Integers I

CSE 351 Winter 2021

Instructor:

Mark Wyse

Teaching Assistants:

Kyrie Dowling

Catherine Guevara

Ian Hsiao

Jim Limprasert

Armin Magness

Allie Pfleger

Cosmo Wang

Ronald Widjaja

Reading Review

- Terminology:
  - Unsigned integers
  - Signed integers (Two's Complement)

- Questions from the Reading?
  - about Unsigned and Signed Integers

Roadmap

C:

```
car *c = malloc(sizeof(car));
c->miles = 100;
c->gals = 17;
float mpg = get_mpg(c);
free(c);
```

Java:

```
Car c = new Car();
c.setMiles(100);
c.setGals(17);
float mpg =
    c.getMPG();
```

Memory & data
Integers & floats
x86 assembly
Procedures & stacks
Executables
Arrays & structs
Memory & caches
Processes
Virtual memory
Memory allocation
Java vs. C

Assembly language:

```
get_mpg:
    pushq   %rbp
    movq    %rsp, %rbp
    ...
    popq    %rbp
    ret
```

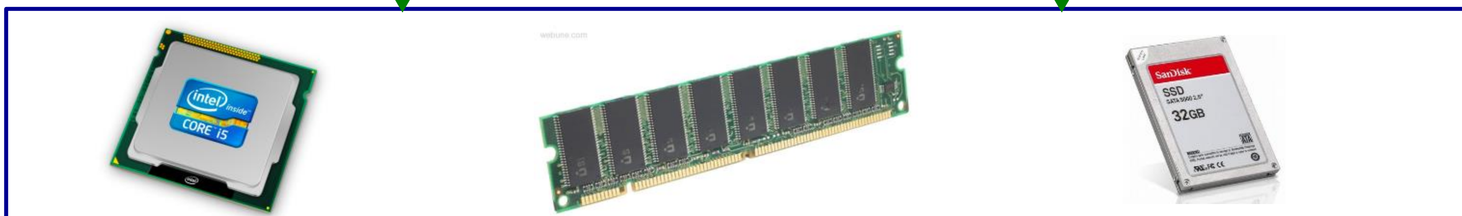
Machine code:

```
0111010000011000
100011010000010000000010
1000100111000010
110000011111101000011111
```

OS:

Windows 10 OS X Yosemite

Computer system:



Encoding Integers

- The hardware (and C) supports two flavors of integers
  - *unsigned* – only the non-negatives
  - *signed* – both negatives and non-negatives
- Cannot represent all integers with w bits
  - Only 2^w distinct bit patterns
  - Unsigned values: $0 \ldots 2^w-1$ → 0b1111.....11
  - Signed values: $-2^{w-1} \ldots 2^{w-1}-1$
- **Example:** 8-bit integers (*e.g.*, `char`)

-∞ ← → +∞

-128	0	$+128$	$+256$
-2^{8-1}	0	$+2^{8-1}$	$+2^{8}$

Unsigned Integers

- Unsigned values follow the standard base 2 system
  - $b_7b_6b_5b_4b_3b_2b_1b_0 = b_72^7 + b_62^6 + \cdots + b_12^1 + b_02^0$

$\rightarrow 2^{8-1} = 2^7$

- Useful formula: $2^{N-1} + 2^{N-2} + \ldots + 2 + 1 = 2^N - 1$
  - *i.e.*, N ones in a row = $2^N - 1$
  - *e.g.*, 0b111111 = 63 $\rightarrow$ 6 bits $\rightarrow 2^6 = 64$

$\rightarrow 2^6 - 1 = 63$

Sign and Magnitude

Not used in practice!

- Designate the high-order bit (MSB) as the "sign bit"
  - `sign=0`: positive numbers; `sign=1`: negative numbers
- Benefits:
  - Using MSB as sign bit matches positive numbers with unsigned
  - All zeros encoding is still = 0
- Examples (8 bits):
  - 0x00 = 00000000_2 is non-negative, because the sign bit is 0
  - 0x7F = 01111111_2 is non-negative ($+127_{10}$)
  - 0x85 = 10000101_2 is negative (-5_{10})
  - 0x80 = 10000000_2 is negative... zero???

