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

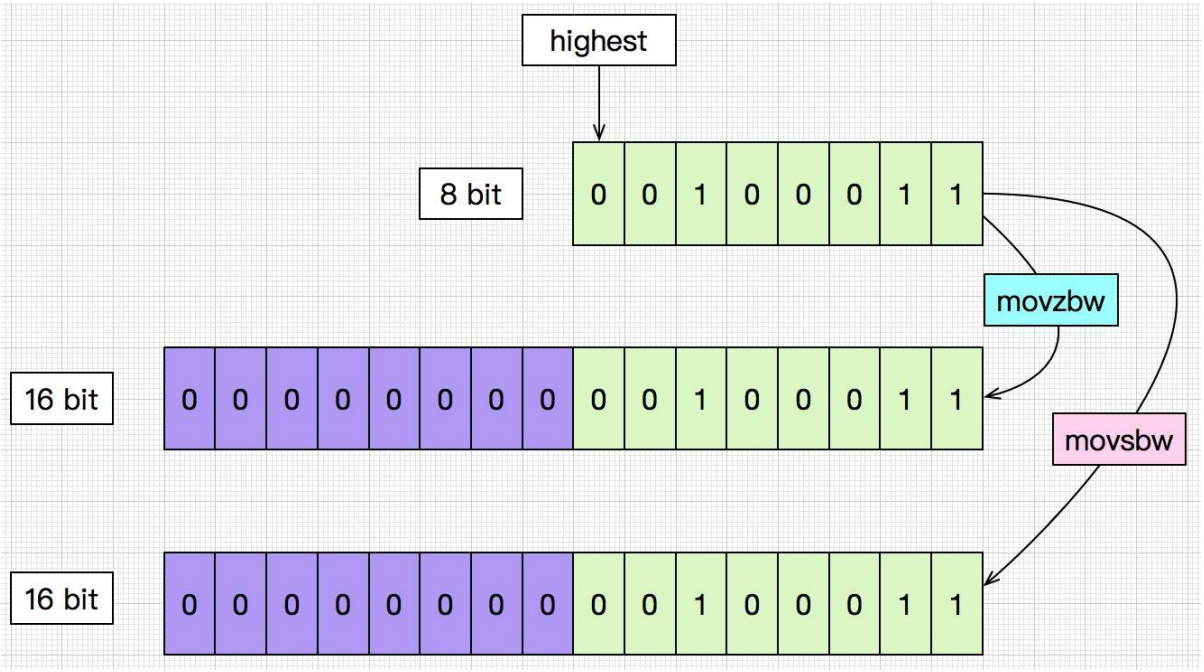
低位转高位，使用零扩展、符号扩展。
高位转低位，使用 mov 指令截取。

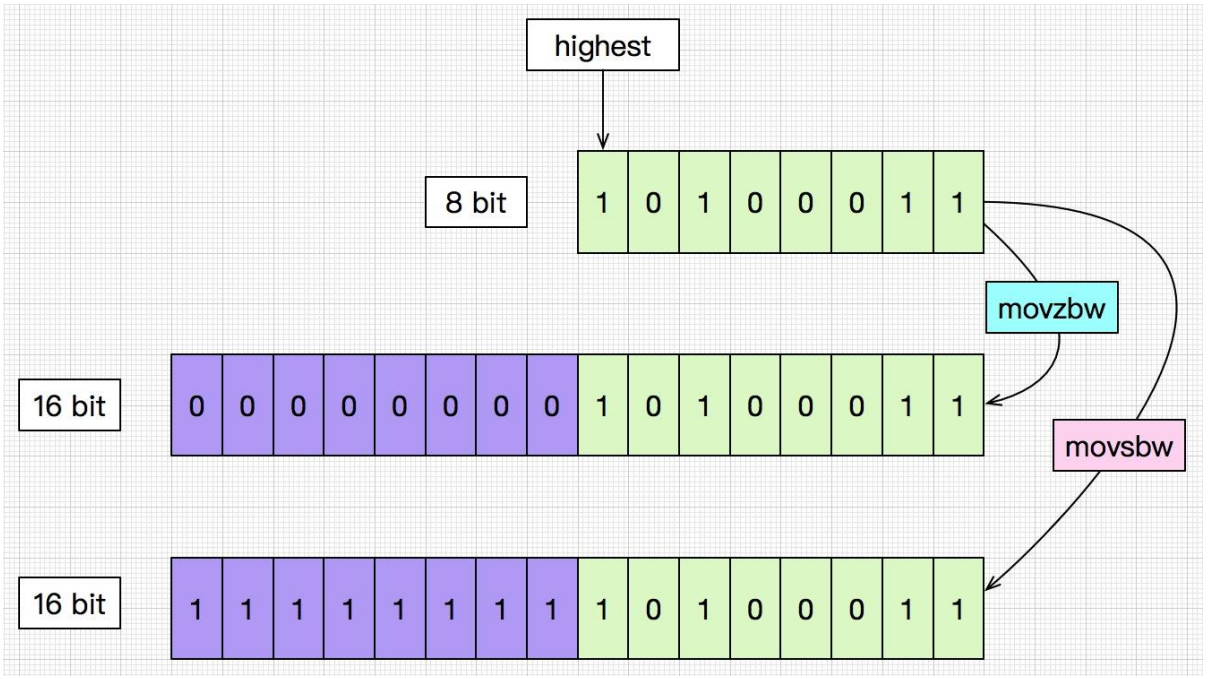
低位转高位

零扩展，把 0 扩展到高位，使用 movzX 指令。X 表示对应的位数。
movzbw 指令，把 8 位零扩展为 16 位。比如，`movzbw %al, %bx` 。
movzbl 指令，把 8 位零扩展为 32 位。比如，`movzbl %al, %eax` 。
movzbq 指令，把 8 位零扩展为 64 位。比如，`movzbq %al, %rax` 。
movzwl 指令，把 16 位零扩展为 32 位。比如，`movzwl %bx, %eax` 。
movzwq 指令，把 16 位零扩展为 64 位。比如，`movzwq %bx, %rax` 。
movl 指令，把 32 位零扩展为 64 位。比如，`movl %eax, %ebx` 。

