UNM SCHOOL *of* ENGINEERING

Department of Computer Science

Computer Data Representation (2): Lecture #3 – Part 2

Prof. Soraya Abad-Mota, PhD

Bits, Bytes, and Integers

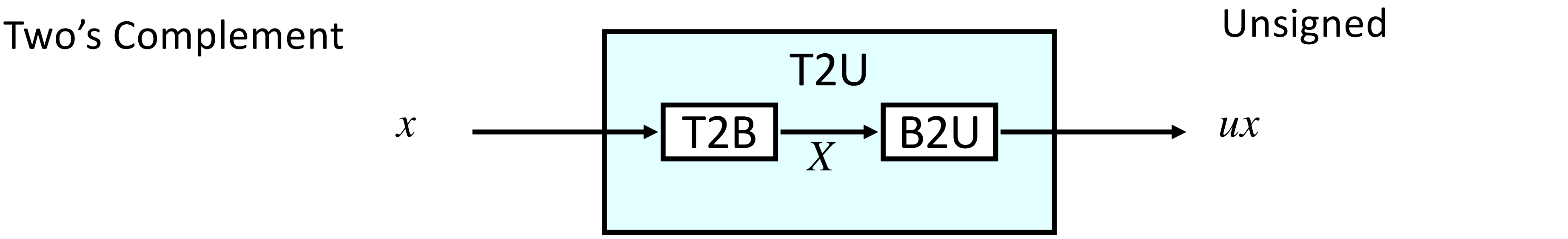
- Integers
  - Representation: unsigned and signed (CS241L)
  - **Conversion, casting**
  - Expanding, truncating
  - Addition, negation, multiplication, shifting
  - Summary
- Representations in memory, pointers, strings

Unsigned & Signed Numeric Values

X	B2U(X)	B2T(X)
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	–8
1001	9	–7
1010	10	–6
1011	11	–5
1100	12	–4
1101	13	–3
1110	14	–2
1111	15	–1

- Equivalence
  - Same encodings for nonnegative values
- Uniqueness
  - Every bit pattern represents unique integer value
  - Each representable integer has unique bit encoding
- $\Rightarrow$ Can Invert Mappings
  - $\mathrm{U2B}(x) \;=\; \mathrm{B2U}^{-1}(x)$
    - Bit pattern for unsigned integer
  - $\mathrm{T2B}(x) \;=\; \mathrm{B2T}^{-1}(x)$
    - Bit pattern for two’s comp integer

Mapping Between Signed & Unsigned



Two's Complement

x

T2U

T2B

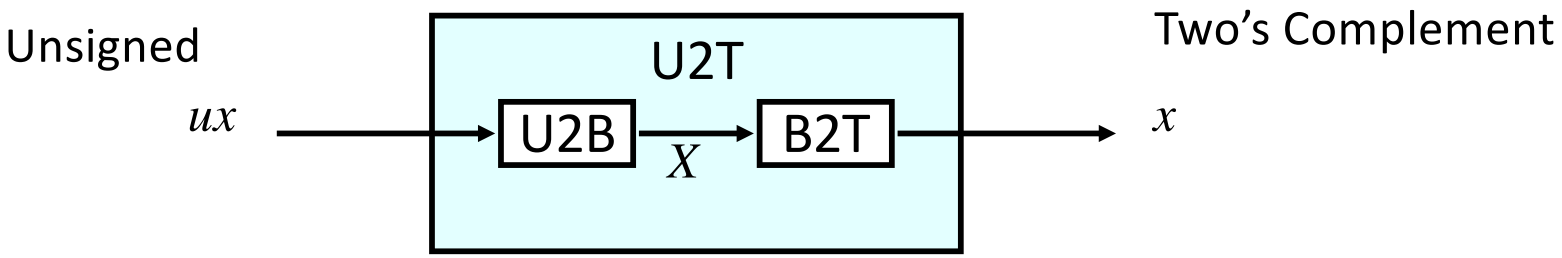
X

B2U

Unsigned

ux

Maintain Same Bit Pattern



Unsigned

ux

U2T

U2B

X

B2T

Two's Complement

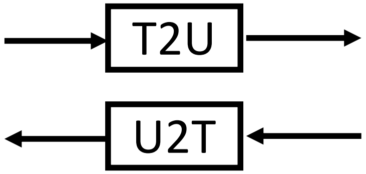
x

Maintain Same Bit Pattern

- Mappings between unsigned and two's complement numbers:
  Keep bit representations and reinterpret

