The Error That Shouldn’t Be

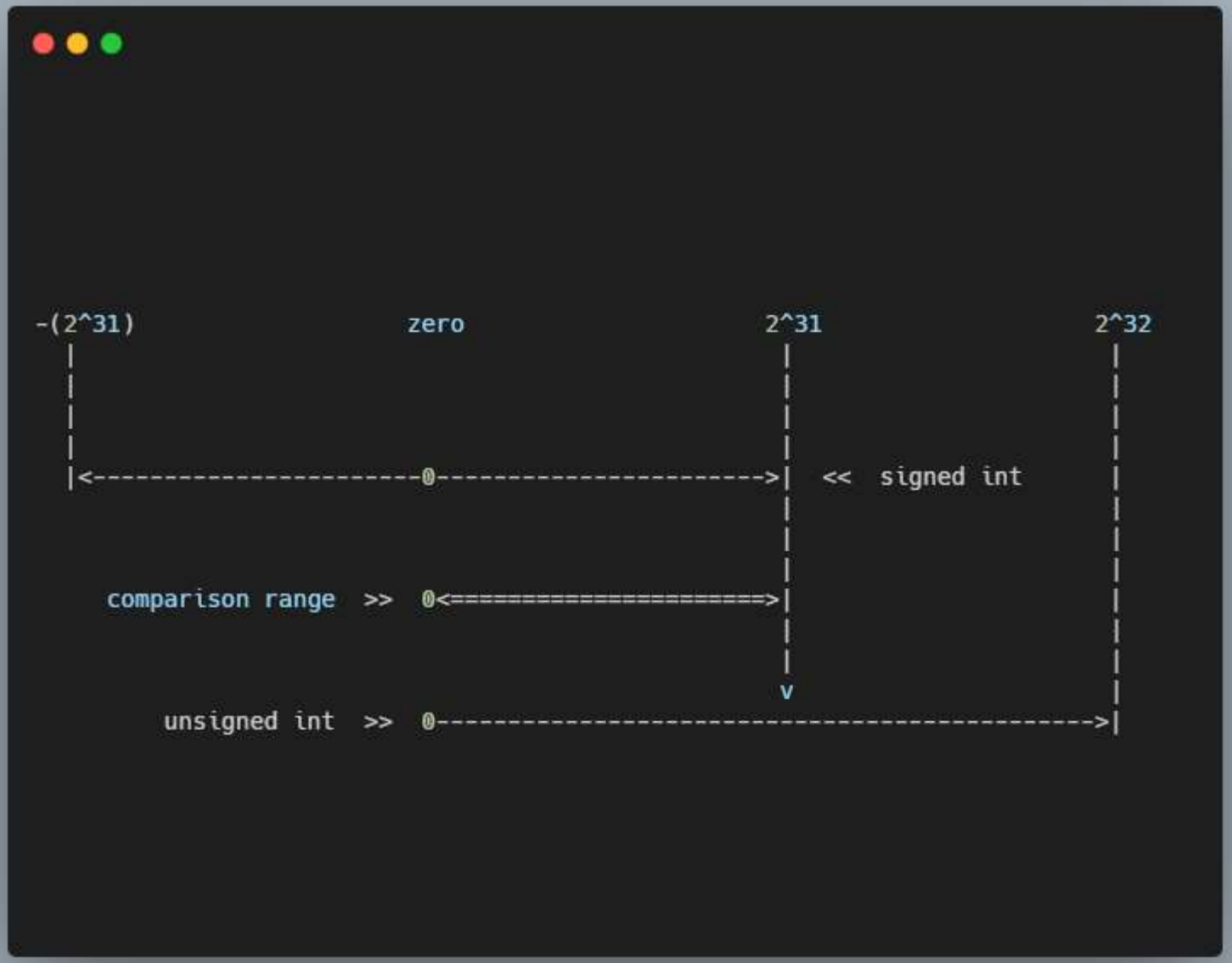
Chris Ryan

Northwest C++ User Group

September 15th, 2021


void foo(int a, uint b)
{
if (a < b)
{
...
...
...
}
else if (a > b)
{
...
...
...
}
else if (a == b)
{
...
...
...
}
else
{
...
...
...
}
}