符号扩展，把符号位扩展到高位，使用 movsX 指令。X 表示对应的位数。
movsbw 指令，把 8 位符号扩展为 16 位。比如，`movsbw %al, %bx` 。
movsbl 指令，把 8 位符号扩展为 32 位。比如，`movsbl %al, %eax` 。
movsbq 指令，把 8 位符号扩展为 64 位。比如，`movsbq %al, %rax` 。
movswl 指令，把 16 位符号扩展为 32 位。比如，`movswl %bx, %eax` 。
movswq 指令，把 16 位符号扩展为 64 位。比如，`movswq %bx, %rax` 。
movslq 指令，把 32 位符号扩展为 64 位。比如，`movslq %eax, %rax` 。

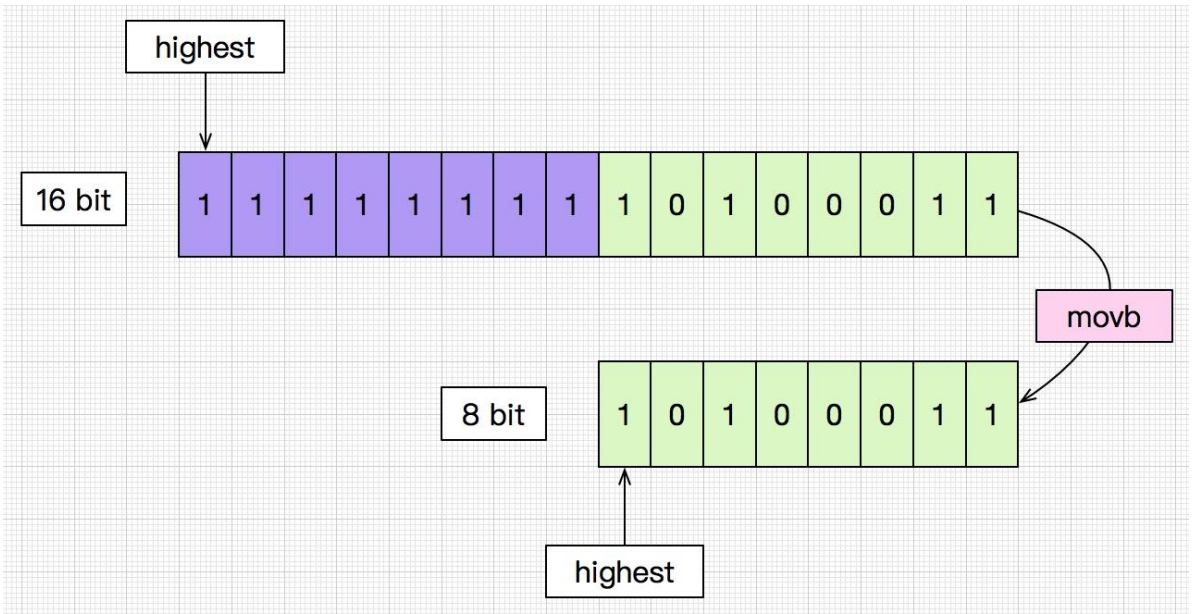
如果某个数的最高位是 0，则零扩展、符号扩展都把 0 扩展到高位，两者结果相同。
如果某个数的最高位是 1，则零扩展把 0 扩展到高位，符号扩展把 1 扩展到高位，两者结果不相同。





