Professor Hamlin
Day 3

## Agenda

1) Tutoring groups, HW 1
2) Review
3) Negative binary representation
   - two's complement

## Review

1. Modular Math
2. Non-base 10-math
3. Dec → Bin/Hex
   - Euclid, Subtraction

1) $35_{10}$ → Binary
(your choice of method)

| | |
|---|---|
| 0 | 0 |
| 2 | 1 |
| 4 | 2 |
| 8 | 3 |
| 16 | 4 |
| 32 | 5 |

$35 = 17 \cdot 2 + 1$
$17 = 8 \cdot 2 + 1$
$8 = 4 \cdot 2 + 0$
$4 = 2 \cdot 2 + 0$
$2 = 1 \cdot 2 + 0$
$1 = 0 \cdot 2 + 1$

$100011_2$

2) $(15+12) \bmod 5 = ?$

$15 \bmod 5 = 0$
$12 \bmod 5 = 2$

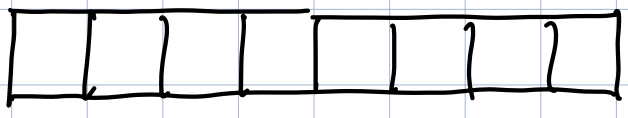
$0 + 2 \bmod 5 = \boxed{2}$

→ $15+12 = 27$
$27 \bmod 5 = 2$

$1 \cdot 100000_2 + 0 \cdot 10000_2 + 0 \cdot 1000_2 + 0 \cdot 100_2 + 1 \cdot 10_2 + 1 \cdot 1_2$

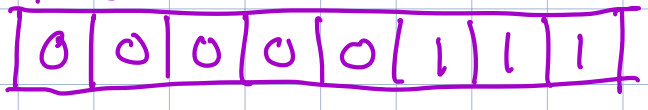
$35 - 32$ R 3 $\quad 3 \cdot 2 =$ R 1

Quick note on how computers store numbers. Remember Bytes? 8-bit



Computers store numbers using the same # of bits

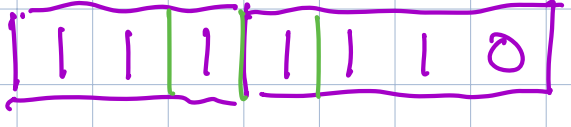
pad out w/ leading zeroes




$7 = 111_2$


$14 = 1110_2$

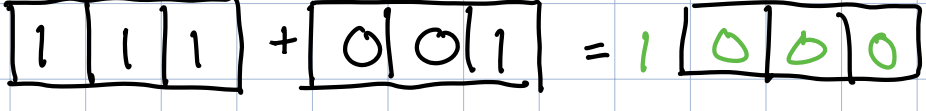
Sometimes 8 bits, 32, 64 etc. Why?




Computer doesn't know where things start or end


easier to find numbers

This means sometimes numbers are too big for their # of bits e.g w/ 3-bits




$7 + 1 = 8$

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ \ } \\ 1\ 1\ 1 \\ +\ 0\ 0\ 1 \\ \hline 1\ 0\ 0\ 0 \end{array}$$

This is kinda like working in mod 8!

# Negative Representation

In most math negative numbers represented by $-17$, $-456$

But computers only store 0/1, how to we indicate negative numbers?

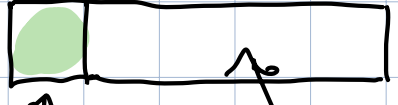
Vocab

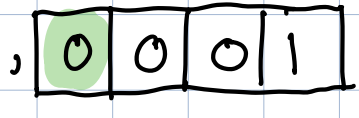
unsigned number: non-negative whole binary numbers e.g $101_2$

sign bit
↓
$-27$ ← magnitude

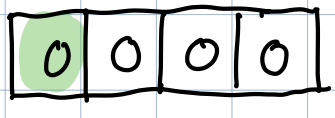
Starter idea! Just add a sign bit infront of number

1 = negative
0 = positive

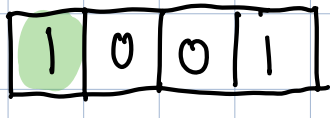


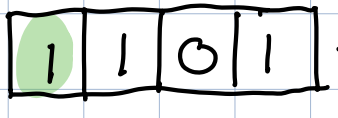

= 1




= 0




= −1

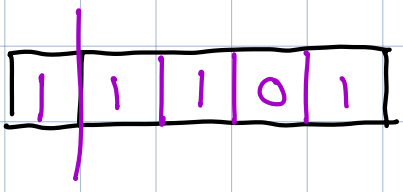



= −5

if unsigned $1001_2 = 9$
$(8+1)$

It's important to tell a computer how to read #

Ex] Write $-13$ in 5 bits




$13 = 6 \cdot 2 + 1$
$6 = 3 \cdot 2 + 0$
$3 = 1 \cdot 2 + 1$
$1 = 0 \cdot 2 + 1$

However just a sign bit has some problems....