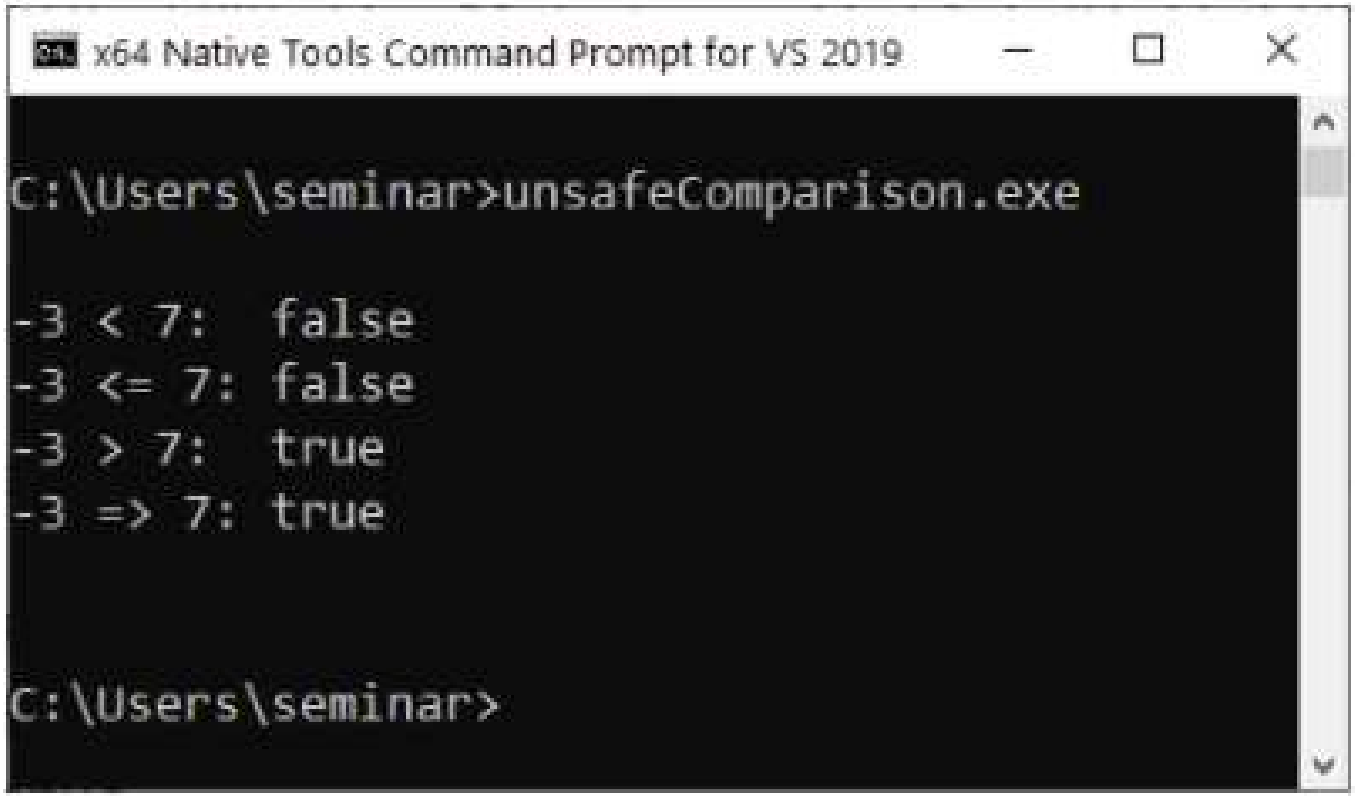
```
void foo(int a, uint b)
{
    if (a < b)                  <<< signed / unsigned mismatch
    {
        ...
        ...
        ...
    }
    else if (a > b)             <<< signed / unsigned mismatch
    {
        ...
        ...
        ...
    }
    else if (a == b)            <<< signed / unsigned mismatch
    {
        ...
        ...
        ...
    }
    else
    {
        ...
        ...
        ...
    }
}
```