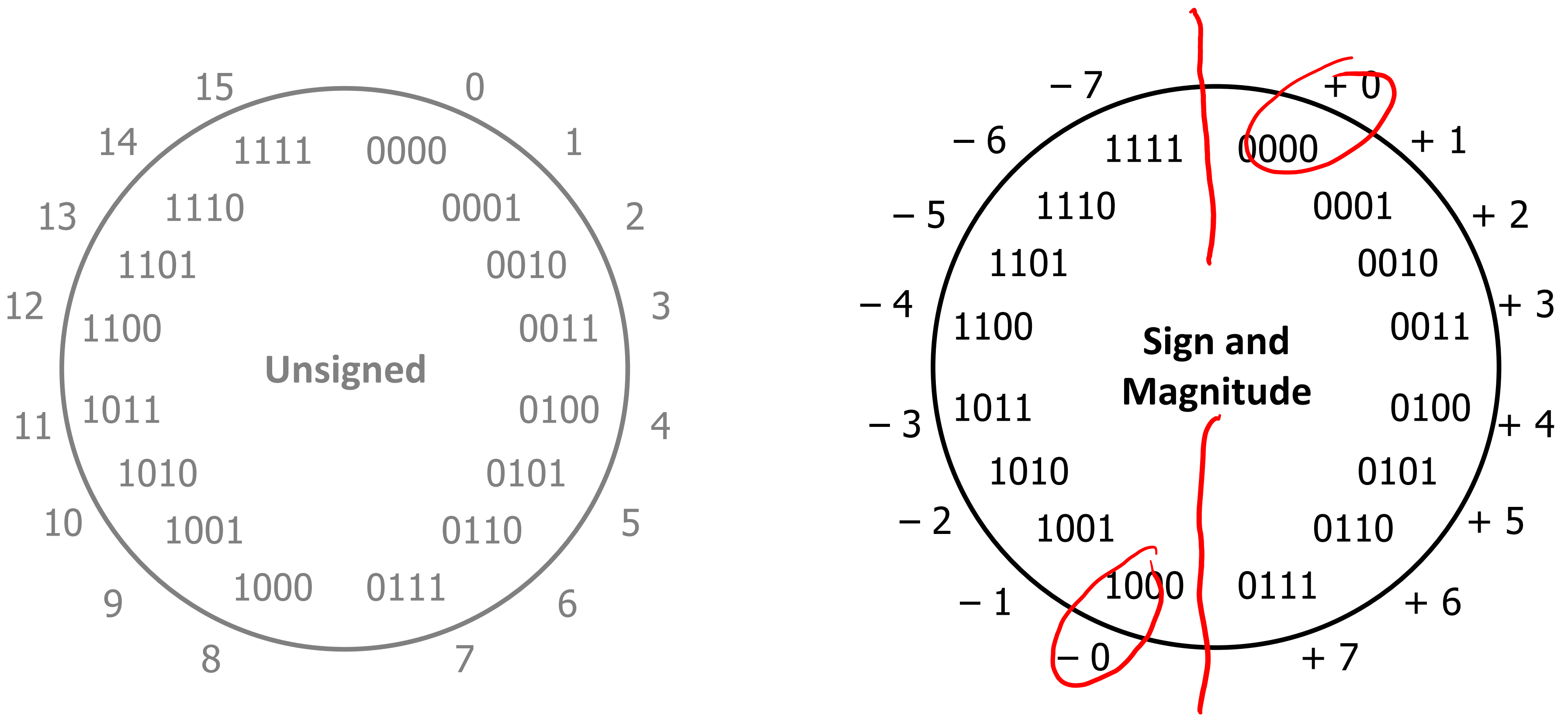
$\emptyset =$

$-\emptyset$

Sign and Magnitude

Not used in practice!

