Numbers & Arithmetic

CS 3410: Computer System Organization and Programming

[K. Bala, A. Bracy, E. Sirer, Z. Susag, and H. Weatherspoon]

Roadmap

Lecture #1

- Number representations

Lecture #2 (today!)

- Addition
- Negative numbers, two's complement
- Addition (two’s compliment)
- Overflow

Binary Addition

Addition works the same for all bases

- Add the digits in each position
- Propagate the carry

Binary addition is pretty easy

- Combine two bits at a time
- Along with a carry

```
  1
 183
+254
 437
```

Carry-in

```
  111   Carry-out
 001110
+011100
 101010
```

1st Try: Sign/Magnitude Representation

First Attempt: Sign/Magnitude Representation

- 1 bit for sign (0=positive, 1=negative)
- N-1 bits for magnitude

0111 = 7

1111 = -7

Problem?

- 2 zero's: +0 different than -0

0000 = +0

1000 = -0

- Complicated circuits
- -2 + 1 = ???

  [Prelim 1, FA19]

IBM 7090, 1959: "a second-generation transistorized version of the earlier IBM 709 vacuum tube mainframe computers"

Final Try: Two's Complement Representation

Positive numbers are represented as usual

- 0 = 0000, 1 = 0001, 3 = 0011, 7 = 0111

Leading 1's for negative numbers

To negate any number:

- complement *all* the bits (i.e. flip all the bits)
- then add 1
- -1: 1 ⇒ 0001 ⇒ 1110 ⇒ 1111
- -3: 3 ⇒ 0011 ⇒ 1100 ⇒ 1101
- -8: 8 ⇒ 1000 ⇒ 0111 ⇒ 1000
- -0: 0 ⇒ 0000 ⇒ 1111 ⇒ 0000 (this is good, -0 = +0)

Two's Complement

Non-negatives unchanged:

- +0 = 0000
- +1 = 0001
- +2 = 0010
- +3 = 0011
- +4 = 0100
- +5 = 0101
- +6 = 0110
- +7 = 0111
- +8 = 1000

How do you express this magnitude in the negative?

flip	Negatives then add 1
$\overline{0} = 1111$	-0 = 0000
$\overline{1} = 1110$	-1 = 1111
$\overline{2} = 1101$	-2 = 1110
$\overline{3} = 1100$	-3 = 1101
$\overline{4} = 1011$	-4 = 1100
$\overline{5} = 1010$	-5 = 1011
$\overline{6} = 1001$	-6 = 1010
$\overline{7} = 1000$	-7 = 1001
$\overline{8} = 0111$	-8 = 1000

Cornell Bowers CIS Computer Science

(8 no longer expressible in 4 bits when moving to signed)

Two's Complement vs. Unsigned

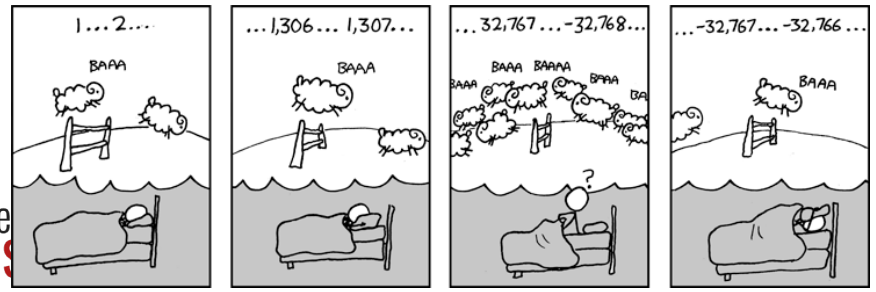
Another way to look at it: the MSB is a negative column (here -8)

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

-8 + 0 + 2 + 1

4 bit Two's Complement -8 … 7

4 bit Unsigned Binary 0 … 15

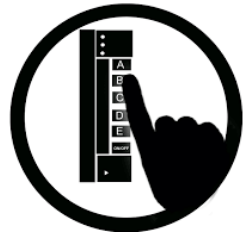


PollEV Question #1:

What is the value (expressed in decimal) of the 2s complement number

`11010`

A. 26
B. 6
C. -6
D. -10
E. -26



Two's Complement Facts

Signed two's complement

- Negative numbers have leading 1's
- zero is unique: +0 = - 0
- wraps from largest positive to largest negative

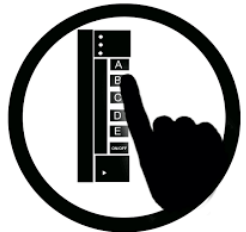
N bits can be used to represent

- unsigned: range $0 \ldots 2^N-1$
  - eg: 8 bits ⇒ 0…255
- signed (two's complement): $-(2^{N-1}) \ldots (2^{N-1} - 1)$
  - E.g.: 8 bits ⇒ (1000 0000) … (0111 1111)
  - -128 … 127

PollEV Question #2:

Suppose I want to express the 2s complement number `1010` in 5 bits instead of 4 bits. What number should I use?

A. 01010
B. 11010
C. 10101
D. 10100
E. Sorry, it is not possible.



Sign Extension & Truncation

Extending to larger size

- 1111 = -1
- 1111 1111 = -1
- 0111 = 7
- 0000 0111 = 7

Truncate to smaller size

- 0000 1111 = 15
- BUT, ~~0000~~ 1111 = 1111 = -1

Two's Complement Addition

Addition as usual. Ignore the sign.

It just works! Examples

1 + -1 =

-3 + -1 =

-7 + 3 =

7 + (-3) =

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

Overflow

When can **overflow** occur?

- adding a negative and a positive?
- adding two positives?
- adding two negatives?

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

Takeaways

- Digital computers are implemented via logic circuits and thus represent *all* numbers in binary (base 2).
- We use decimal or hex for convenience and need to be able to convert to binary and back.
- Adding two 1-bit numbers generalizes to adding two numbers of any size since 1-bit full adders can be cascaded.
- Using Two's complement number representation simplifies adder Logic circuit design (0 is unique, easy to negate).
- Subtraction is adding, where one operand is negated (to negate in two's complement: flip the bits and add 1).
- Overflow not enough bits for the result; i.e. if sign of input operands A and B are the same, but different than the of the result S.