Mapping Signed ↔ Unsigned

Bits	Signed	Unsigned
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	-8	8
1001	-7	9
1010	-6	10
1011	-5	11
1100	-4	12
1101	-3	13
1110	-2	14
1111	-1	15

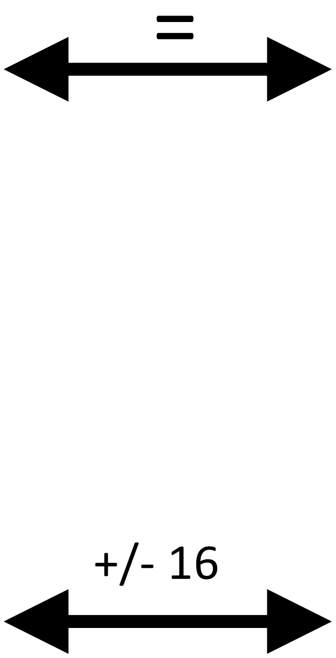


T2U

U2T

Mapping Signed ↔ Unsigned

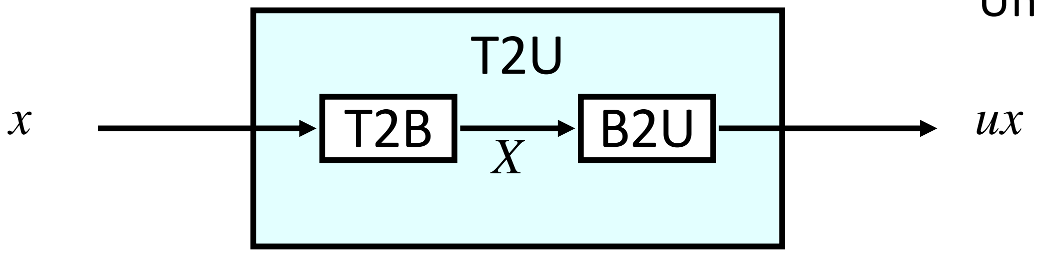
Bits	Signed	Unsigned
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	-8	8
1001	-7	9
1010	-6	10
1011	-5	11
1100	-4	12
1101	-3	13
1110	-2	14
1111	-1	15



