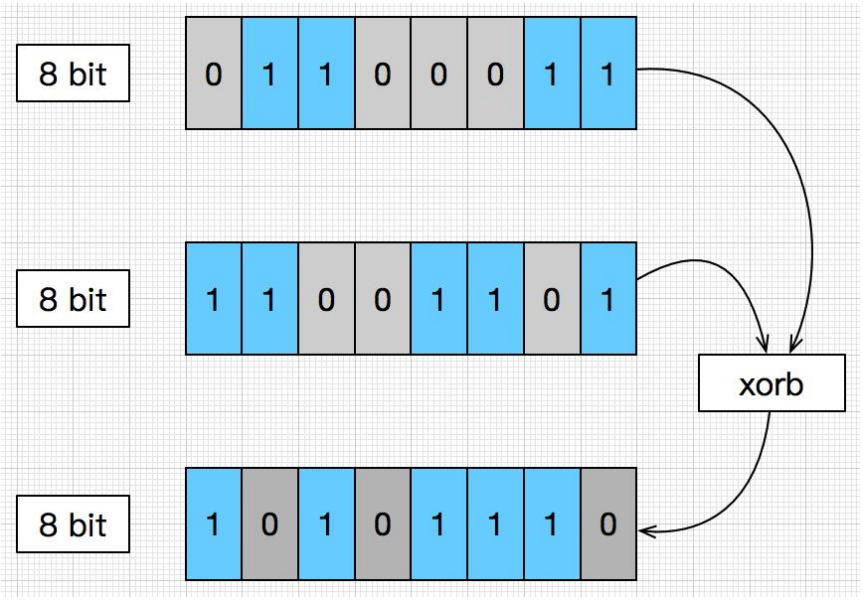
作者：代兴　邮箱：503268771@qq.com　Vx 公众号：东方架构师

指令说明

xor 指令表示位异或。
xor 指令分为 8 位 xorb、16 位 xorw、32 位 xorl、64 位 xorq。
xor 指令可以操作立即数、寄存器、内存。

语法格式 `xor bb, aa` 表示 `aa = aa ^ bb` 。
对应的位执行位异或操作。如果 2 个位相反，则结果为 1，否则结果为 0。



用汇编代码分析

编写代码：xor_bit.s

```
.data

str64 :
    .string " int64  %#llX \n"

str32 :
    .string " int32  %#X \n"

.text
.global main

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

```
# 64 位。操作寄存器
movq $0x1010101010101010, %rax  # 64 位，寄存器
movq $0x0101010101010101, %rbx  # 64 位，寄存器
xorq %rbx, %rax                 # 64 位，位异或
movq $str64, %rdi
movq %rax, %rsi
callq printf

# 64 位。操作寄存器
movq $0x55667788AABBCCDD, %rbx  # 64 位，寄存器
xorq %rbx, %rbx                 # 64 位，位异或
movq $str64, %rdi
movq %rbx, %rsi
callq printf

# 32 位。操作寄存器
movl $0xAABBCCDD, %ecx  # 32 位，寄存器
movl $0x00BBFF00, %edx  # 32 位，寄存器
xorl %edx, %ecx         # 32 位，位异或
movq $str32, %rdi
movl %ecx, %esi
callq printf

# 32 位。操作栈内存
movl $0xAABBCCDD, -8(%rbp)  # 32 位，栈
xorl $0x00BBFF00, -8(%rbp)  # 32 位，位异或
movq $str32, %rdi
movl -8(%rbp) , %esi
callq printf

# 16 位。操作寄存器
movl $0xAAAAAAAA, %ebx  # 32 位，占位
movw $0xCCCC, %bx       # 16 位，寄存器
xorw $0xFF00, %bx       # 16 位，位异或
movq $str32, %rdi
movl %ebx, %esi
callq printf

# 8 位。操作寄存器
movl $0xAAAAAAAA, %ebx  # 32 位，占位
movb $0xCC, %bl         # 8 位，寄存器
xorb $0xFF, %bl         # 8 位，位异或
movq $str32, %rdi
movl %ebx, %esi
callq printf

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

```
    retq
```

编译代码：

```
gcc xor_bit.s -o xor_bit
```

运行代码：

```
[root@local binary]# ./xor_bit
 int64  0X1111111111111111
 int64  0
 int32  0XAA0033DD
 int32  0XAA0033DD
 int32  0XAAAA33CC
 int32  0XAAAAAA33
```

分析结果：

汇编代码	结果和分析

# 64 位。操作寄存器 	
movq $0x1010101010101010, %rax # 64 位，寄存器
movq $0x0101010101010101, %rbx # 64 位，寄存器
xorq %rbx, %rax # 64 位，位异或
	int64 0X1111111111111111
	
	操作 64 位寄存器 rbx、rax。
	0x1010101010101010 ^ 0x0101010101010101 = 0X1111111111111111

# 64 位。操作寄存器 movq $0x55667788AABBCCDD, %rbx # 64 位，寄存器 xorq %rbx, %rbx # 64 位，位异或	int64 0 操作 64 位寄存器 rbx。 0x55667788AABBCCDD ^ 0x55667788AABBCCDD = 0

# 32 位。操作寄存器 	
movl $0xAABBCCDD, %ecx # 32 位，寄存器
movl $0x00BBFF00, %edx # 32 位，寄存器
xorl %edx, %ecx # 32 位，位异或
	int32 0XAA0033DD
	
	操作 32 位寄存器 edx、ecx。
	0xAABBCCDD ^ 0x00BBFF00 = 0XAA0033DD

# 32 位。操作栈内存 movl $0xAABBCCDD, -8(%rbp) # 32 位，栈 xorl $0x00BBFF00, -8(%rbp) # 32 位，位异或	int32 0XAA0033DD 操作栈内存。 0xAABBCCDD ^ 0x00BBFF00 = 0XAA0033DD

# 16 位。操作寄存器 	
movl $0xAAAAAAAA, %ebx # 32 位，占位
movw $0xCCCC, %bx # 16 位，寄存器
xorw $0xFF00, %bx # 16 位，位异或
	int32 0XAAAA33CC
	
	操作 16 位寄存器 bx。
	高 16 位，没有变化，都为 AAAA。
	低 16 位，发生变化，从 CCCC 变成 33CC。
	CCCC ^ FF00 = 33CC

# 8 位。操作寄存器 	
movl $0xAAAAAAAA, %ebx # 32 位，占位
movb $0xCC, %bl # 8 位，寄存器
xorb $0xFF, %bl # 8 位，位异或
	int32 0XAAAAAA33
	
	操作 8 位寄存器 bl。
	高 24 位，没有变化，都为 AAAAAA。
	低 8 位，发生变化，从 CC 变成 33。
	CC ^ FF = 33

二进制 0 与 0 执行异或，等于 0。
二进制 1 与 1 执行异或，等于 0。
某数与自身异或等于 0，可以用于快速清零。比如，`xorq %rbx, %rbx`。

十六进制 F 等于二进制 1111。
十六进制某数异或 F，等价于 F 减去该数。
十六进制 C 异或 F，等价于 `F - C = 3` 。

十六进制 0 等于二进制 0000。
十六进制某数异或 0，等价于该数。
十六进制 C 异或 0，等价于自身 C。