aa > bb
    movq $str_big, %rdi
    jmp tmp_out

tmp_small :     # aa < bb
    movq $str_small, %rdi
    jmp tmp_out

tmp_eq :        # aa == bb
    movq $str_equal, %rdi

tmp_out :   # 输出
    movq int64_aa(%rip), %rsi   # 变量 aa
    movq %rsi, %rdx             # 变量 aa
    movq int64_bb(%rip), %rcx   # 变量 bb
    movq %rcx, %r8              # 变量 bb
    callq printf

    popq %rbp
    retq
```

编译代码：

```
gcc signed_int.s -o signed_int
```

运行代码：

```
[root@local cmp]# ./signed_int
```

```
1 3
Result :  1 0X1   <   3 0X3

[root@local cmp]# ./signed_int
3 1
Result :  3 0X3   >   1 0X1

[root@local cmp]# ./signed_int
3 3
Result :  3 0X3   ==   3 0X3

[root@local cmp]# ./signed_int
-1 3
Result :  -1 0XFFFFFFFFFFFFFFFF   <   3 0X3

[root@local cmp]# ./signed_int
-1 -3
Result :  -1 0XFFFFFFFFFFFFFFFF   >   -3 0XFFFFFFFFFFFFFFFD
```

分析结果：
比较有符号整数。
输入 1 和 3，输出 1 小于 3。
输入 3 和 1，输出 3 大于 1。
输入 3 和 3，输出 3 等于 3。
输入-1 和 3，输出-1 小于 3。
输入-1 和-3，输出-1 大于-3。

无符号整数的比较

编写代码：unsigned_int.s

```
.data

int64_aa:
    .quad 0x0

int64_bb:
    .quad 0x0

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

str_big :   # 大于
    .string "Result :  %lld %#llX   >   %lld %#llX \n\n"

str_small :  # 小于
    .string "Result :  %lld %#llX   <   %lld %#llX \n\n"
```

```
str_equal :  # 等于
    .string "Result :  %lld %#llX   ==   %lld %#llX \n\n"

.text
.global main

main :
    pushq %rbp
    movq %rsp, %rbp

    # 输入 2 个变量
    movq $str_read, %rdi
    movq $int64_aa, %rsi        # 变量 aa
    movq $int64_bb, %rdx        # 变量 bb
    callq scanf

    # 比较
    movq int64_aa(%rip), %rax   # 变量 aa
    movq int64_bb(%rip), %rbx   # 变量 bb
    cmpq %rbx, %rax             # 比较 result = rax - rbx

    # 分支。大于、小于、等于
    ja tmp_big              # aa > bb
    jb tmp_small            # aa < bb
    je tmp_eq               # aa == bb

tmp_big :
    movq $str_big, %rdi
    jmp tmp_out

tmp_small :
    movq $str_small, %rdi
    jmp tmp_out

tmp_eq :
    movq $str_equal, %rdi

tmp_out :   # 输出
    movq int64_aa(%rip), %rsi   # 变量 aa
    movq %rsi, %rdx             # 变量 aa
    movq int64_bb(%rip), %rcx   # 变量 bb
    movq %rcx, %r8              # 变量 bb
    callq printf

    popq %rbp
    retq
```

编译代码：

```
gcc unsigned_int.s -o unsigned_int
```

运行代码：

```
[root@local cmp]# ./unsigned_int
1 3
Result :  1 0X1   <   3 0X3

[root@local cmp]# ./unsigned_int
3 1
Result :  3 0X3   >   1 0X1

[root@local cmp]# ./unsigned_int
3 3
Result :  3 0X3   ==   3 0X3

[root@local cmp]# ./unsigned_int
-1 3
Result :  -1 0XFFFFFFFFFFFFFFFF   >   3 0X3

[root@local cmp]# ./unsigned_int
-1 -3
Result :  -1 0XFFFFFFFFFFFFFFFF   >   -3 0XFFFFFFFFFFFFFFFD
```

分析结果：

比较无符号整数。

输入 1 和 3，输出 1 小于 3。

输入 3 和 1，输出 3 大于 1。

输入 3 和 3，输出 3 等于 3。

输入-1 和 3，输出-1 大于 3。-1 表示十六进制 0XFFFFFFFFFFFFFFFF，3 表示十六进制 0X3，0XFFFFFFFFFFFFFFFF 大于 0X3。

输入-1 和-3，输出-1 大于-3。-1 表示十六进制 0XFFFFFFFFFFFFFFFF，-3 表示十六进制 0XFFFFFFFFFFFFFFFD，0XFFFFFFFFFFFFFFFF 大于 0XFFFFFFFFFFFFFFFD。