-(2^31)
zero
2^31
2^32
<< signed int
comparison range >>
unsigned int >>
0

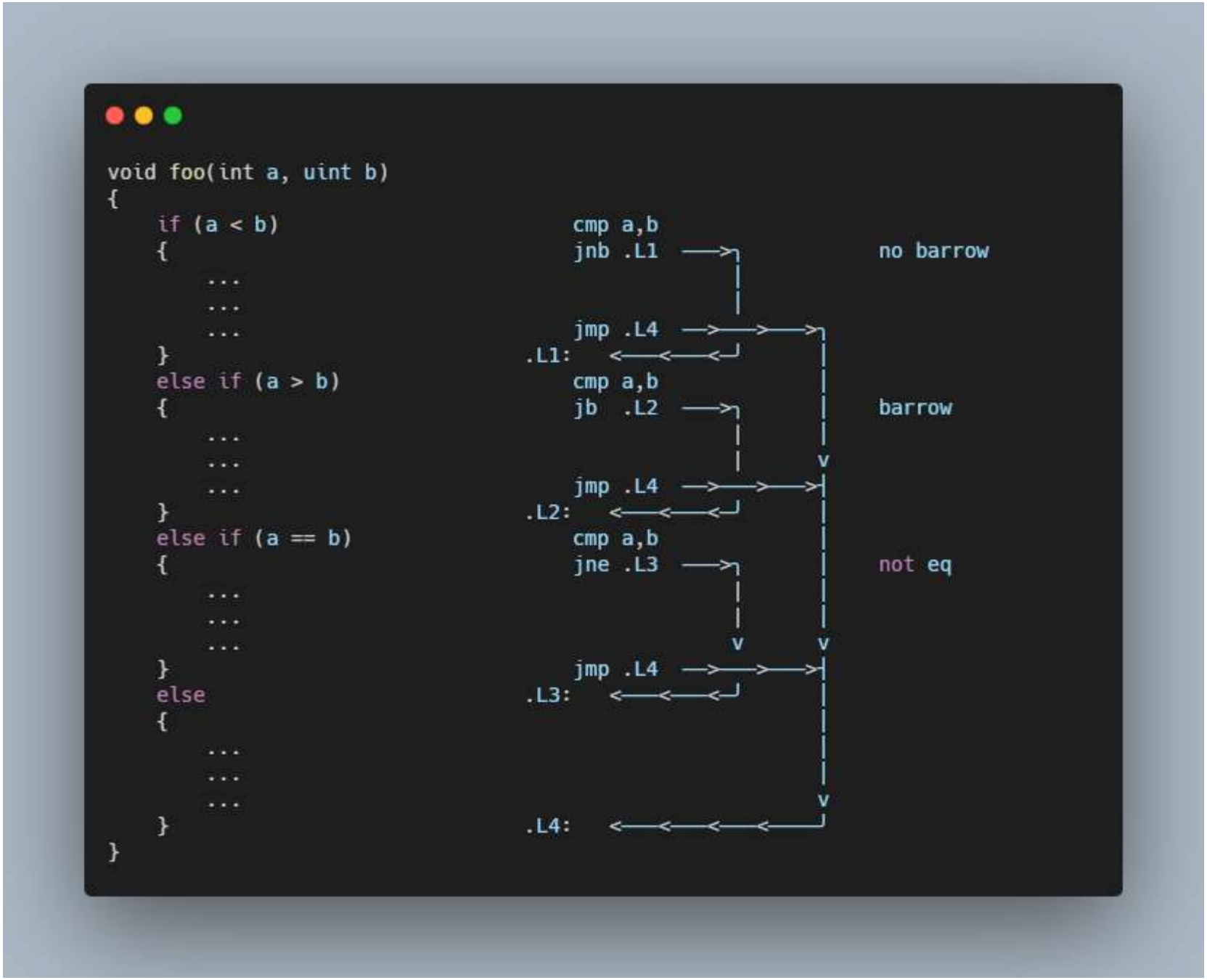
Rainer Grimm: modernescpp.com

```cpp
// unsafeComparison.cpp

#include <iostream>

int main() {

    std::cout << std::endl;

    std::cout << std::boolalpha;

    int x = -3;                 // (1)
    unsigned int y = 7;         // (2)

    std::cout << "-3 < 7:  " << (x < y) << std::endl;
    std::cout << "-3 <= 7: " << (x <= y) << std::endl;
    std::cout << "-3 > 7:  " << (x > y) << std::endl;
    std::cout << "-3 => 7: " << (x >= y) << std::endl;

    std::cout << std::endl;

}
```

