Branching Structures:

Naming structure requires the first letter to be (b) and the other the abbreviation of the comparator (eq, ne, lt, gt, le, ge)

Interpreting Bits: Binary and Hexadecimal Numbers:

Binary to Decimal Practices

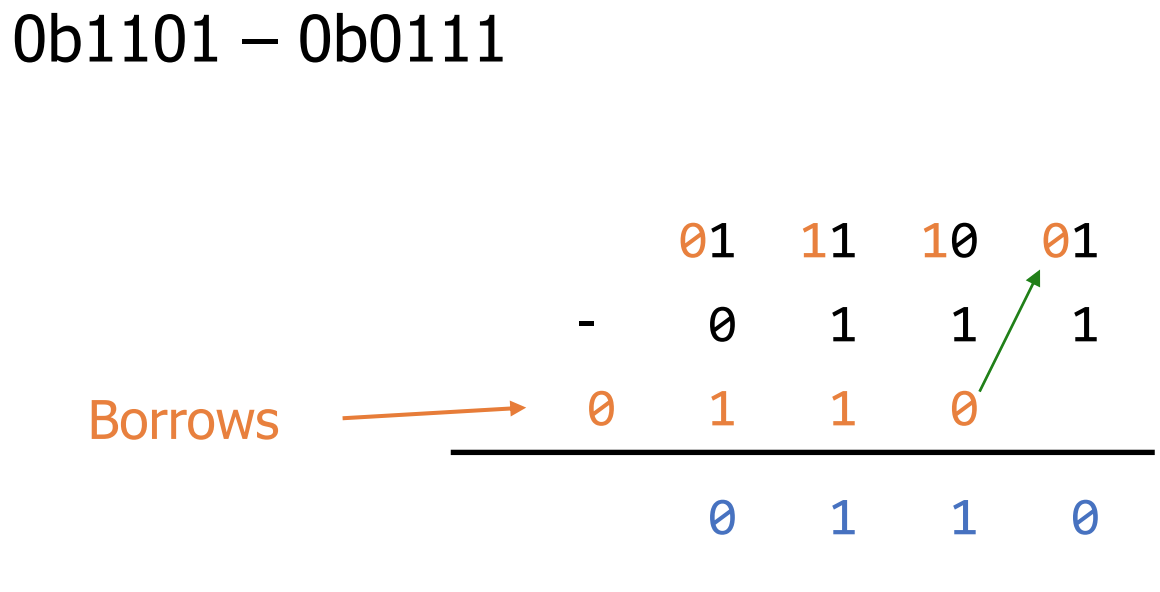
“If you add a zero to the value is doubled”