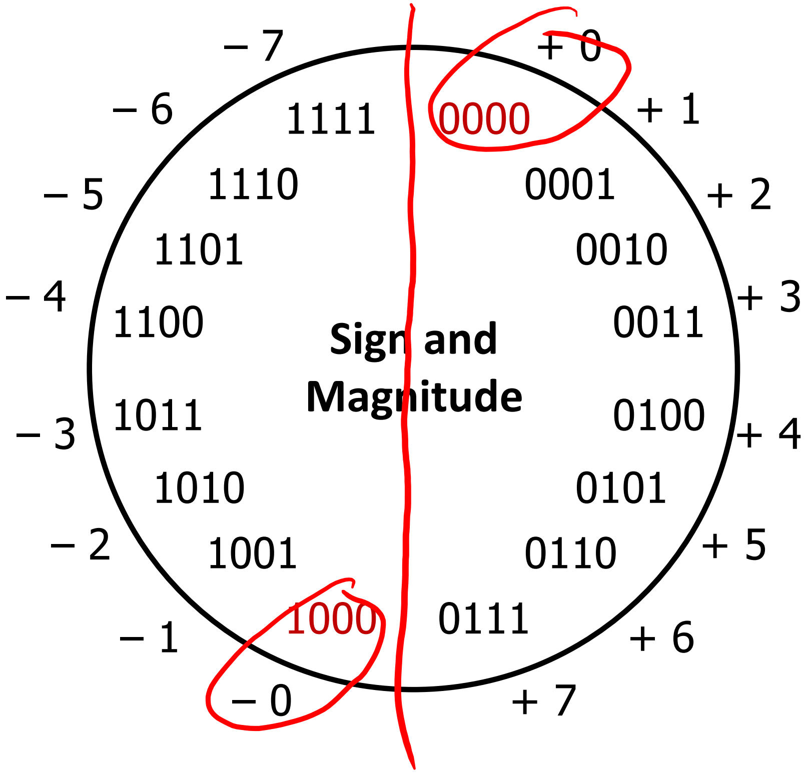
- MSB is the sign bit, rest of the bits are magnitude
- Drawbacks?



Sign and Magnitude

Not used in practice!

- MSB is the sign bit, rest of the bits are magnitude
- Drawbacks:
  - Two representations of 0 (bad for checking equality)



Sign and Magnitude

Not used in practice!

- MSB is the sign bit, rest of the bits are magnitude
- Drawbacks:
  - Two representations of 0 (bad for checking equality)
  - Arithmetic is cumbersome
    - Example: `4-3 != 4+(-3)`