https://www.modernescpp.com/index.php/safe-comparisons-of-integrals-with-c-20


x64 Native Tools Command Prompt for VS 2019
C:\Users\seminar>unsafeComparison.exe
-3 < 7: false
-3 <= 7: false
-3 > 7: true
-3 => 7: true
C:\Users\seminar>

```cpp
// safeComparison.cpp

#include <iostream>
#include <utility>

int main() {

    std::cout << std::endl;

    std::cout << std::boolalpha;

    int x = -3;
    unsigned int y = 7;

    std::cout << "3 == 7:  " << std::cmp_equal(x, y) << std::endl;
    std::cout << "3 != 7:  " << std::cmp_not_equal(x, y) << std::endl;
    std::cout << "-3 < 7:  " << std::cmp_less(x, y) << std::endl;
    std::cout << "-3 <= 7: " << std::cmp_less_equal(x, y) << std::endl;
    std::cout << "-3 > 7:  " << std::cmp_greater(x, y) << std::endl;
    std::cout << "-3 => 7: " << std::cmp_greater_equal(x, y) << std::endl;

    std::cout << std::endl;

}
```


void foo(int a, uint b)
{
if (a < b)
{
...
...
...
}
else if (a > b)
{
...
...
...
}
else if (a == b)
{
...
...
...
}
else
{
...
...
...
}
}
cmp a,b
jnb .L1
no barrow
jmp .L4
.L1:
cmp a,b
jb .L2
barrow
jmp .L4
.L2:
cmp a,b
jne .L3
not eq
jmp .L4
.L3:
.L4:

Processor Flags

The x86 processors have a large set of flags that represent the state of the processor, and the conditional jump instructions can key off of them in combination.

- **CF - carry flag**

  Set on high-order bit carry or borrow; cleared otherwise

- **PF - parity flag**

  Set if low-order eight bits of result contain an even number of "1" bits; cleared otherwise