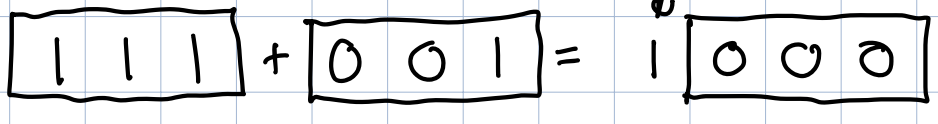
1) Two ways of representing zero
(4-bit example)

0 0 0 0 = +0

1 0 0 0 = −0

This means we aren't the most efficient!

2) Sign bit doesn't always add correctly!
(3-bit)




$$\begin{array}{r} {\scriptstyle 1\ 1\ 1} \\ +\ 0\ 0\ 1 \\ \hline 1\ 0\ 0\ 0 \end{array}$$

$-3$ $\quad 1 \quad =$ ~~$-2$~~ 0

(One's complement)

# Two's complement: a better negative rep.

Intition: most significant (biggest) place value is negative, everything else positive

e.g. Two's complement value in 3 bits

1 1 1

Two ways to think about this

1)

| -4 | 2 | 1 |
|---|---|---|
| 1 | 1 | 1 |

$-4+2+1=-1$

| -8 | 4 | 2 | 1 |
|---|---|---|---|
| 1 | 0 | 1 | 1 |

$-8\cdot1+0\cdot4+1\cdot2+1$

2) $-1\cdot100_2+1\cdot10_2+1\cdot1_2$

$-1\cdot2^2+1\cdot2+1=-1$

1011

$-1\cdot1000_2+0\cdot100_2+1\cdot10_2+1\cdot1_2$

$-1\cdot8+0\cdot4+1\cdot2\cdot1\cdot1$

Exercise:

1) 0 1 0 1 in 4-bit two's complement to decimal

| -8 | 4 | 2 | 1 |
|---|---|---|---|
| 0 | 1 | 0 | 1 |

$0\cdot-8+4\cdot1+0\cdot2+1\cdot1$

5

2) 1 1 0 1 in 4-bit two's to decimal

| -8 | 4 | 2 | 1 |
|---|---|---|---|
| 1 | 1 | 0 | 1 |

$-1\cdot-8+1\cdot4+2\cdot0+1\cdot1$

$-8+4+1$

-3

What pattern do we see (3-bits)

| -4 | 2 | 1 | |
|---|---|---|---|
| 1 | 0 | 0 | -4 |
| 1 | 0 | 1 | -3 |
| 1 | 1 | 0 | -2 |
| 1 | 1 | 1 | -1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |

1) only one zero :)

2) Math work? Yes

$-4, \rightarrow 3$

Checking math

$-2 + 3 = 1$

$110 + 011 = 001$

$$\begin{array}{r} 1 \\ 1\ 1\ 0 \\ +\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 1 \end{array}$$

Note: discarding digits isn't always bad: It only counts as overflow if the result is wrong

$011 + 001 = 100$

$3 + 1 = 4$

$[-4, 3]$

Overflow - when outcome cannot be rep. w/ # of bits

Connection to Mod

The magnitude is too big for 2 bits - like mod

2 bits → mod 4 ($2^2$)  things wrap around

Careful this is only true for positive!

Exercise: Which of the following values fit into their bits

1) 0 unsigned 2-bit Yes 00
2) -2 unsigned 3-bit No unsigned can't be neg
3) 0 2 bit two's comp. Yes 00
4) -4 3 bit two's comp Yes
5) -4 4 bit two's comp Yes
6) 5 4 bit two's comp Yes, 101 → 0101
7) 10 4 bit two's comp No, too big
8) -3 4 bit two's comp Yes

This leads us to ask what values can we represent w/ N-bits

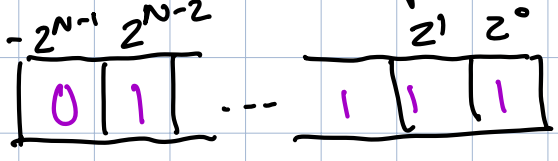
Unsigned


smallest - 0

largest - 111111....111

$1 \cdot 2^{N-1} + \ldots\ldots 1 \cdot 2^1 + 1 \cdot 2^0$

Two's complement




smallest = 1000....