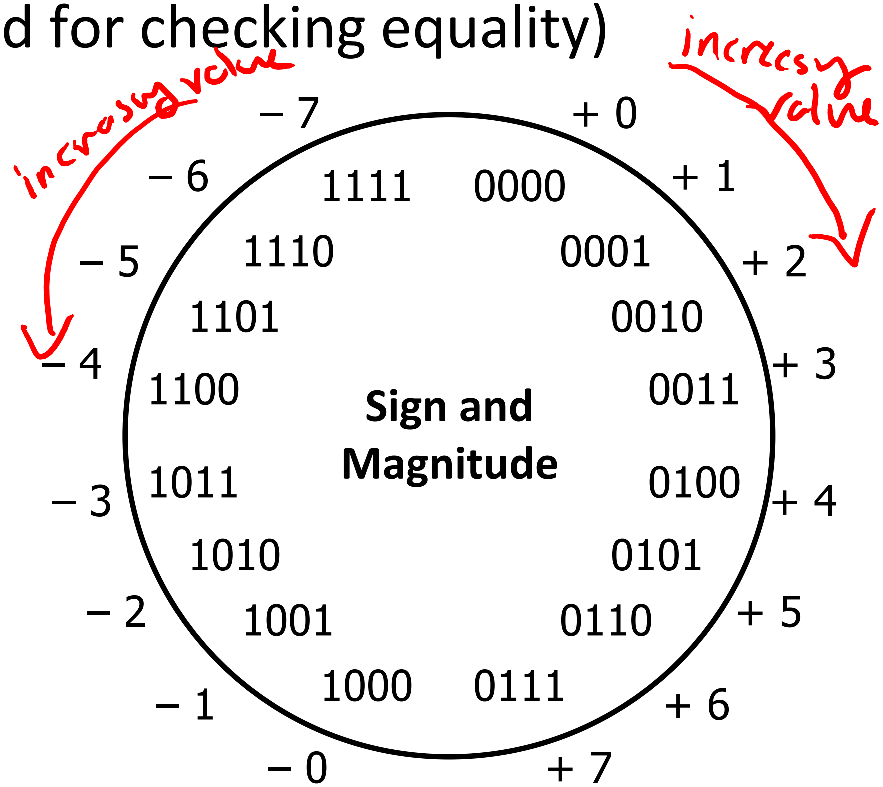
4	0100
– 3	– 0011
1	0001

✓

4	0100
+ -3	+ 1011
-7	1111

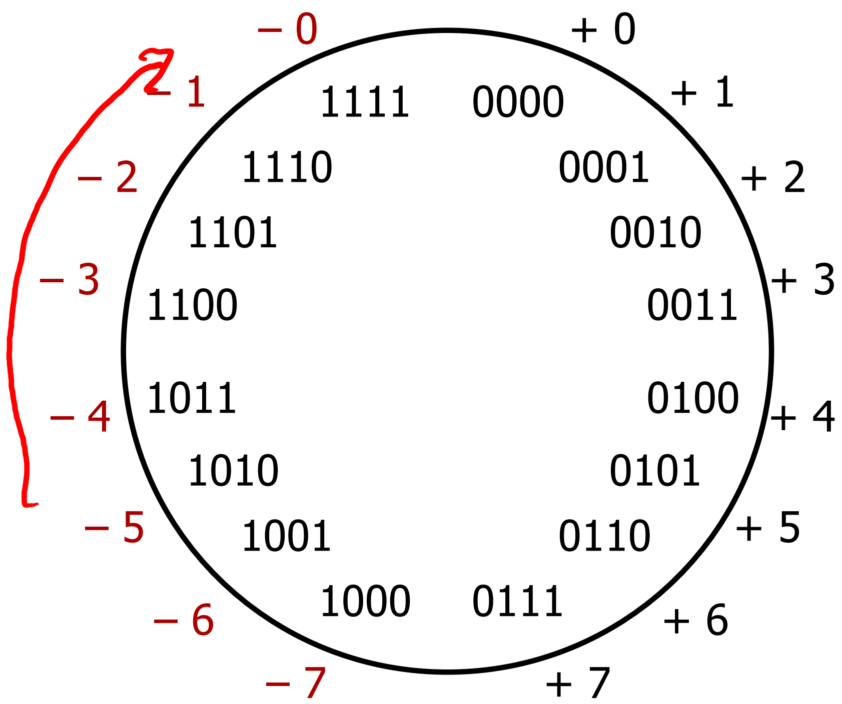
✗

    - Negatives "increment" in wrong direction!



