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

# 指令说明

sub 指令表示整数减法。
sub 指令分为 8 位 subb、16 位 subw、32 位 subl、64 位 subq。
sub 指令可以操作立即数、寄存器、内存。

语法格式 `sub ee,ff` 表示 `ff = ff - ee` 。

# 简单的减法

编写代码： sub.s

```
.data

num_int64:
    .quad 0x0  # 64 位

str_int8:
    .string "  int8   value = %#X \n"

str_int16:
    .string "  int16  value = %#X \n"

str_int32:
    .string "  int32  value = %#X \n"

str_int64:
    .string "  int64  value = %lld \n"

.text
.global main

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

    # 64 位。操作寄存器
    movq $2000, %rcx
    subq $3, %rcx       # 应用于寄存器
    movq $str_int64, %rdi
    movq %rcx, %rsi
    callq printf
```

```
    # 64 位。操作栈
    movq $2000, -8(%rbp)
    subq $5, -8(%rbp)   # 应用于栈
    movq $str_int64, %rdi
    movq -8(%rbp) , %rsi
    callq printf

    # 64 位。操作数据段
    movq $2000, num_int64(%rip)
    subq $7, num_int64(%rip)    # 应用于数据段
    leaq str_int64(%rip), %rdi
    movq num_int64(%rip) , %rsi
    callq printf

    # 8 位。操作寄存器
    movb $0x33, %al     # 写 1 个字节
    subb $0x11, %al     # 减法，8 位
    movzbl %al, %esi    # 把 1 个字节扩展为 4 个字节
    leaq str_int8(%rip), %rdi
    callq printf

    # 16 位。操作寄存器
    movw $0x3333, %ax   # 写 2 个字节
    subw $0x2222, %ax   # 减法，16 位
    movzwl %ax, %esi    # 把 2 个字节扩展为 4 个字节
    leaq str_int16(%rip), %rdi
    callq printf

    # 32 位。操作寄存器
    movl $0x55555555, %esi # 写 4 个字节
    subl $0x2222, %esi     # 减法，32 位
    leaq str_int32(%rip), %rdi
    callq printf

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

编译代码：

```
gcc sub.s -o sub
```

运行代码：

```
[root@local int]# ./sub
  int64  value = 1997
  int64  value = 1995
  int64  value = 1993
  int8   value = 0X22
  int16  value = 0X1111
```

```
int32  value = 0X55553333
```

分析结果：

| 汇编代码 | 结果和分析 |
| --- | --- |
| # 64 位。操作寄存器<br>movq $2000, %rcx<br>subq $3, %rcx  # 应用于寄存器 | int64  value = 1997<br><br>把 64 位 2000 写到寄存器 rcx。<br>rcx 减去 3。<br>2000 - 3 = 1997 |
| # 64 位。操作栈<br>movq $2000, -8(%rbp)<br>subq $5, -8(%rbp)  # 应用于栈 | int64  value = 1995<br><br>把 64 位 2000 写到栈内存-8(%rbp)。<br>-8(%rbp)减去 5。<br>2000 - 5 = 1995 |
| # 64 位。操作数据段<br>movq $2000, num_int64(%rip)<br>subq $7, num_int64(%rip)  # 应用于数据段 | int64  value = 1993<br><br>把 64 位 2000 写到变量 num_int64。<br>num_int64 减去 7。<br>2000 - 7 = 1993 |
| # 8 位。操作寄存器<br>movb $0x33, %al  # 写 1 个字节<br>subb $0x11, %al  # 减法，8 位 | int8  value = 0X22<br><br>把 8 位 0x33 写到寄存器 al。<br>al 减去 0x11。<br>0x33 - 0x11 = 0X22 |
| # 16 位。操作寄存器<br>movw $0x3333, %ax  # 写 2 个字节<br>subw $0x2222, %ax  # 减法，16 位 | int16  value = 0X1111<br><br>把 16 位 0x3333 写到寄存器 ax 。<br>ax 减去 0x2222。<br>0x3333 - 0x2222 = 0X1111 |
| # 32 位。操作寄存器<br>movl $0x55555555, %esi # 写 4 个字节<br>subl $0x2222, %esi  # 减法，32 位 | int32  value = 0X55553333<br><br>把 32 位 0x55555555 写到寄存器 esi。<br>esi 减去 0x2222。<br>0x55555555 - 0x2222 = 0X55553333 |

# 复杂的减法

编写代码： sub_hard.s

```
.data

num_int64:
    .quad 0x0

str_int64:
    .string "subl  int64  value = %#llX \n"
```

```
str_flow_int32:
    .string "flow  int32  value = %#X \n"

str_flow_int64:
    .string "flow  int64  value = %#llX \n"

.text
.global main

main :
    pushq %rbp
    movq %rsp, %rbp

    # 64 位变量，执行 32 位减法。操作数据段
    movq $0x3333333377777777, %r9
    movq %r9, num_int64(%rip)
    subl $0x11111111, num_int64(%rip)
    movq num_int64(%rip), %rsi
    leaq str_int64(%rip), %rdi
    callq printf