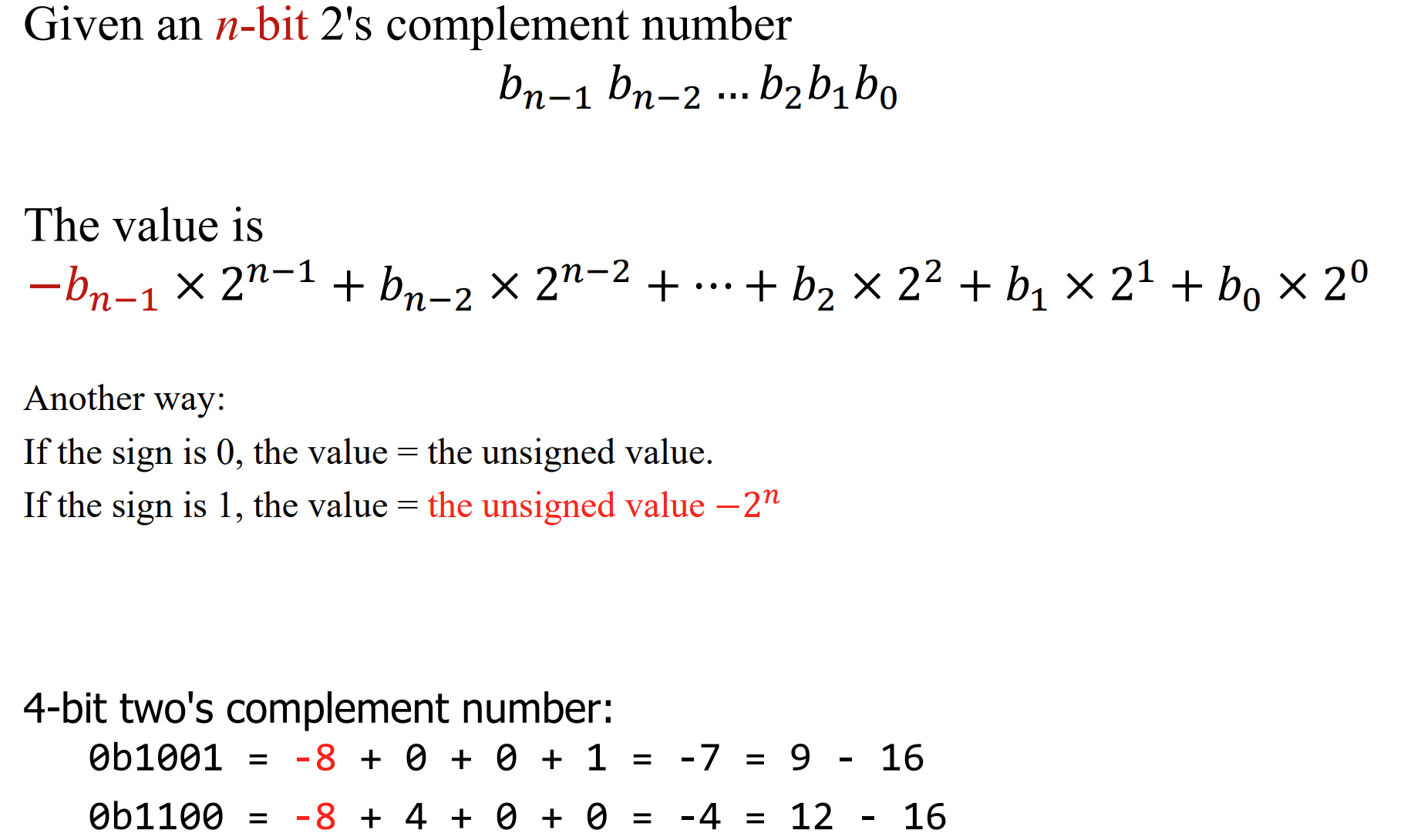
Shift left value is doubled

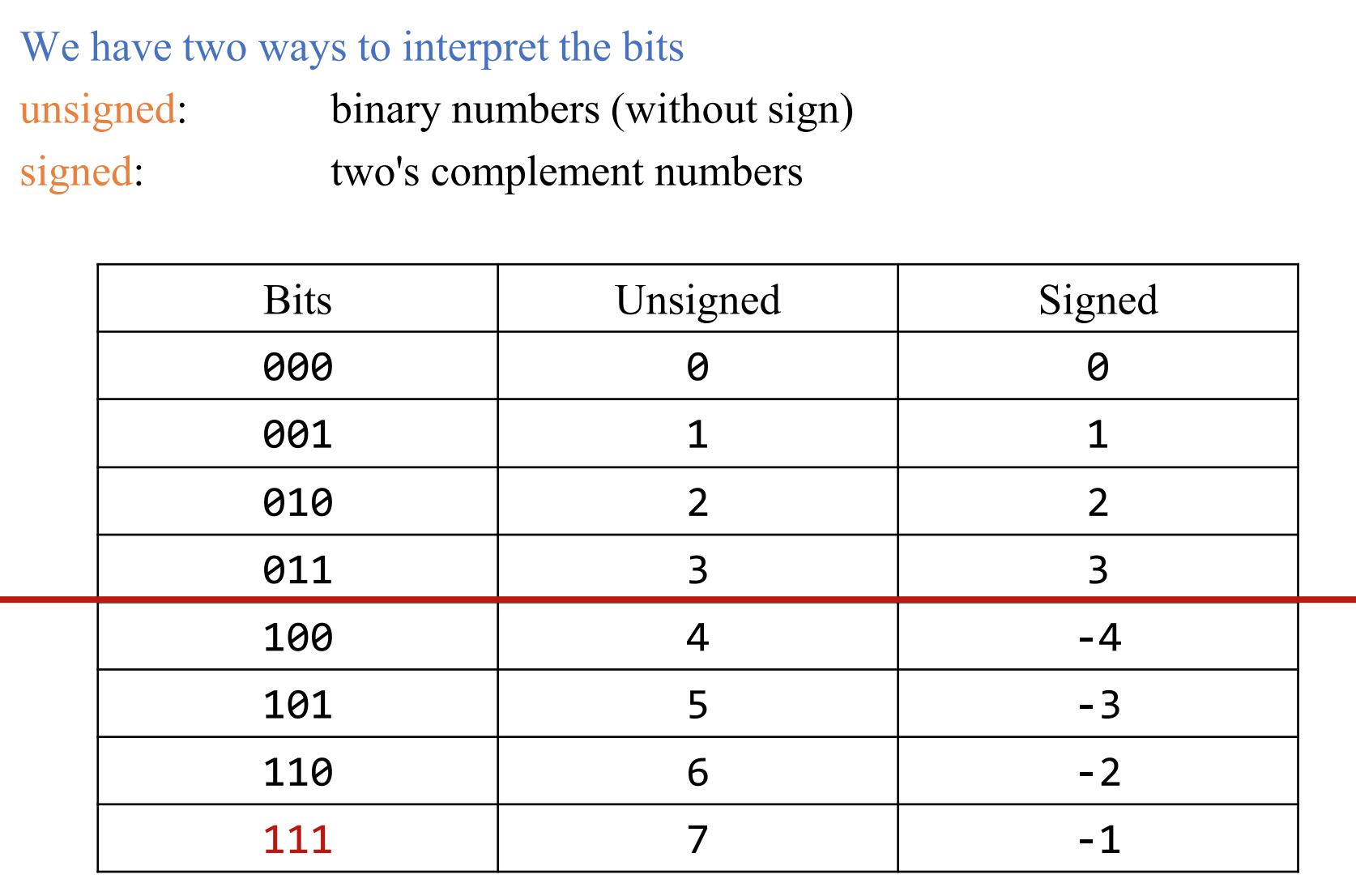
Shift right value is halved

N-bit Binary Numbers

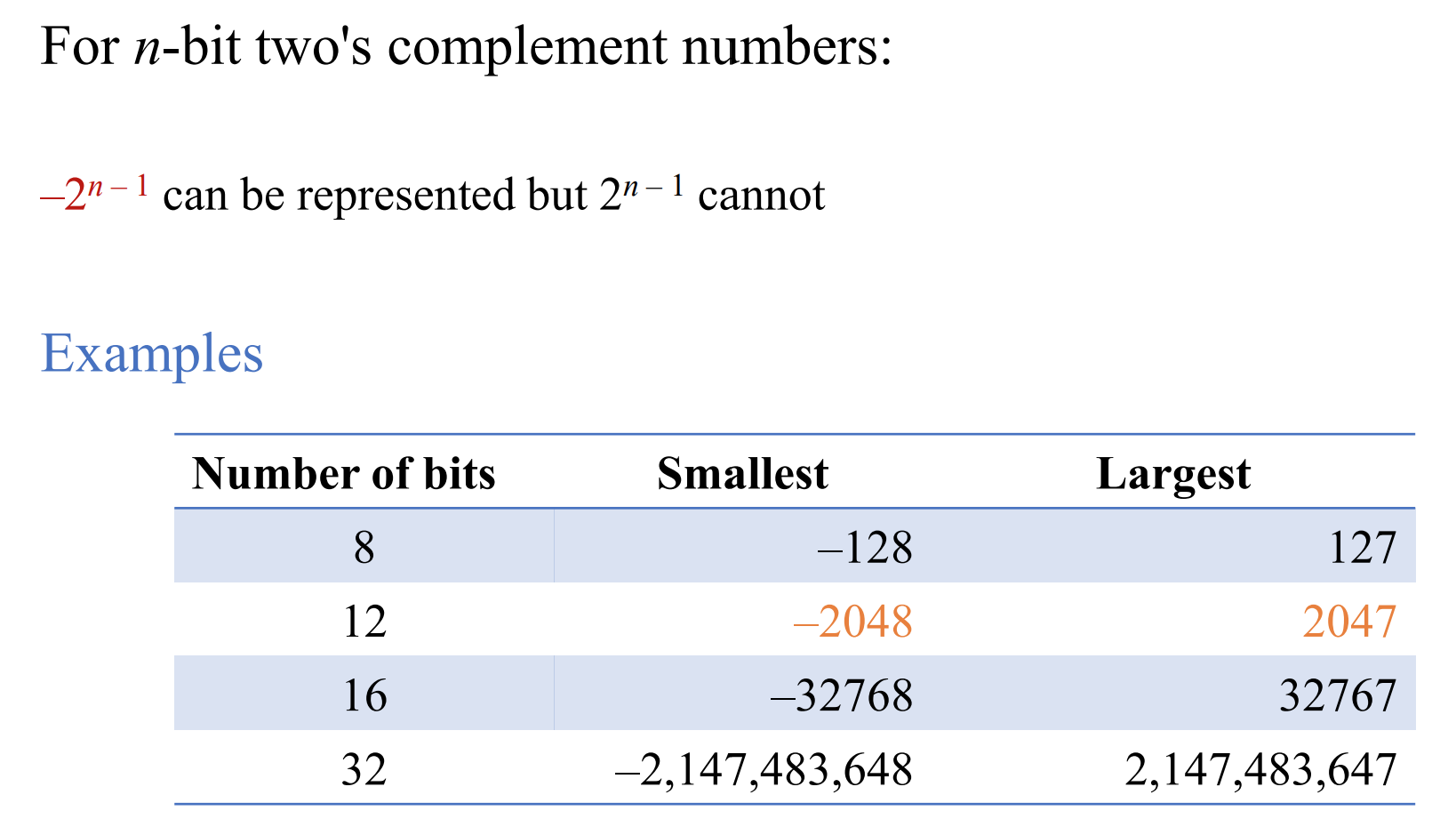
* Very Often # of bits is fixed
* The range of value can be represented by n bits is
  + 0 to 2^n -1
  + For example:
    - 4, 0 to 15
    - 10, 0 to 1023
    - 16, 0 to 65535
    - 32, 0 to 4294967295

