CS 261
Fall 2016

Mike Lam, Professor

Binary Arithmetic

Binary Arithmetic

- Topics
  - Basic addition
  - Overflow
  - Multiplication

Basic addition

- Binary and hex addition are fundamentally the same as decimal addition
  - Add digit-by-digit, using a carry as necessary
  - Result generally requires more bits than the two operands

Dec

```
  12540
+  4683
```

Bin

```
  10011100
+  1010110
```

Hex

```
  b0994f
+   7120
```

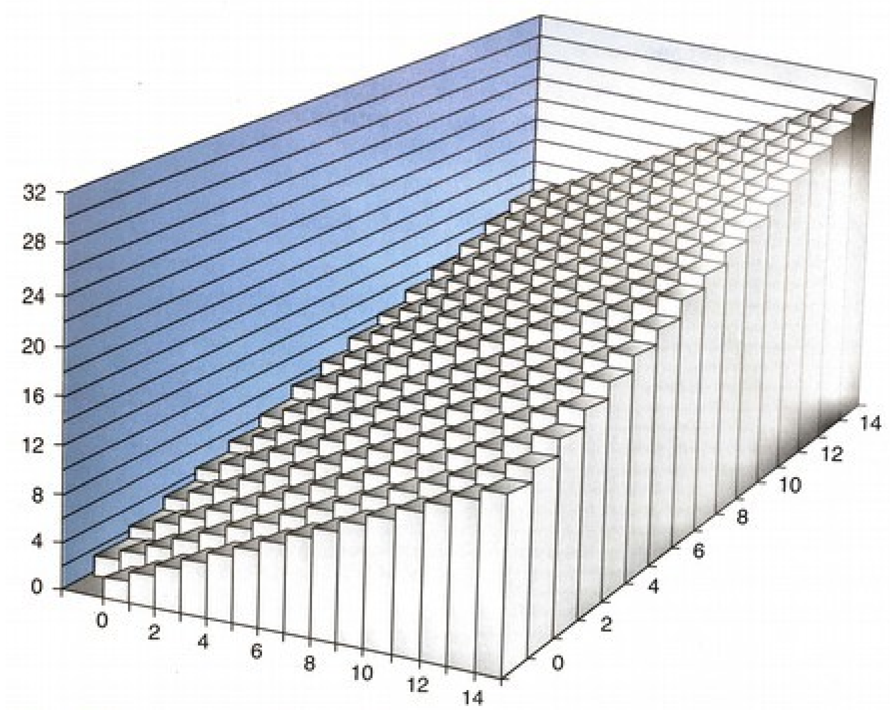


Figure 2.21 Integer addition. With a 4-bit word size, the sum could require 5 bits.

Basic addition

- Binary and hex addition are fundamentally the same as decimal addition
  - Add digit-by-digit, using a carry as necessary
  - Result generally requires more bits than the two operands

