

指令说明

dec 指令表示自减。

dec 指令分为 8 位 decb、16 位 decw、32 位 decl、64 位 decq。

dec 指令可以操作寄存器、内存。

语法格式 `dec ee` 表示 `ee = ee - 1`。

用汇编代码分析

编写代码：dec.s

```
.data

int64 : .quad 7000
int32 : .long 6000

str64 : .string " int64 = %lld \n"
str64X : .string " int64 = %#llx \n"
str32 : .string " int32 = %d \n"

.text
.global main

main :
    pushq %rbp
    movq %rsp, %rbp

    # 64 位，操作寄存器
    movq $8000, %rbx
    decq %rbx          # 64 位。
    movq $str64, %rdi
    movq %rbx, %rsi
    callq printf

    # 32 位，操作寄存器
    movq $0x2222222255555555, %rbx # 64 位。占位
    decl %ebx           # 32 位。触发高位清 0
    movq $str64X, %rdi
    movq %rbx, %rsi
    callq printf

    # 16 位，操作寄存器
    movq $0x2222222200000000, %rbx # 64 位。占位
```

```
decw %bx          # 16 位
movq $str64X, %rdi
movq %rbx, %rsi
callq printf

# 8 位，操作寄存器
movq $0x2222222200000000, %rbx # 64 位。占位
decb %bl          # 8 位
movq $str64X, %rdi
movq %rbx, %rsi
callq printf

# 64 位，操作内存
decq int64(%rip)   # 64 位
movq $str64, %rdi
movq int64(%rip), %rsi
callq printf

# 32 位，操作内存
decl int32(%rip)   # 32 位
movq $str32, %rdi
movl int32(%rip), %esi
callq printf

popq %rbp
retq
```

编译代码：

```
gcc dec.s -o dec
```

运行代码：

```
[root@local self]# ./dec
int64 = 7999
int64 = 0X55555554
int64 = 0X222222220000FFFF
int64 = 0X22222222000000FF
int64 = 6999
int32 = 5999
```

分析结果：

汇编代码	结果和分析
# 64 位，操作寄存器 movq \$8000, %rbx decq %rbx # 64 位。 movq \$str64, %rdi movq %rbx, %rsi callq printf	int64 = 7999 把 64 位 8000 写到 rbx。 64 位 rbx 自减。 结果为 7999。
# 32 位，操作寄存器	int64 = 0X55555554

<pre>movq \$0x2222222255555555, %rbx # 64 位。占位 decl %ebx # 32 位。触发高位清 0 movq \$str64X, %rdi movq %rbx, %rsi callq printf</pre>	<p>把 64 位 0x2222222255555555 写到 rbx。</p> <p>32 位 ebx 自减。</p> <p>触发高位清零，结果为 0X55555554。</p>
<pre># 16 位，操作寄存器 movq \$0x2222222200000000, %rbx # 64 位。占位 decw %bx # 16 位 movq \$str64X, %rdi movq %rbx, %rsi callq printf</pre>	<pre>int64 = 0X222222220000FFFF</pre> <p>把 64 位 0x2222222200000000 写到 rbx。</p> <p>16 位 bx 自减。</p> <p>结果为 0X222222220000FFFF。</p> <p>低 16 位变化，从 0000 变为 FFFF。</p> <p>其他位不变，都为 222222220000。</p>
<pre># 8 位，操作寄存器 movq \$0x2222222200000000, %rbx # 64 位。占位 decb %bl # 8 位 movq \$str64X, %rdi movq %rbx, %rsi callq printf</pre>	<pre>int64 = 0X22222222000000FF</pre> <p>把 64 位 0x2222222200000000 写到 rbx。</p> <p>8 位 bl 自减。</p> <p>结果为 0X22222222000000FF。</p> <p>低 8 位变化，从 00 变为 FF。</p> <p>其他位不变，都为 2222222200000000。</p>
<pre># 64 位，操作内存 decq int64(%rip) # 64 位 movq \$str64, %rdi movq int64(%rip), %rsi callq printf</pre>	<pre>int64 = 6999</pre> <p>64 位 int64 的初始值为 7000。</p> <p>64 位 int64 自减。</p> <p>结果为 6999。</p>
<pre># 32 位，操作内存 decl int32(%rip) # 32 位 movq \$str32, %rdi movl int32(%rip), %esi callq printf</pre>	<pre>int32 = 5999</pre> <p>32 位 int32 的初始值为 6000。</p> <p>32 位 int32 自减。</p> <p>结果为 5999。</p>