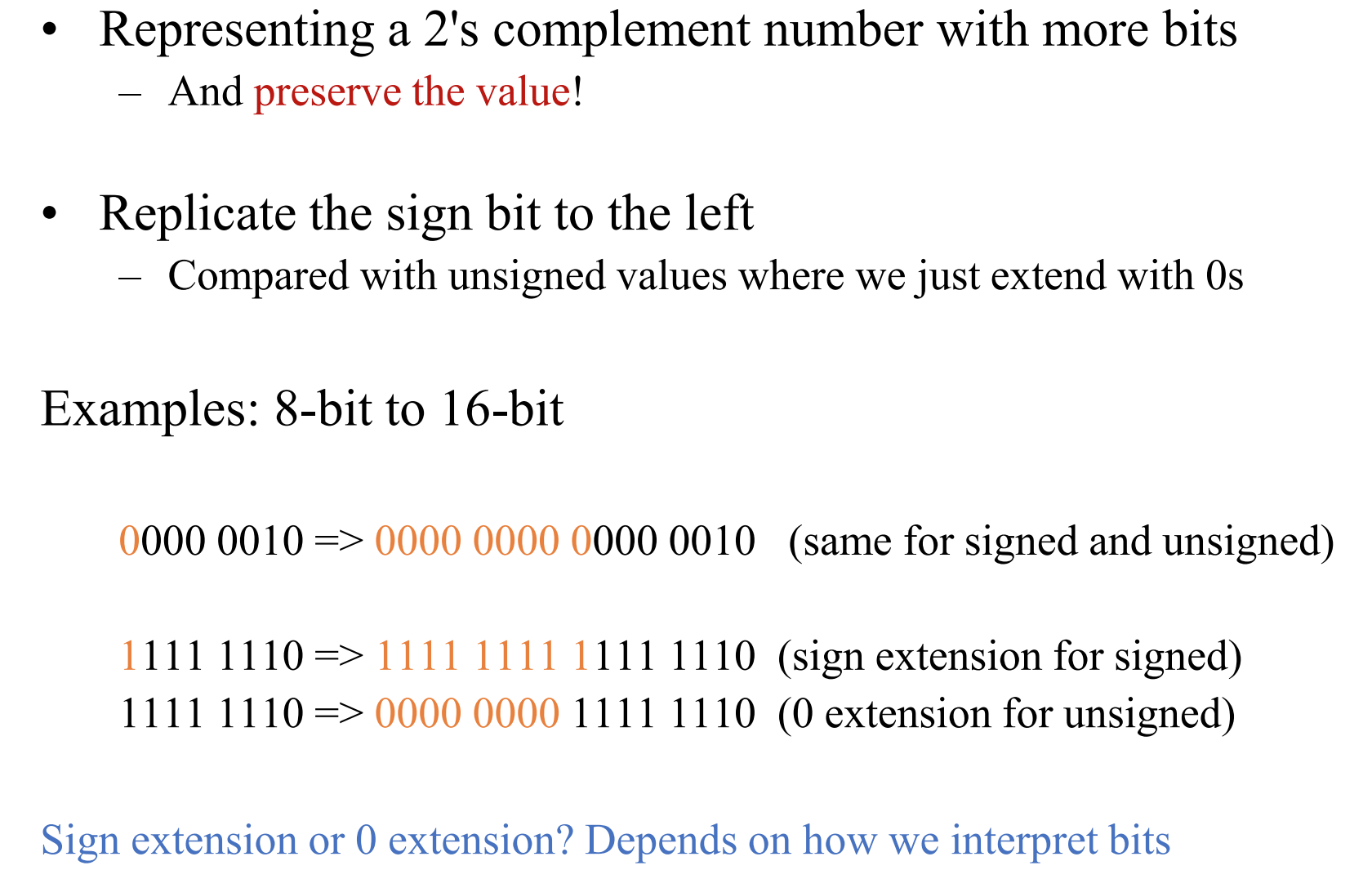


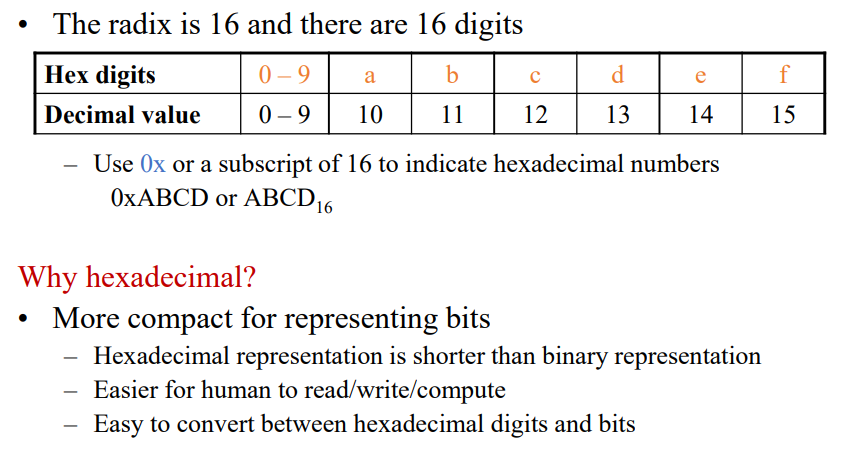




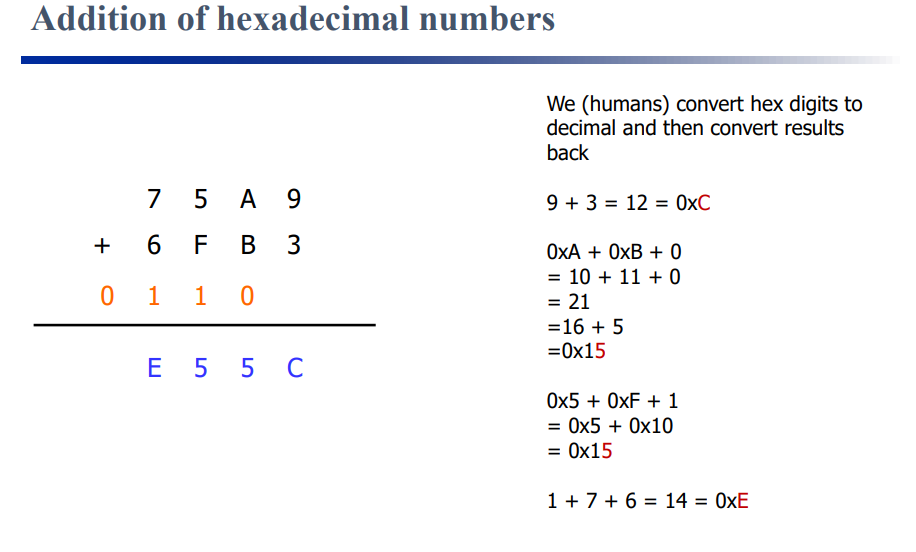
– We always write leading 0s for 2’s complement numbers





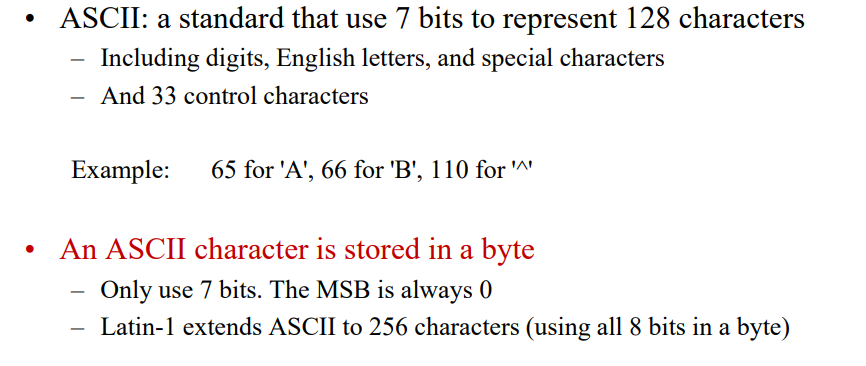


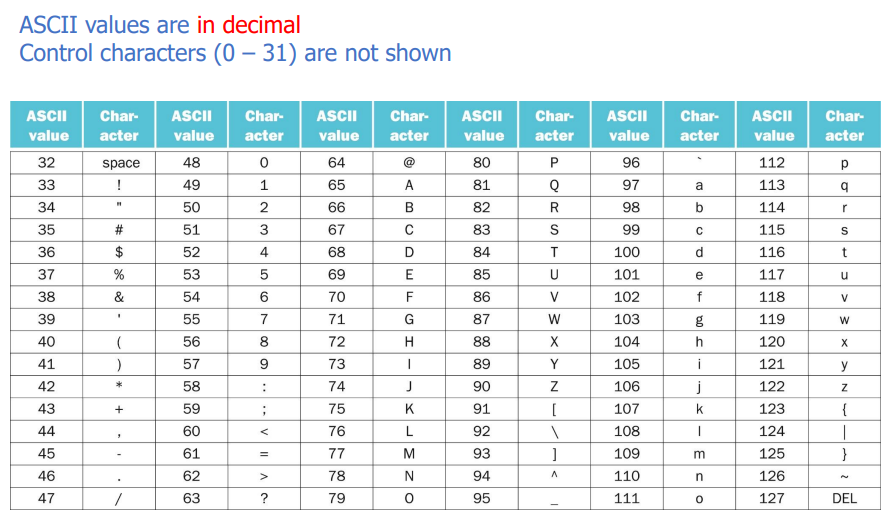
Hex can easily be translated into binary sets of 4