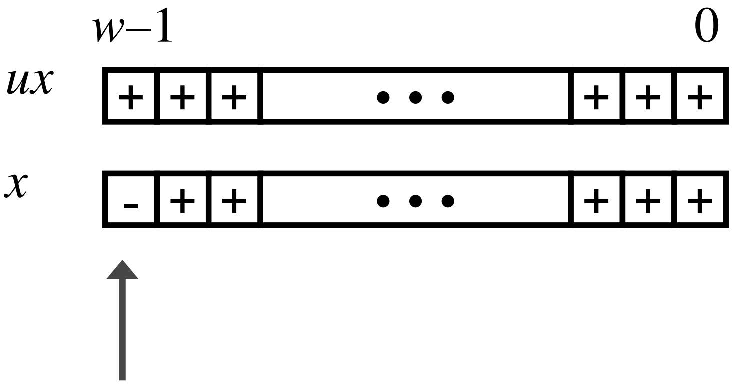
Relation between Signed & Unsigned

Two's Complement

Unsigned



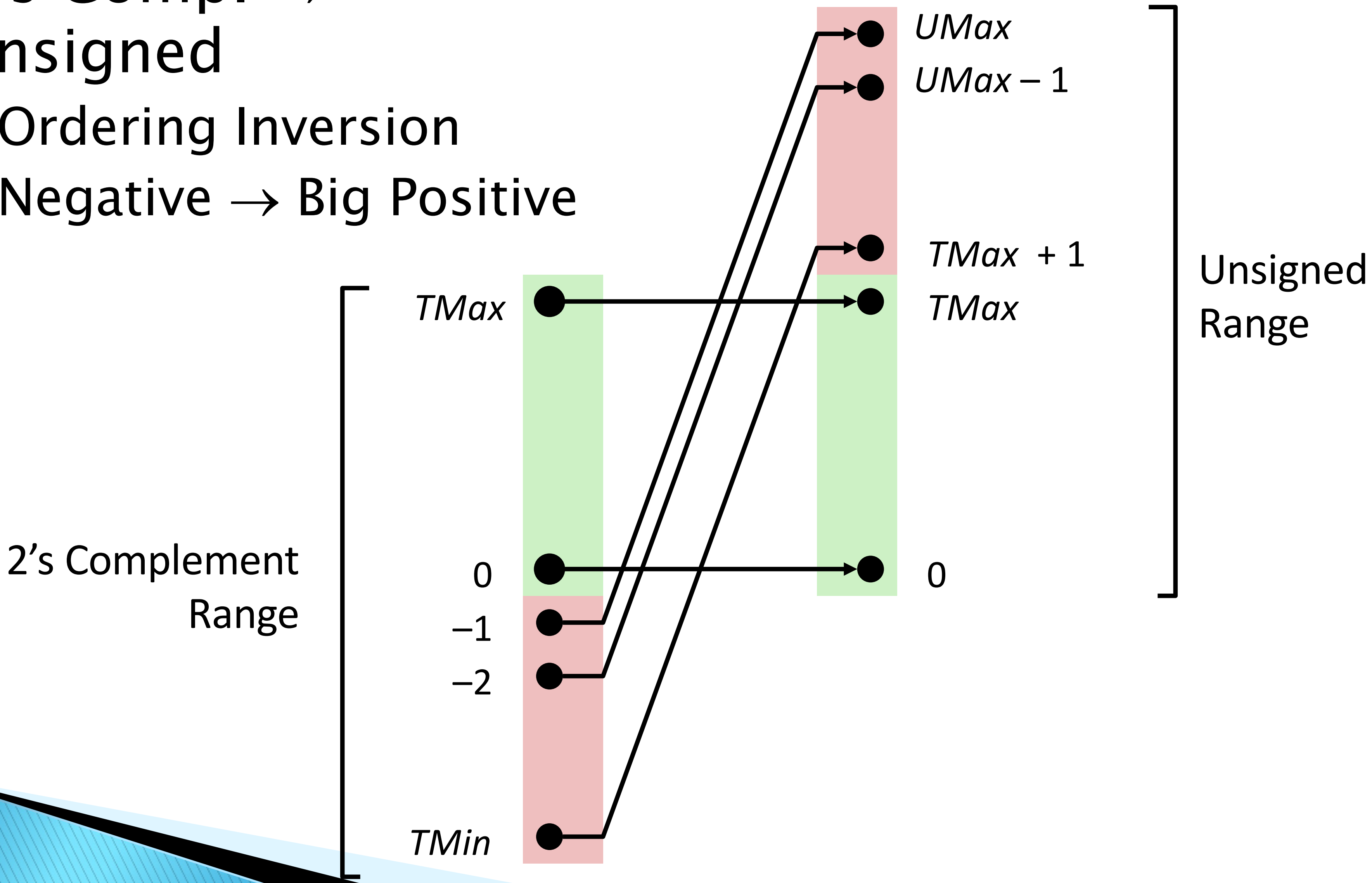
Maintain Same Bit Pattern



Large negative weight
becomes
Large positive weight

Conversion Visualized

- 2’s Comp. → Unsigned
  - Ordering Inversion
  - Negative → Big Positive



Signed vs. Unsigned in C

- Constants
  - By default are considered to be signed integers
  - Unsigned if have “U” as suffix

    ```
    0U, 4294967259U
    ```
- Casting
  - Explicit casting between signed & unsigned same as U2T and T2U

    ```
    int tx, ty;
    unsigned ux, uy;
    tx = (int) ux;
    uy = (unsigned) ty;
    ```
  - Implicit casting also occurs via assignments and procedure calls

    ```
    tx = ux;
    uy = ty;
    ```

Casting Surprises (1)

- Expression Evaluation
  - If there is a mix of unsigned and signed in single expression,
    signed values implicitly cast to unsigned
  - Including comparison operations **<, >, ==, <=, >=**
  - Examples for $W = 32$:
    - TMIN = –2,147,483,648 ,
    - TMAX = 2,147,483,647

Note: *signed* means two's complement

Casting Surprises (2)

Constant$_1$	Relation	Constant$_2$	Type	Eval
0	==	0U	unsigned	1
-1	<	0	signed	1
-1	<	0U	unsigned	0*
2147483647	>	-2147483647-1	signed	1
2147483647U	>	-2147483647-1	unsigned	0*
-1	>	-2	signed	1
(unsigned)-1	>	-2	unsigned	1
2147483647	>	(int) 2147483648U	signed	1*

- Reasoning e.g. -1 < 0U? -1 is transformed to unsigned
  - T2U_w(-1)=UMax_w ➔ 4294967295U < 0U is not true, but is unusual
- Continue on your own, following the table