Dec

```
  11
 12540
+ 4683
 17223
```

Bin

```
   111
 10011100
+ 1010110
 11110010
```

Hex

```
   1
 b0994f
+  7120
 b10a6f
```

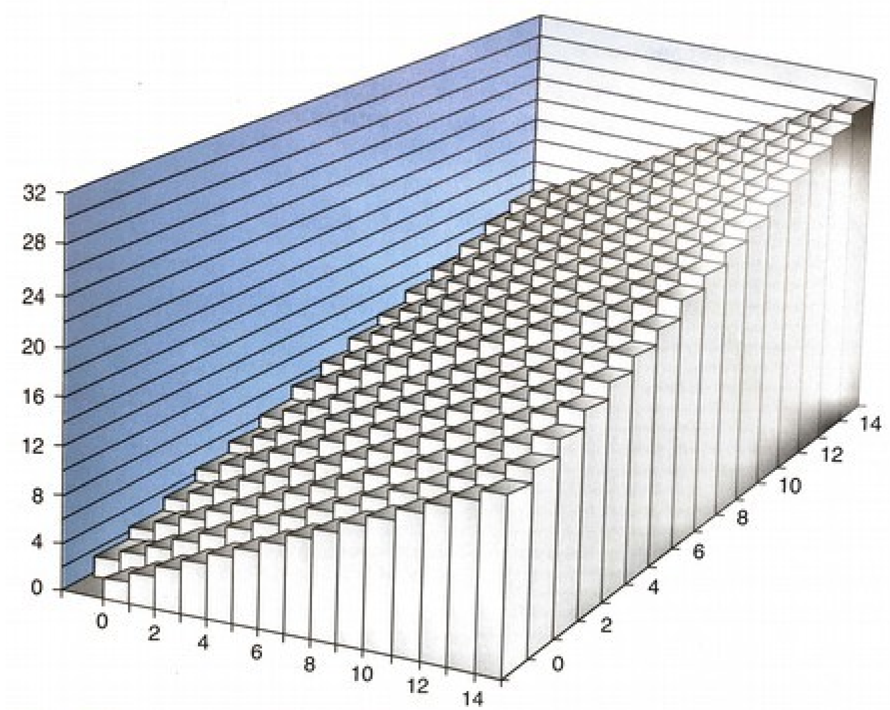


Figure 2.21 Integer addition. With a 4-bit word size, the sum could require 5 bits.

Overflow

- Unsigned addition is subject to overflow
  - Caused by truncation to integer size

```
 1
   994f
+  7120
 10a6f = 0a6f
```

Truncation!

(assume a 16-bit integer)

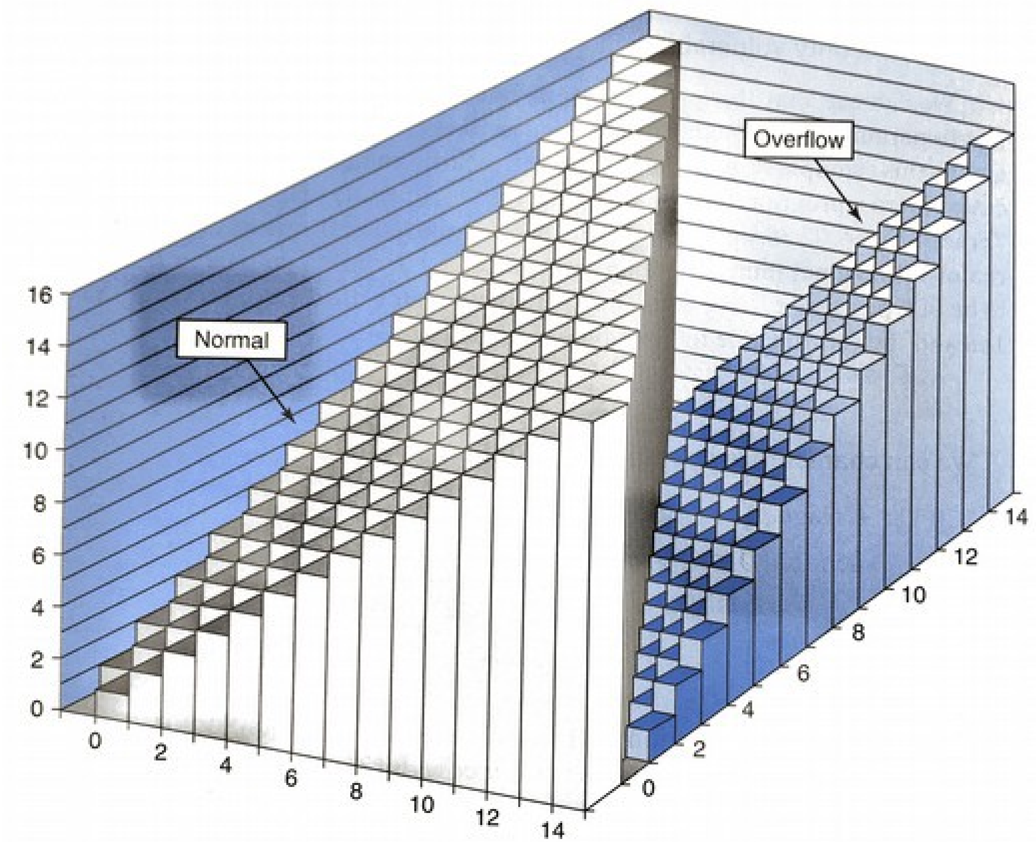


Figure 2.23 Unsigned addition. With a 4-bit word size, addition is performed modulo 16.

