Announcements

- Assignment 1
  - *Cellular Automata* with strings in C
  - Due 5:00 p.m., Friday, September 26
- Test 1
  - Friday, September 26
  - Covers up to Lecture 8

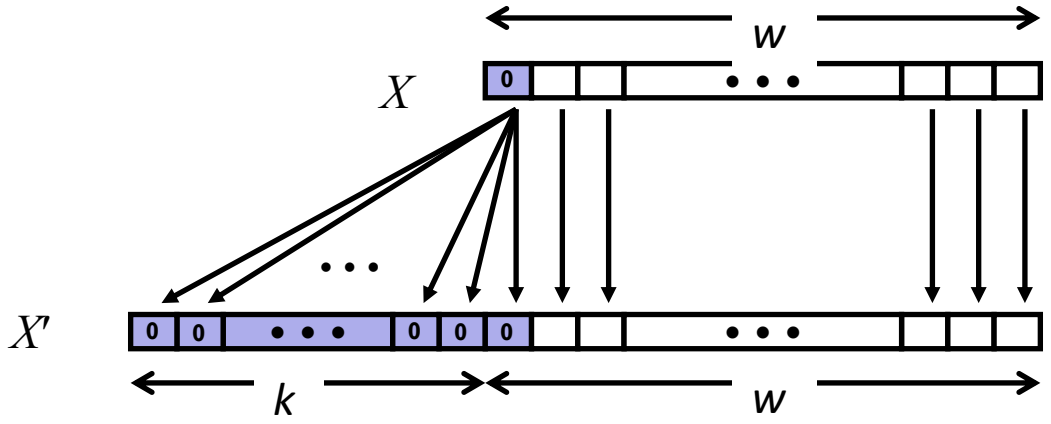
Lecture 8

Expansion and Trunction

CPSC 275
Introduction to Computer Systems

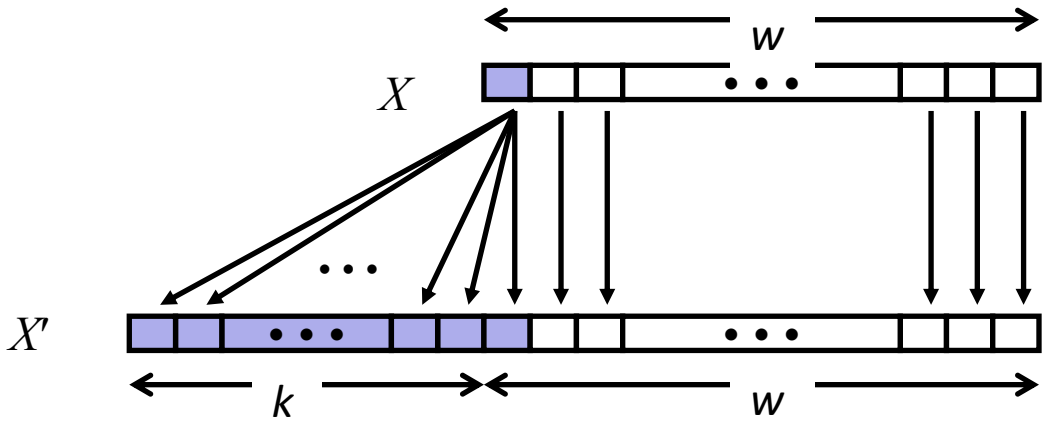
Expanding Bit Representation

- Conversion between integers having different word sizes
  - Must retain the same numeric value
  - May not be possible when the destination data type is too small
  - Conversion from a smaller to a larger data type always be possible
- For unsigned integers, add leading zeros (*zero extension*):



Sign Extension

- Conversion between integers having different word sizes
  - Must retain the same numeric value
  - May not be possible when the destination data type is too small
  - Conversion from a smaller to a larger data type always be possible
- For two's complement integers, add the sign bit (*sign extension*):



Sign Extension Example

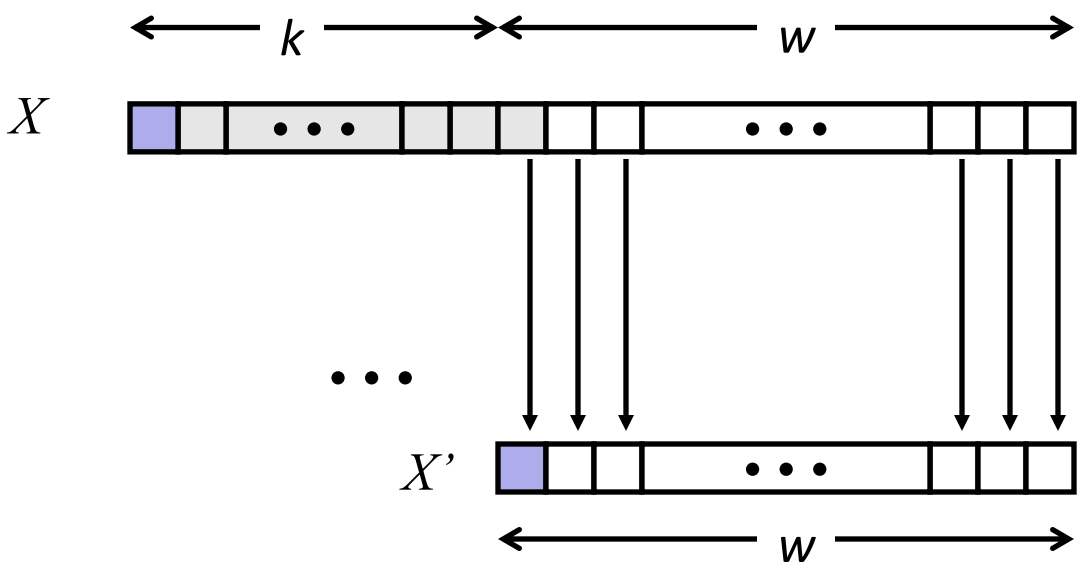
```
short int x =  15213;
int ix = (int) x;
short int y = -15213;
int iy = (int) y;
```

	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

Truncation

- Task:
  - Given k+*w*-bit signed or unsigned integer *X*
  - Convert it to *w*-bit integer X' with same value for "small enough" X
- Rule:
  - Drop top *k* bits:
  - *X'* = $x_{w-1}, x_{w-2}, \ldots, x_0$



Truncation Example

No sign change

	-16	8	4	2	1
2 =	0	0	0	1	0

	-8	4	2	1
2 =	0	0	1	0

	-16	8	4	2	1
-6 =	1	1	0	1	0

	-8	4	2	1
-6 =	1	0	1	0

Sign change

	-16	8	4	2	1
10 =	0	1	0	1	0

	-8	4	2	1
-6 =	1	0	1	0

	-16	8	4	2	1
-10 =	1	0	1	1	0

	-8	4	2	1
6 =	0	1	1	0

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
  - For small (in magnitude) numbers yields expected behavior