ASCIIl: Representing Characters

Use bits to represent characters





Computers only work in bits

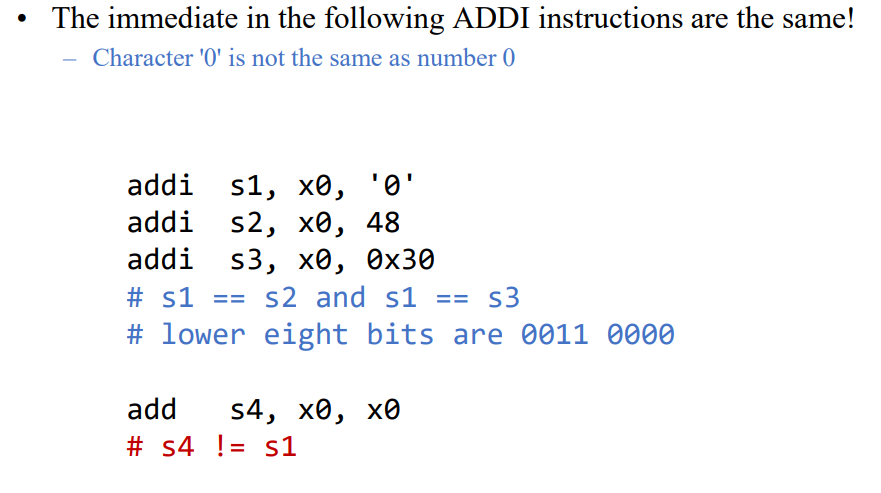
Can write in any format

Memorize

Powers of 2 to 1024

Single hex and 4 bits

Single hex digits 0 to 15

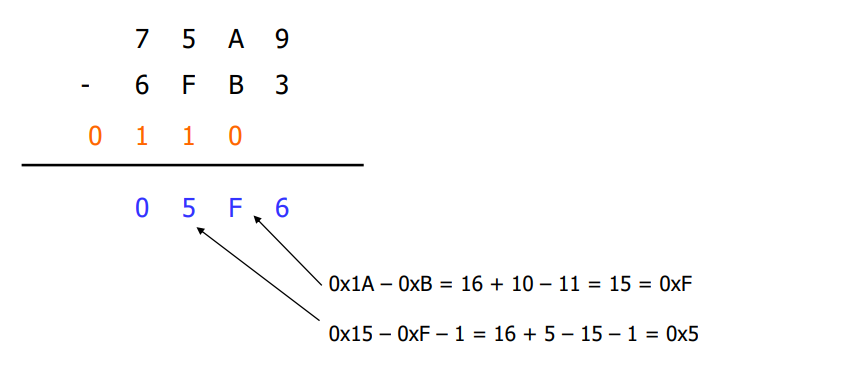


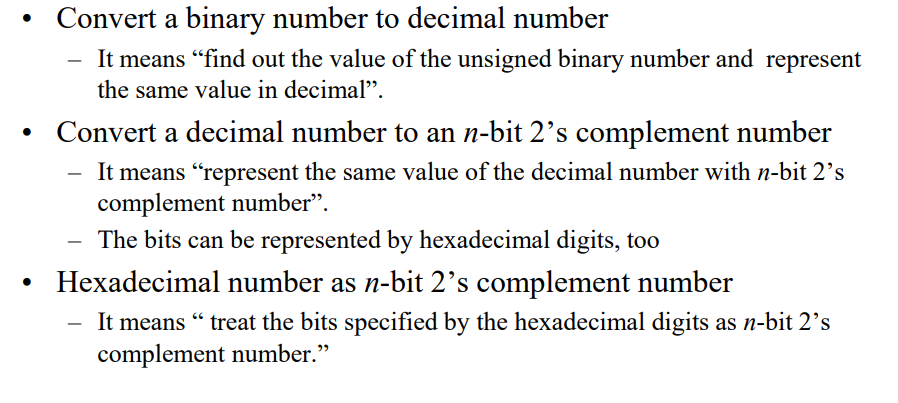
2’s-Complement (signed) numbers

The left-most value is the sign bit

Most-negative: 0b 1000 0000

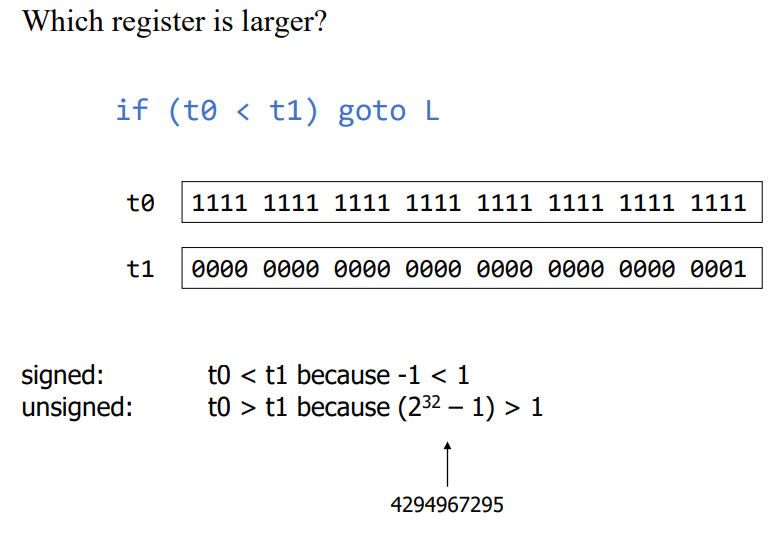
Most-positive: 0b 0111 1111

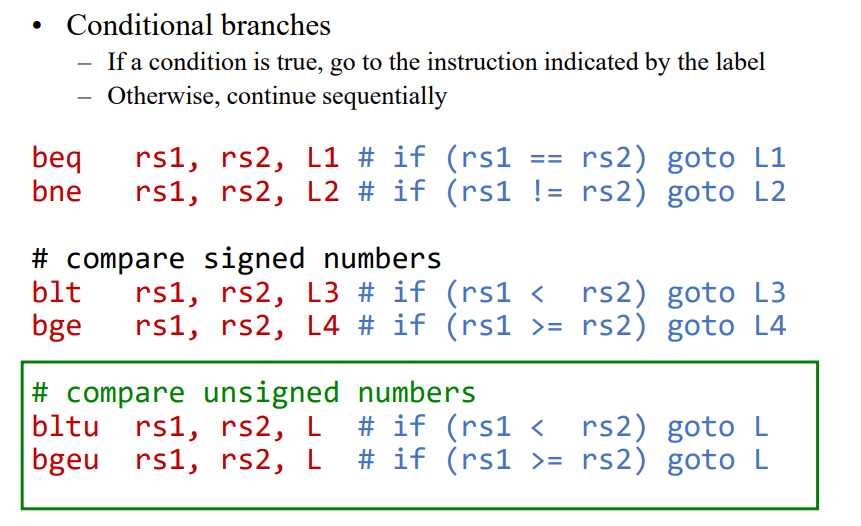




| Lec 2.1: Lots of conversions between binary, decimal and hexadecimal. Just be smart and double check and remember the shortcuts and two complement numbers. |
| --- |