Summary
Casting Signed ↔ Unsigned: Basic Rules

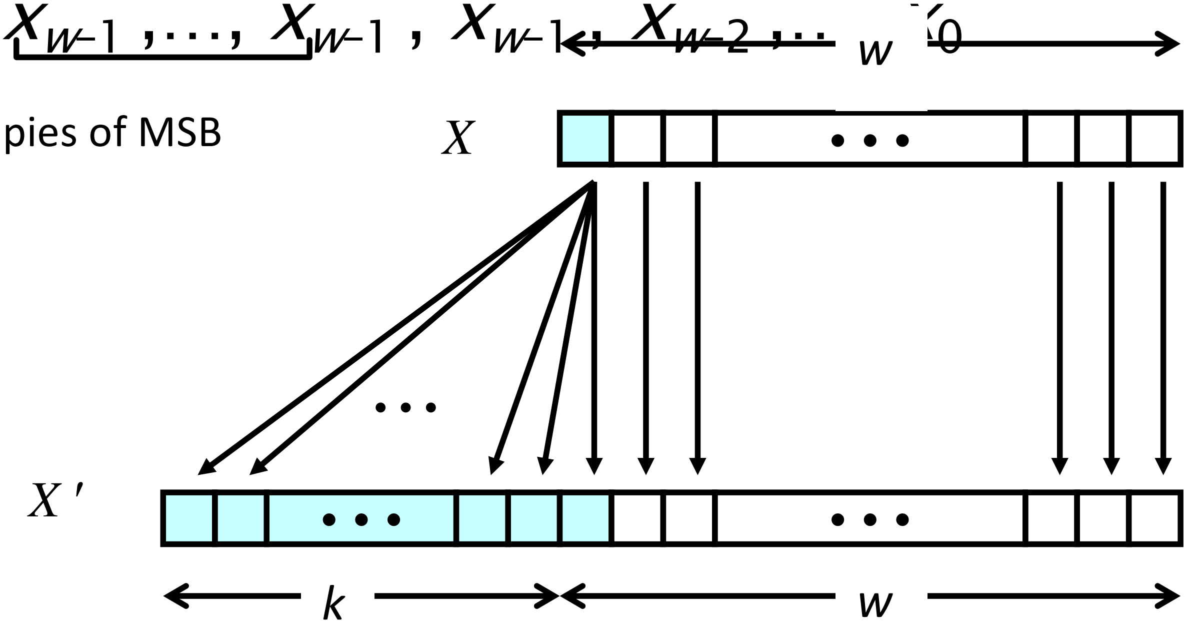
- Bit pattern is maintained but reinterpreted
- Can have unexpected effects: adding or subtracting 2^w (2^4 in examples with 4 bits)
- Expression containing signed and unsigned int
  - `int` is cast to `unsigned`!!

Bits, Bytes, and Integers

- Integers
  - Representation: unsigned and signed (CS241L)
  - Conversion, casting
  - **Expanding, truncating**
  - Addition, negation, multiplication, shifting
  - Summary
- Representations in memory, pointers, strings

Expanding and Truncating: Sign Extension (related to Q about arith. >>)

- Task:
  - Given w–bit signed integer x
  - Convert it to $w+k$–bit integer with same value
- Rule:
  - Make k copies of sign bit:
  - $X' = \underbrace{x_{w-1}, \ldots, x_{w-1}}_{k \text{ copies of MSB}}, x_{w-1}, x_{w-2}, \ldots, x_0$



Sign Extension Example

```
short int x =  15213;
int      ix = (int) x;
short int y = -15213;
int      iy = (int) y;
```

	Decimal	Hex	Binary
x	15213	3B 6D	00111011 01101101
ix	15213	00 00 3B 6D	00000000 00000000 00111011 01101101
y	-15213	C4 93	11000100 10010011
iy	-15213	FF FF C4 93	11111111 11111111 11000100 10010011

- Converting from smaller to larger integer data type
- C automatically performs sign extension

Practice problem 2.22 (now)

- Show that each of the following bit vectors is, a two's-complement representation of −5 by applying Equation 2.3 (conv. B2T)

A. 1011

B. 11011

c. 111011

You may also do the one I left last time:

```
  101
 1101
11101
```

Summary of Expanding, Truncating: Basic Rules

- Expanding (e.g., short int to int)
  - Unsigned: zeros added
  - Signed: sign extension
  - Both yield expected result

- Truncating (e.g., unsigned to unsigned short)
  - Unsigned/signed: bits are truncated
  - Result reinterpreted
  - Unsigned: mod operation
  - Signed: similar to mod
  - For small numbers yields expected behavior

Practice Truncation (on your own)

- With Problem 2.24