- **ZF - zero flags**

  Set if result is zero; cleared otherwise

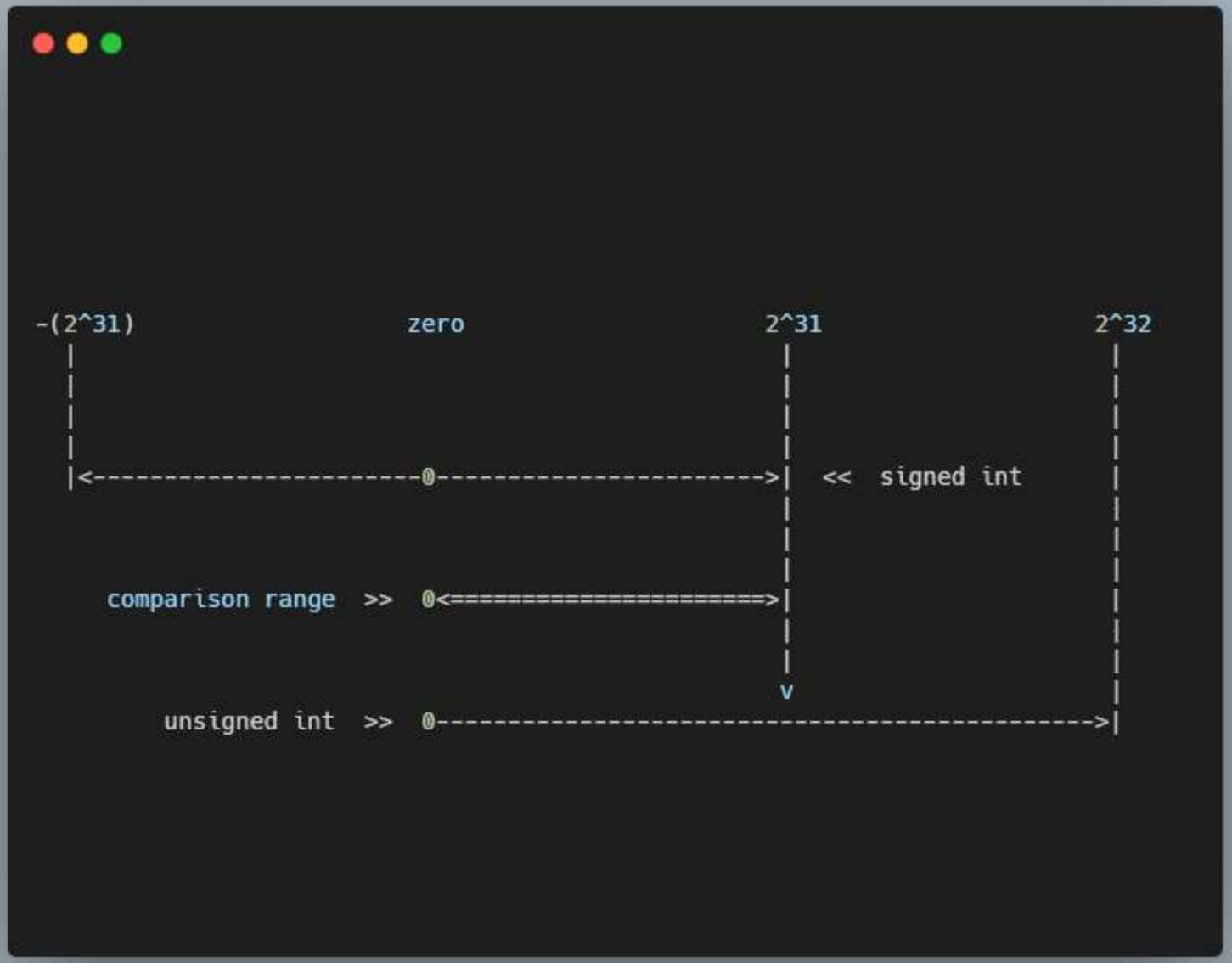
- **SF - sign flag**

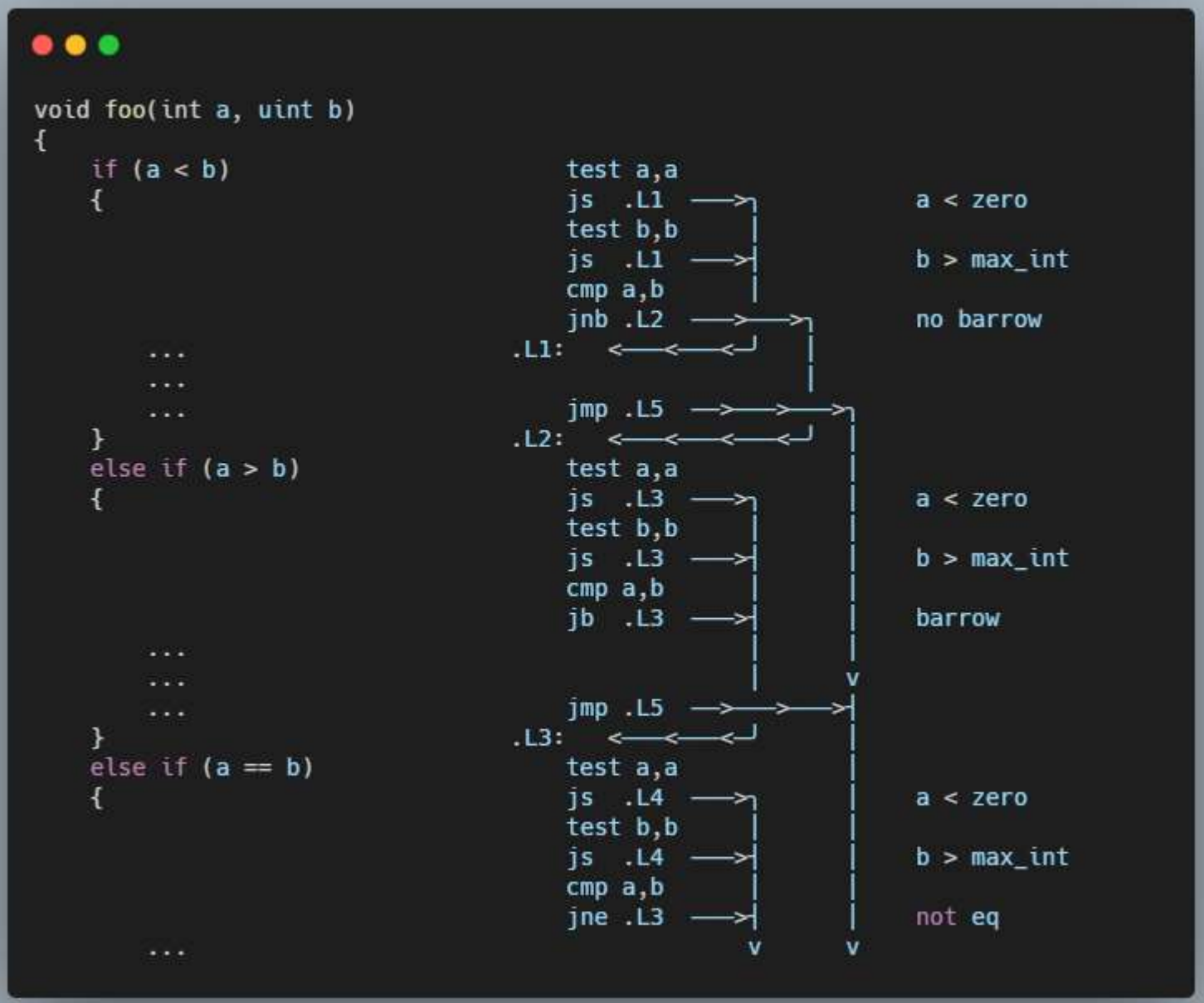
  Set equal to high-order bit of result (0 if positive 1 if negative)