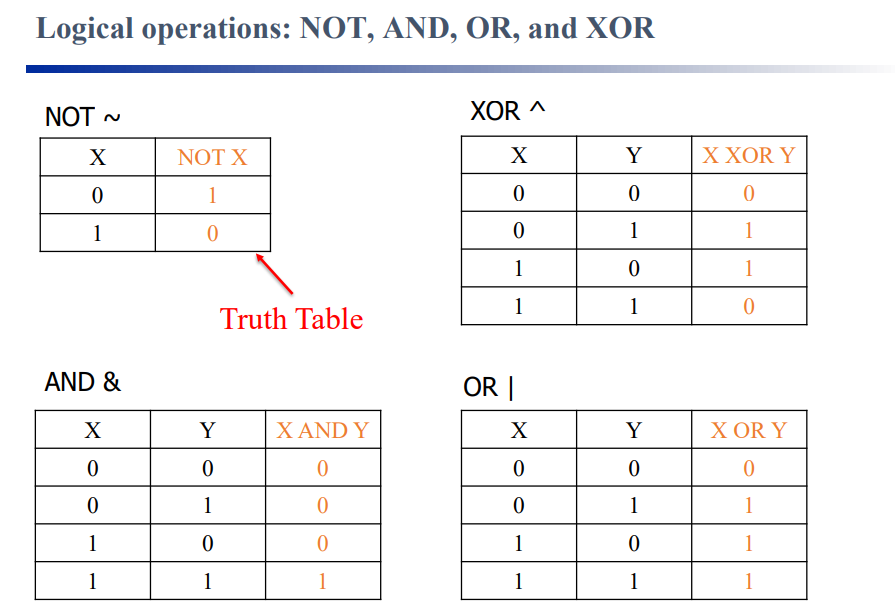
Logical and Bitwise Operations:

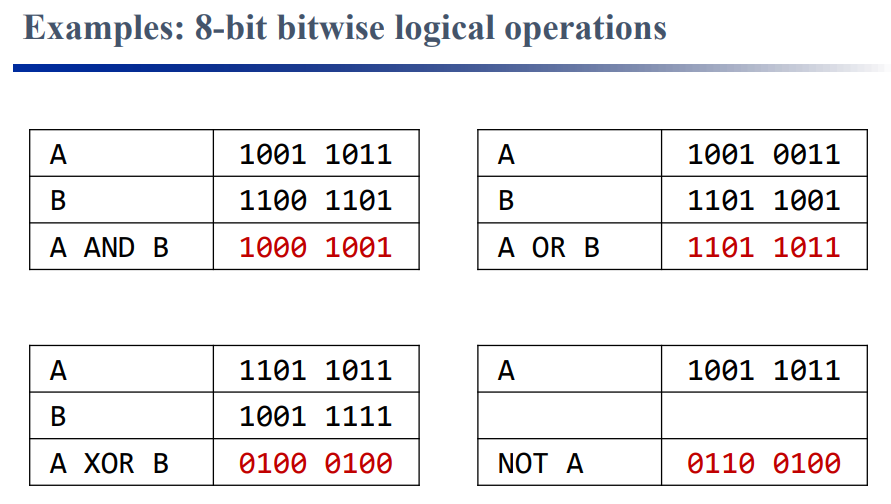




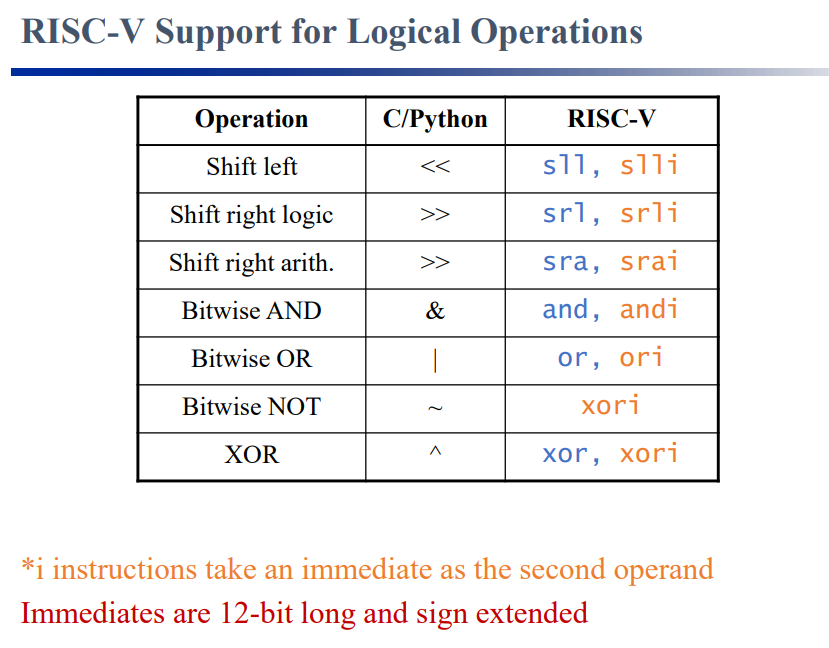
Begu t0, s1, L\_error

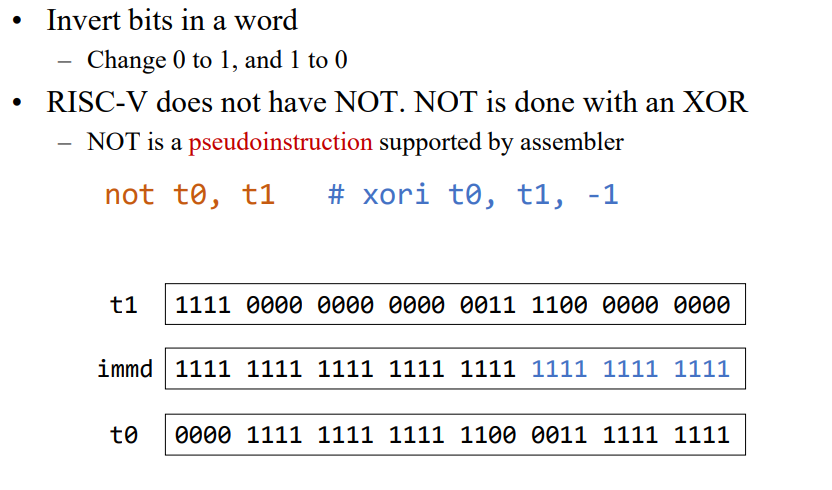
If (t0 < 0) || (t0 >= s1) goto L\_error

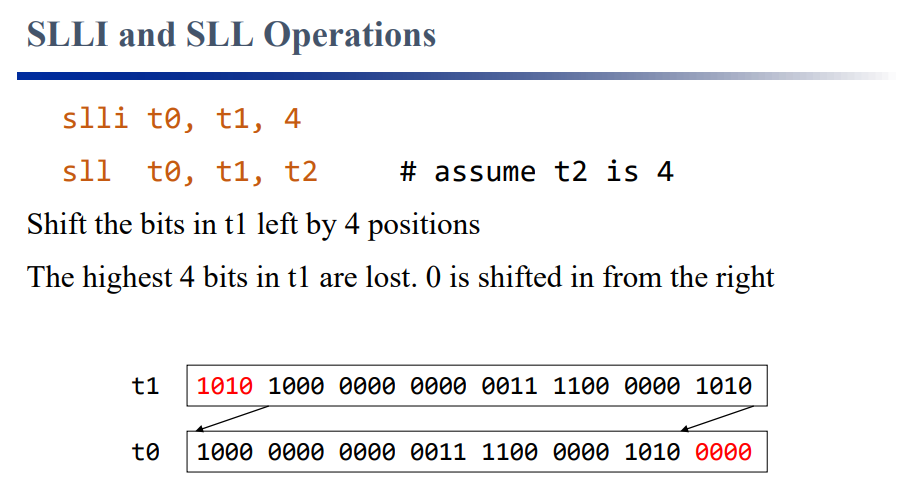


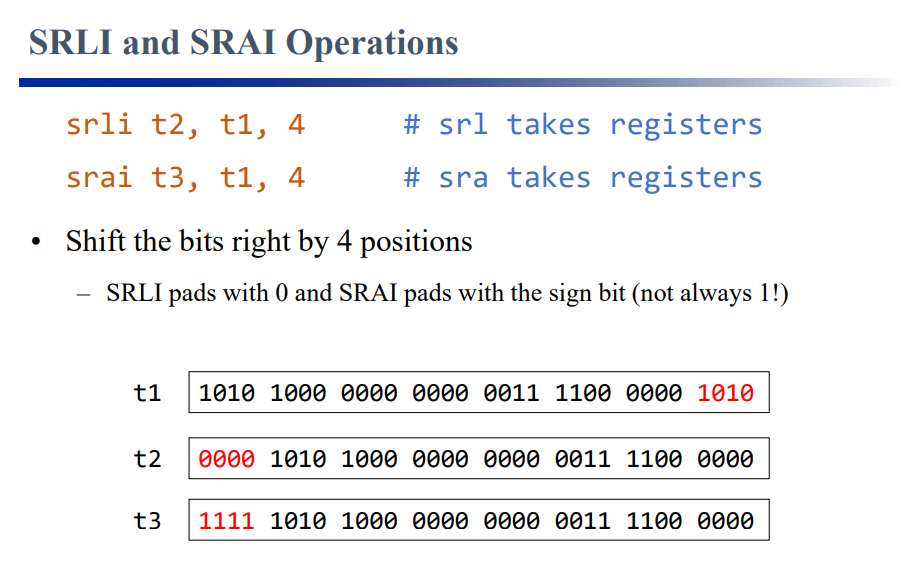


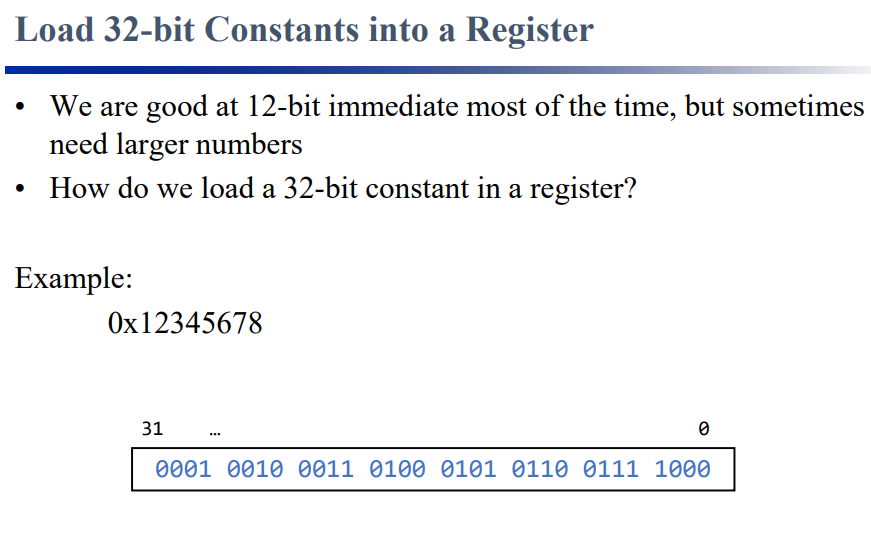