$-1 \cdot 2^{N-1} + 0 \cdot 2^{N-2} + \ldots\ldots = -2^{N-1}$

largest = 01111...

$0 \cdot 2^{N-1} + 1 \cdot 2^{N-2} \ldots\ldots 1 \cdot 2^0 = 2^{N-1} - 1$

$2^N - 1$

5-digits
11111 → one less
100000

Exercise: With 5 bits can rep. $N=5$

1) unsigned
$0 \rightarrow 2^5 - 1$
$0 \rightarrow 31$

2) two's comp
$-2^{5-1} \rightarrow 2^{5-1} - 1$
$-16 \rightarrow 15$

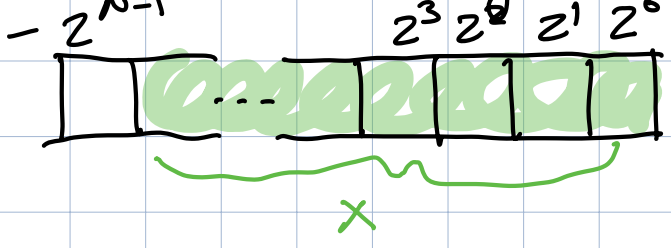
## Two's comp: Dec → Binary

1) First we have to check if it fits in the number of bits

e.g 4-bits (two's comp) Would 13 fit
$2^{4-1} - 1 = 2^3 - 1 = 7$

2) if value is positive – use Euclid or subtraction (remember to pad w/ leading zeros!)

3) if value is negative – solve for x




Value $= -2^{N-1} + x$

$x =$ value $+ -2^{N-1}$

a) value $= -2^{N-1} + x$
b) just solve for x
c) turn x into binary
d) append leading 1

Example: $-4$ as 4-bit two's comp.

find $x$: $-2^{4-1} + x = -4$

$-8 + x = -4$

$x = 4$

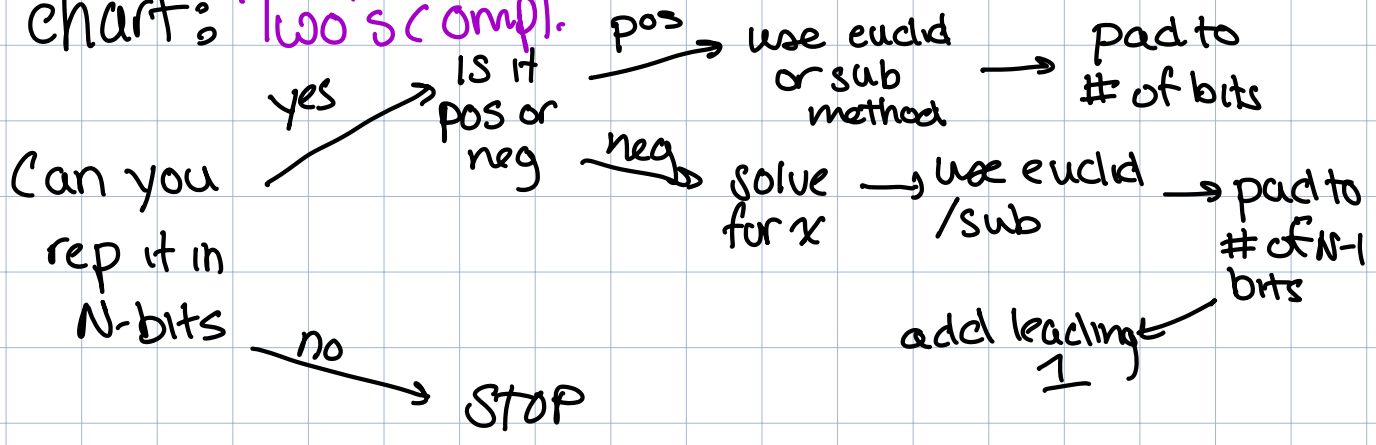
$x$ to binary:

$4 = 100_2$

add any leading zeroes: + leading 1

| 1 | 1 | 0 | 0 |
|---|---|---|---|

Flow chart: Two's compl.




Exercise: Represent following as 6-bit two's comp

1) $-5$

$-32 + x = -5$

$x = 27$

$27 \rightarrow 11011_2$

$111011_2$

2) 5

$5_{10} = 101_2$

$000101_2$

3) 32

6 bits

$-32$ to 31

$2^{6-1} - 1$

too big