    # 32 位变量，32 位溢出。操作寄存器
    movl $0x00000001, %edx
    subl $0x2, %edx
    leaq str_flow_int32(%rip), %rdi
    movl %edx, %esi
    callq printf

    # 64 位变量，32 位溢出。操作寄存器
    movq $0x5555555500000001, %rbx
    subl $0x2, %ebx    # 导致高位清零
    leaq str_flow_int64(%rip), %rdi
    movq %rbx, %rsi
    callq printf

    # 64 位变量，16 位溢出。操作寄存器
    movq $0x5555555500000001, %rbx
    subw $0x2, %bx
    leaq str_flow_int64(%rip), %rdi
    movq %rbx, %rsi
    callq printf

    # 64 位变量，8 位溢出。操作寄存器
    movq $0x5555555500000001, %rbx
    subb $0x2, %bh
    leaq str_flow_int64(%rip), %rdi
    movq %rbx, %rsi
    callq printf
```

```
    # 64 位变量，8 位溢出。操作寄存器
    movq $0x5555555500000001, %rbx
    subb $0x2, %bl
    leaq str_flow_int64(%rip), %rdi
    movq %rbx, %rsi
    callq printf

    # 64 位变量，32 位溢出。操作数据段
    movq $0x5555555500000001, %r9
    movq %r9, num_int64(%rip)
    subl $0x2, num_int64(%rip)
    leaq str_flow_int64(%rip), %rdi
    movq num_int64(%rip), %rsi
    callq printf

    popq %rbp
    retq
```

编译代码：

```
gcc sub_hard.s -o sub_hard
```

运行代码：

```
[root@local int]# ./sub_hard
subl  int64  value = 0X3333333366666666
flow  int32  value = 0XFFFFFFFF
flow  int64  value = 0XFFFFFFFF
flow  int64  value = 0X555555550000FFFF
flow  int64  value = 0X555555550000FE01
flow  int64  value = 0X55555555000000FF
flow  int64  value = 0X55555555FFFFFFFF
```

分析结果：

| 汇编代码 | 输出结果 |
| --- | --- |
| # 64 位变量，执行 32 位减法。操作数据段<br>movq $0x3333333377777777, %r9<br>movq %r9, num_int64(%rip)<br>subl $0x11111111, num_int64(%rip) | subl  int64  value = 0X3333333366666666<br><br>把 64 位 0x3333333377777777 写到变量 num_int64。<br>num_int64 减去 32 位 0x11111111。<br>结果为 0X3333333366666666。<br>低 32 位受影响，从 77777777 变为 66666666。<br>其他位不受影响。 |
| # 32 位变量，32 位溢出。操作寄存器<br>movl $0x00000001, %edx<br>subl $0x2, %edx | flow  int32  value = 0XFFFFFFFF<br><br>把 32 位 0x00000001 写到 edx。<br>edx 减去 32 位 0x2。<br>结果为 0XFFFFFFFF。溢出。 |
| # 64 位变量，32 位溢出。操作寄存器<br>movq $0x5555555500000001, %rbx<br>subl $0x2, %ebx  # 导致高位清零 | flow  int64  value = 0XFFFFFFFF<br><br>把 64 位 0x5555555500000001 写到寄存器 rbx。<br>ebx 减去 32 位 0x2。 |

| | |
|---|---|
| | 结果为 0XFFFFFFFF。<br>触发高 32 位清零。 |
| # 64 位变量，16 位溢出。操作寄存器<br>movq $0x5555555500000001, %rbx<br>subw $0x2, %bx | flow int64 value = 0X555555550000FFFF<br><br>把 64 位 0x5555555500000001 写到寄存器 rbx。<br>bx 减去 16 位 0x2。<br>结果为 0X555555550000FFFF。<br>低 16 位受影响，从 0001 变为 FFFF。<br>其他位不受影响。 |
| # 64 位变量，8 位溢出。操作寄存器<br>movq $0x5555555500000001, %rbx<br>subb $0x2, %bh | flow int64 value = 0X555555550000FE01<br><br>把 64 位 0x5555555500000001 写到寄存器 rbx。<br>bh 减去 8 位 0x2。<br>结果为 0X555555550000FE01。<br>高 8 位位受影响，从 00 变为 FE。<br>其他位不受影响。 |
| # 64 位变量，8 位溢出。操作寄存器<br>movq $0x5555555500000001, %rbx<br>subb $0x2, %bl | flow int64 value = 0X55555555000000FF<br><br>把 64 位 0x5555555500000001 写到寄存器 rbx。<br>bl 减去 8 位 0x2。<br>结果为 0X55555555000000FF。<br>低 8 位位受影响，从 01 变为 FF。<br>其他位不受影响。 |
| # 64 位变量，32 位溢出。操作数据段<br>movq $0x5555555500000001, %r9<br>movq %r9, num_int64(%rip)<br>subl $0x2, num_int64(%rip) | flow int64 value = 0X55555555FFFFFFFF<br><br>把 64 位 0x5555555500000001 写到变量 num_int64。<br>num_int64 减去 32 位 0x2。<br>结果为 0X55555555FFFFFFFF。<br>结果溢出。<br>低 32 位受影响，从 00000001 变为 FFFFFFFF。<br>其他位不受影响。 |