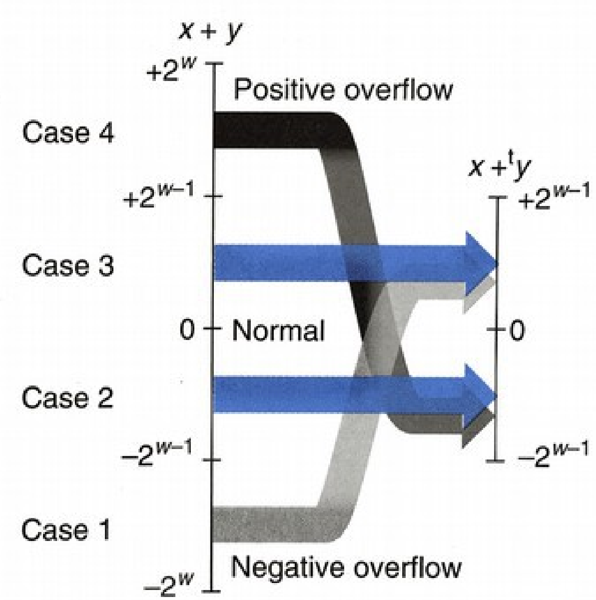
Overflow

- Two's complement addition
  - Works just like unsigned addition mechanically
  - Subject to both positive and negative overflow
  - Overflows if carry-in and carry-out differ for sign bit

x	y	$x + y$	$x +^t_4 y$	Case
−8 [1000]	−5 [1011]	−13 [10011]	3 [0011]	1
−8 [1000]	−8 [1000]	−16 [10000]	0 [0000]	1
−8 [1000]	5 [0101]		−3 [1101]	2
2 [0010]	5 [0101]		7 [0111]	3
5 [0101]	5 [0101]	10 [01010]	−6 [1010]	4

Figure 2.25 Two's-complement addition examples. The bit-level representation of the 4-bit two's-complement sum can be obtained by performing binary addition of the operands and truncating the result to 4 bits.

Figure 2.24
Relation between integer and two's-complement addition. When $x + y$ is less than -2^{w-1}, there is a negative overflow. When it is greater than or equal to 2^{w-1}, there is a positive overflow.



Overflow

- Two's complement addition
  - Works just like unsigned addition mechanically
  - Subject to both positive and negative overflow
  - Overflows if carry-in and carry-out differ for sign bit