高位转低位

高位转低位，使用 mov 指令截取。对于有符号数、无符号数，都适用。
如果高位数据大，转低位可能导致数据丢失。比如 16 位 0xA2B3 转 8 位 0xB3。



用汇编代码分析

编写代码： int_cvt.s

```
.data
```

```
str_int32 :   # 查看 16 进制
    .string " int32   movsX = %#X   movzX = %#X  \n"

str_int64 :   # 查看 16 进制
    .string " int64   movsX = %#llX   movzX = %#llX \n"

str_64to16 :  # 64 位转 16 位
    .string " to_small  int64 = %lld %#llX   int16 = %d %#X \n"

str_64to32 :  # 64 位转 32 位
    .string " to_small  int64 = %lld %#llX   int32 = %d %#X \n"

.text
.global main

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

    # 8 位转 32 位。操作寄存器
    movb $0x41, %cl     # 8 位。最高位是 0
    movsbl %cl, %eax    # 符号扩展
    movzbl %cl, %ebx    # 零扩展
    movq $str_int32, %rdi
    movl %eax, %esi
    movl %ebx, %edx
    callq printf

    # 8 位转 32 位。操作寄存器
    movb $0x81, %cl     # 8 位。最高位是 1
    movsbl %cl, %eax    # 符号扩展
    movzbl %cl, %ebx    # 零扩展
    movq $str_int32, %rdi
    movl %eax, %esi
    movl %ebx, %edx
    callq printf

    # 16 位转 64 位。操作寄存器
    movw $0x8221, %cx   # 16 位。最高位是 1
    movswq %cx, %rax    # 符号扩展
    movzwq %cx, %rbx    # 零扩展
    movq $str_int64, %rdi
    movq %rax, %rsi
    movq %rbx, %rdx
    callq printf

   # 16 位转 64 位。操作内存
    movw $0x8331, -2(%rbp)  # 16 位。最高位是 1
```

```
    movswq -2(%rbp), %r8    # 符号扩展
    movzwq -2(%rbp), %r9    # 零扩展
    movq $str_int64, %rdi
    movq %r8, %rsi
    movq %r9, %rdx
    callq printf

    # 32 位转 64 位。操作寄存器
    movl $0x81111551, %ecx  # 32 位。最高位是 1
    movslq %ecx, %rax       # 符号扩展
    movl %ecx, %ebx         # 零扩展
    movq $str_int64, %rdi
    movq %rax, %rsi
    movq %rbx, %rdx
    callq printf

    # 64 位转 16 位。操作寄存器
    movq $-99, %rax     # 64 位。最高位是 1
    movw %ax, %bx       # 截取 16 位
    movq $str_64to16, %rdi
    movq %rax, %rsi
    movq %rax, %rdx
    movswl %bx, %ecx
    movl %ecx, %r8d
    callq printf

    # 64 位转 32 位。操作内存
    movq $77, -32(%rbp)      # 64 位。最高位是 0
    movl -32(%rbp), %eax     # 截取 32 位
    movq $str_64to32, %rdi
    movq -32(%rbp), %rsi
    movq %rsi, %rdx
    movl %eax, %ecx
    movl %ecx, %r8d
    callq printf

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

编译代码：