Check by writing them top/bottom

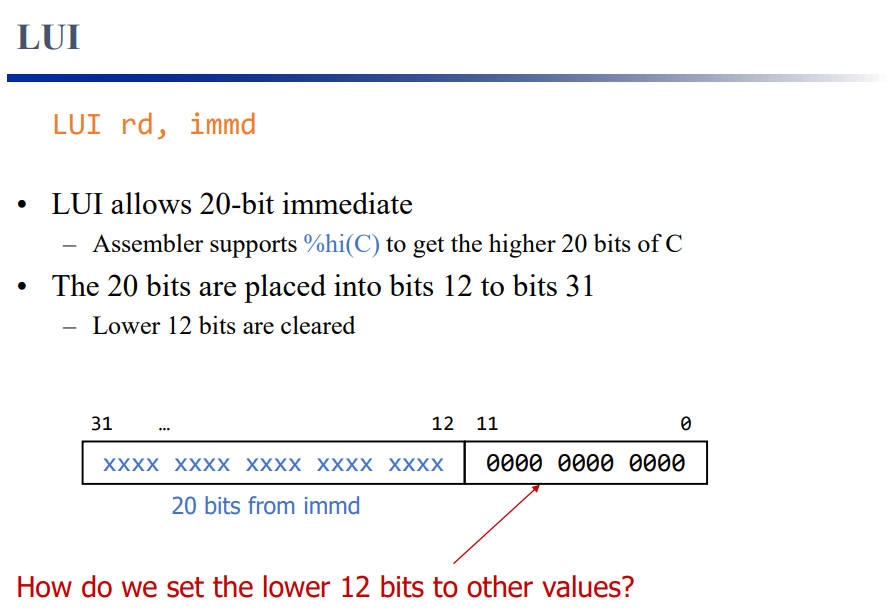


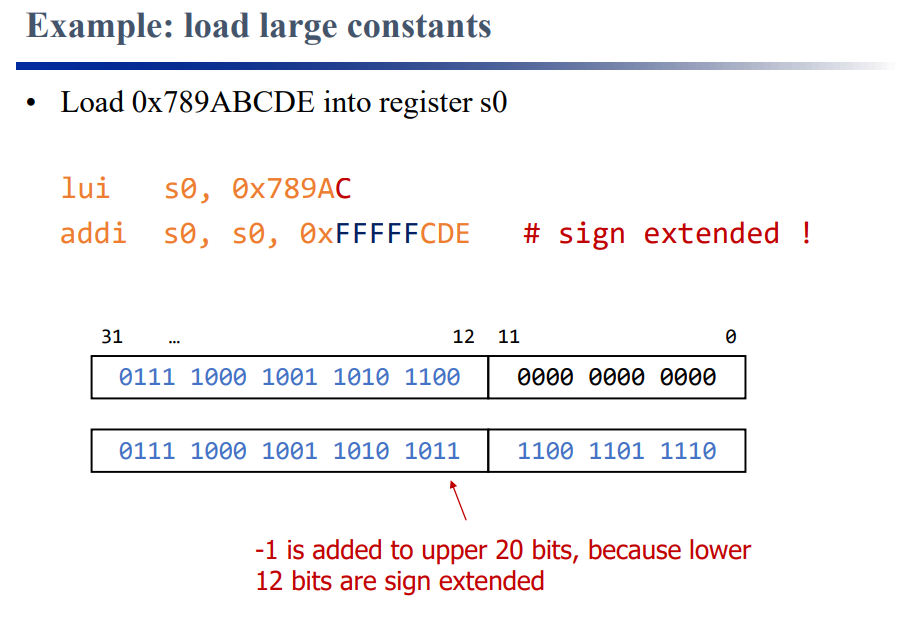


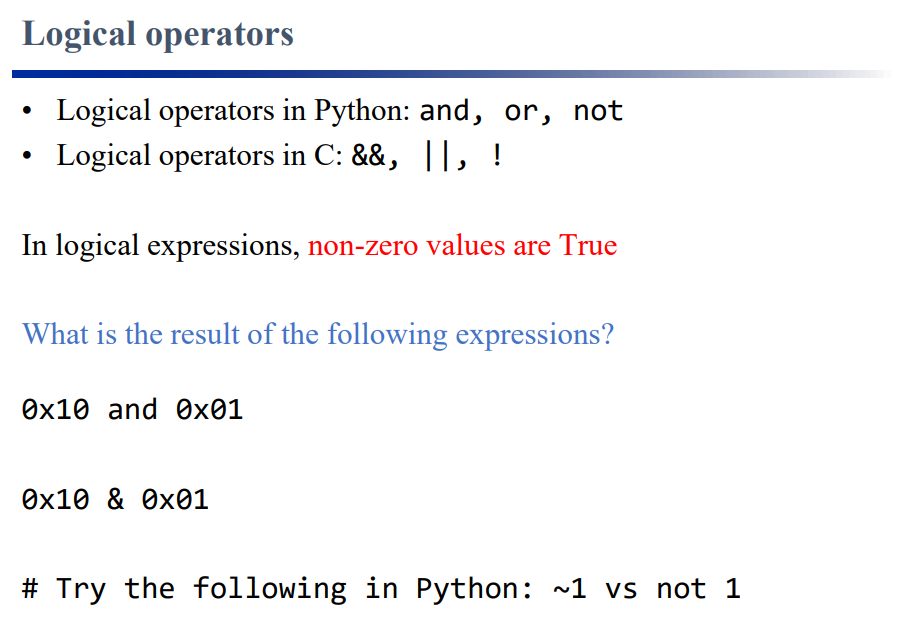


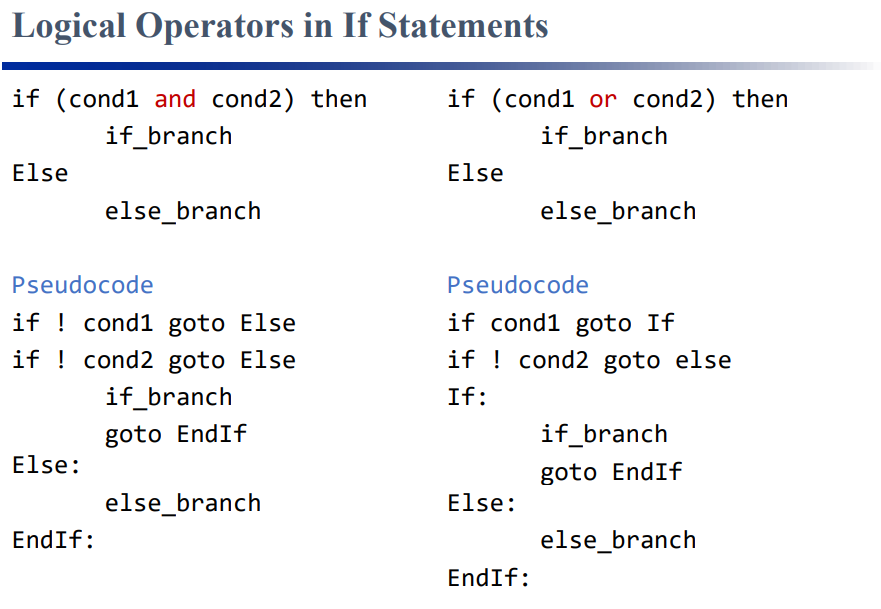












| 1/30: A lot of logical expression and changing binary to fit our code |
| --- |