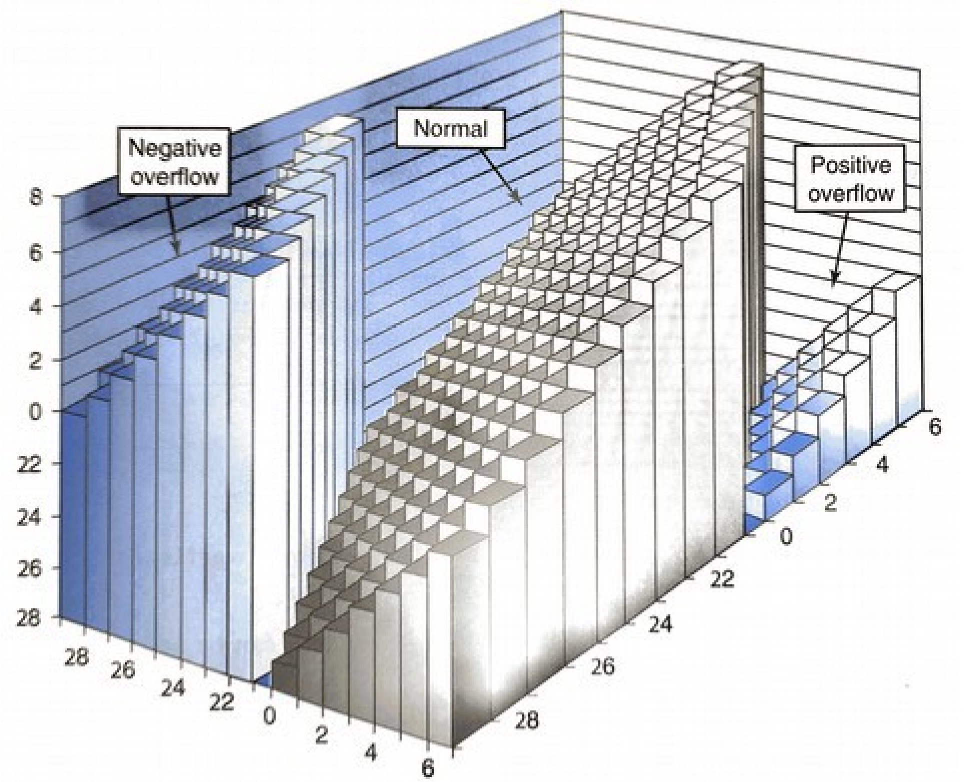


Figure 2.26 Two's-complement addition. With a 4-bit word size, addition can have a negative overflow when $x + y < -8$ and a positive overflow when $x + y \geq 8$.

Multiplication

- Like addition, fundamentally the same as base 10
  - Actually, it's even simpler!
  - Same regardless of encoding

```
  101 (5)
x  11 (3)
  101
 101
 1111 (15)
```

Mode	x		y		$x \cdot y$		Truncated $x \cdot y$	
Unsigned	5	[101]	3	[011]	15	[001111]	7	[111]
Two's complement	–3	[101]	3	[011]	–9	[110111]	–1	[111]
Unsigned	4	[100]	7	[111]	28	[011100]	4	[100]
Two's complement	–4	[100]	–1	[111]	4	[000100]	–4	[100]
Unsigned	3	[011]	3	[011]	9	[001001]	1	[001]
Two's complement	3	[011]	3	[011]	9	[001001]	1	[001]

Figure 2.27 Three-bit unsigned and two's-complement multiplication examples. Although the bit-level representations of the full products may differ, those of the truncated products are identical.

Multiplication

- Special case: multiply by powers of 2 (shift left)

```
2 << 1 = 4        (2 * 2)
1 << 2 = 4        (1 * 2 * 2)

1 << 4 = 16       (1 * 2 * 2 * 2 * 2)
4 << 1 = 8        (4 * 2)
4 << 2 = 16       (4 * 2 * 2)
```

Division

- General case is expensive!
  - Special case: divide by powers of two (shift right)

k	>> k (binary)	Decimal	$12{,}340/2^k$
0	0011000000110100	12,340	12,340.0
1	0001100000011010	6,170	6,170.0
4	0000001100000011	771	771.25
8	0000000000110000	48	48.203125

Figure 2.28 Dividing unsigned numbers by powers of 2. The examples illustrate how performing a logical right shift by k has the same effect as dividing by 2^k and then rounding toward zero.

k	>> k (binary)	Decimal	$-12{,}340/2^k$
0	1100111111001100	−12,340	−12,340.0
1	1110011111100110	−6,170	−6,170.0
4	1111110011111100	−772	−771.25
8	1111111111001111	−49	−48.203125

Two's complement

Figure 2.29 Applying arithmetic right shift. The examples illustrate that arithmetic right shift is similar to division by a power of 2, except that it rounds down rather than toward zero.

Division

- General case is expensive!
  - Special case: divide by powers of two (shift right)

k	Bias	−12,340 + bias (binary)	>> k (binary)	Decimal	$-12{,}340/2^k$
0	0	1100111111001100	1100111111001100	−12,340	−12,340.0
1	1	110011111100110*1*	*1*110011111100110	−6,170	−6,170.0
4	15	110011111101*1011*	*1111*110011111101	−771	−771.25
8	255	11010000*11001011*	*11111111*11010000	−48	−48.203125

Figure 2.30 Dividing two's-complement numbers by powers of 2. By adding a bias before the right shift, the result is rounded toward zero.

Quiz

- What is 5 + 2?
- What is 4 – 3?
- What is 2 << 3?
- What is 3 << 3?
- What is 16 >> 2?

Show your work using two’s complement in both hex and binary using 8-bit integers