```
gcc int_cvt.s -o int_cvt
```

运行代码：

```
[root@local cvt]# ./int_cvt
 int32   movsX = 0X41   movzX = 0X41
 int32   movsX = 0XFFFFFF81   movzX = 0X81
 int64   movsX = 0XFFFFFFFFFFFF8221   movzX = 0X8221
 int64   movsX = 0XFFFFFFFFFFFF8331   movzX = 0X8331
```

```
int64   movsX = 0XFFFFFFFF81111551   movzX = 0X81111551
to_small  int64 = -99 0XFFFFFFFFFFFFFF9D   int16 = -99 0XFFFFFF9D
to_small  int64 = 77 0X4D   int32 = 77 0X4D
```

分析结果：

汇编代码	结果和分析
# 8 位转 32 位。操作寄存器 movb $0x41, %cl # 8 位。最高位是 0 movsbl %cl, %eax # 符号扩展 movzbl %cl, %ebx # 零扩展	int32 movsX = 0X41 movzX = 0X41 8 位数字 0x41 的最高位是 0。 符号扩展，高位扩展为 0，结果为 0X41。 零扩展，高位扩展为 0，结果为 0X41。
# 8 位转 32 位。操作寄存器 movb $0x81, %cl # 8 位。最高位是 1 movsbl %cl, %eax # 符号扩展 movzbl %cl, %ebx # 零扩展	int32 movsX = 0XFFFFFF81 movzX = 0X81 8 位数字 0x81 的最高位是 1。 符号扩展，高位扩展为 1，结果为 0XFFFFFF81。 零扩展，高位扩展为 0，结果为 0X81。
# 16 位转 64 位。操作寄存器 movw $0x8221, %cx # 16 位。最高位是 1 movswq %cx, %rax # 符号扩展 movzwq %cx, %rbx # 零扩展	int64 movsX = 0XFFFFFFFFFFFF8221 movzX = 0X8221 16 位数字 0x8221 的最高位是 1。 符号扩展，高位扩展为 1，结果为 0XFFFFFFFFFFFF8221。 零扩展，高位扩展为 0，结果为 0X8221。
# 16 位转 64 位。操作内存 movw $0x8331, -2(%rbp) # 16 位。最高位是 1 movswq -2(%rbp), %r8 # 符号扩展 movzwq -2(%rbp), %r9 # 零扩展	int64 movsX = 0XFFFFFFFFFFFF8331 movzX = 0X8331 16 位数字 0x8331 的最高位是 1。 符号扩展，高位扩展为 1，结果为 0XFFFFFFFFFFFF8331。 零扩展，高位扩展为 0，结果为 0X8331。

# 32 位转 64 位。操作寄存器 	
movl $0x81111551, %ecx # 32 位。最高位是 1
movslq %ecx, %rax # 符号扩展
movl %ecx, %ebx # 零扩展
	int64 movsX = 0XFFFFFFFF81111551 movzX = 0X81111551
	
	32 位数字 0x81111551 的最高位是 1。
	符号扩展，高位扩展为 1，结果为 0XFFFFFFFF81111551。
	零扩展，高位扩展为 0，结果为 0X81111551。

# 64 位转 16 位。操作寄存器 	
movq $-99, %rax # 64 位。最高位是 1
movw %ax, %bx # 截取 16 位
	to_small int64 = -99 0XFFFFFFFFFFFFFF9D int16 = -99 0XFFFFFF9D
	
	64 位数字-99 的最高位是 1。
	高位包含很多 1，表示为 0XFFFFFFFFFFFFFF9D。
	截取 32 位，表示为 0XFFFFFF9D。

# 64 位转 32 位。操作内存 	
movq $77, -32(%rbp) # 64 位。最高位是 0
movl -32(%rbp), %eax # 截取 32 位
	to_small int64 = 77 0X4D int32 = 77 0X4D
	
	64 位数字 77 的最高位是 0。
	高位包含很多 0，表示为 0X4D。
	截取 32 位，表示为 0X4D。