- **OF - overflow flag**

  Set if result is too large a positive number or too small a negative number (excluding sign bit) to fit in destination operand; cleared otherwise

Instruction	Description	signed-ness	Flags	short jump opcodes	near jump opcodes
JO	Jump if overflow		OF = 1	70	0F 80
JNO	Jump if not overflow		OF = 0	71	0F 81
JS	Jump if sign		SF = 1	78	0F 88
JNS	Jump if not sign		SF = 0	79	0F 89
JE **JZ**	Jump if equal Jump if zero		ZF = 1	74	0F 84
JNE **JNZ**	Jump if not equal Jump if not zero		ZF = 0	75	0F 85
JB **JNAE** **JC**	Jump if below Jump if not above or equal Jump if carry	unsigned	CF = 1	72	0F 82
JNB **JAE** **JNC**	Jump if not below Jump if above or equal Jump if not carry	unsigned	CF = 0	73	0F 83
JBE **JNA**	Jump if below or equal Jump if not above	unsigned	CF = 1 or ZF = 1	76	0F 86
JA **JNBE**	Jump if above Jump if not below or equal	unsigned	CF = 0 and ZF = 0	77	0F 87
JL **JNGE**	Jump if less Jump if not greater or equal	signed	SF <> OF	7C	0F 8C
JGE **JNL**	Jump if greater or equal Jump if not less	signed	SF = OF	7D	0F 8D
JLE **JNG**	Jump if less or equal Jump if not greater	signed	ZF = 1 or SF <> OF	7E	0F 8E
JG **JNLE**	Jump if greater Jump if not less or equal	signed	ZF = 0 and SF = OF	7F	0F 8F
JP **JPE**	Jump if parity Jump if parity even		PF = 1	7A	0F 8A
JNP **JPO**	Jump if not parity Jump if parity odd		PF = 0	7B	0F 8B
JCXZ **JECXZ**	Jump if %CX register is 0 Jump if %ECX register is 0		%CX = 0 %ECX = 0	E3	


-(2^31)
zero
2^31
2^32
<< signed int
comparison range >>
unsigned int >>
0


void foo(int a, uint b)
{
if (a < b)
{
...
...
...
}
else if (a > b)
{
...
...
...
}
else if (a == b)
{
...
test a,a
js .L1
test b,b
js .L1
cmp a,b
jnb .L2
.L1:
jmp .L5
.L2:
test a,a
js .L3
test b,b
js .L3
cmp a,b
jb .L3
jmp .L5
.L3:
test a,a
js .L4
test b,b
js .L4
cmp a,b
jne .L3
a < zero
b > max_int
no barrow
a < zero
b > max_int
barrow
a < zero
b > max_int
not eq