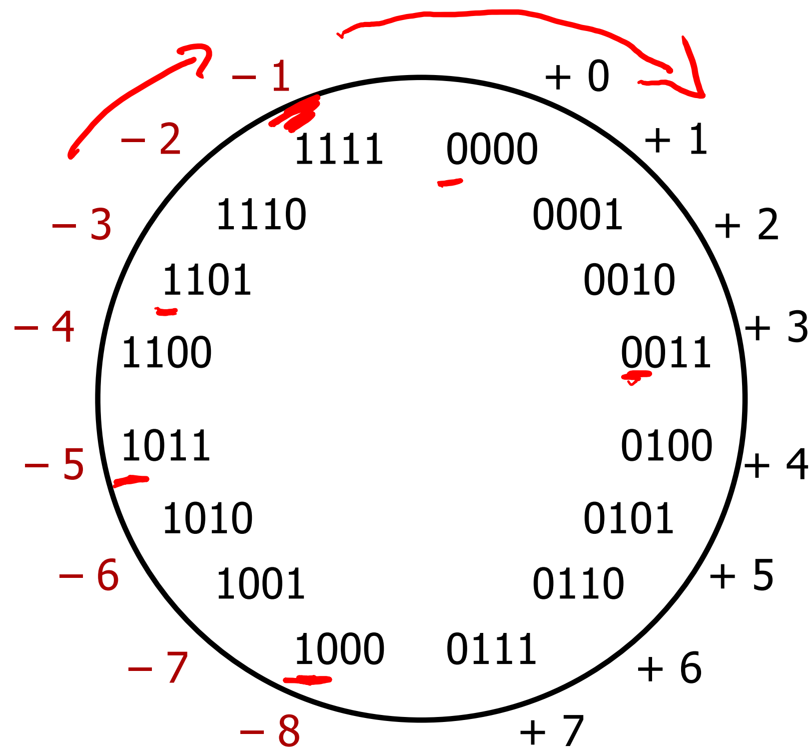
Two's Complement

- Let's fix these problems:
  1) "Flip" negative encodings so incrementing works



Two's Complement

- Let's fix these problems:
  1) "Flip" negative encodings so incrementing works
  2) "Shift" negative numbers to eliminate –0
- MSB *still* indicates sign!
  - This is why we represent one more negative than positive number (-2^{N-1} to 2^{N-1} –1)



Two's Complement Negatives

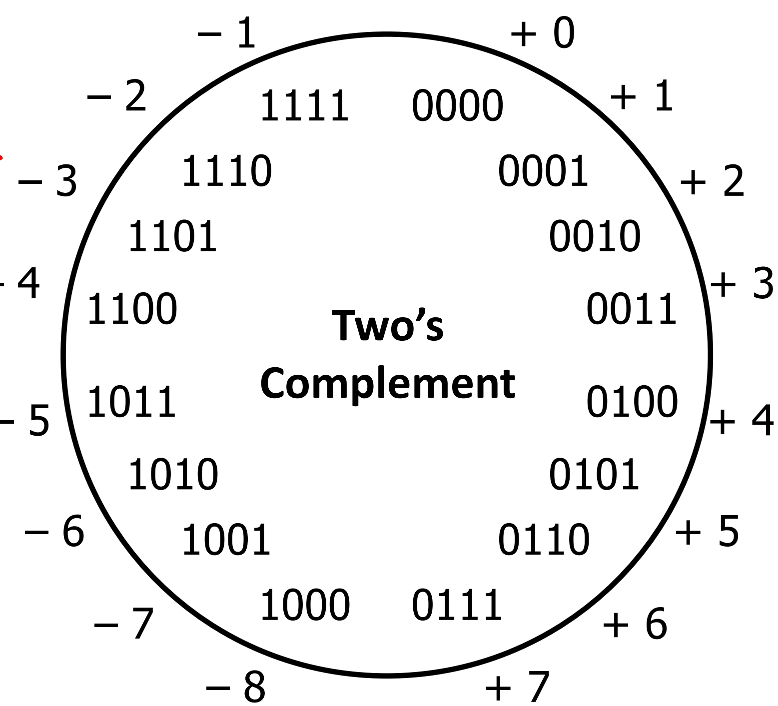
- Accomplished with one neat mathematical trick!

b_{w-1} has weight -2^{w-1}, other bits have usual weights $+2^i$

b_{w-1}	b_{w-2}	. . .	b_0

negative weight

- 4-bit Examples:
  - 1010_2 unsigned:
    $1*2^3+0*2^2+1*2^1+0*2^0$ = **10** = 8 + 2
  - 1010_2 two's complement:
    $-1*2^3+0*2^2+1*2^1+0*2^0$ = **–6** = -8+2
- -1 represented as:
  $1111_2 = -2^3+(2^3 - 1)$ from formula
  - MSB makes it super negative, add up all the other bits to get back up to -1



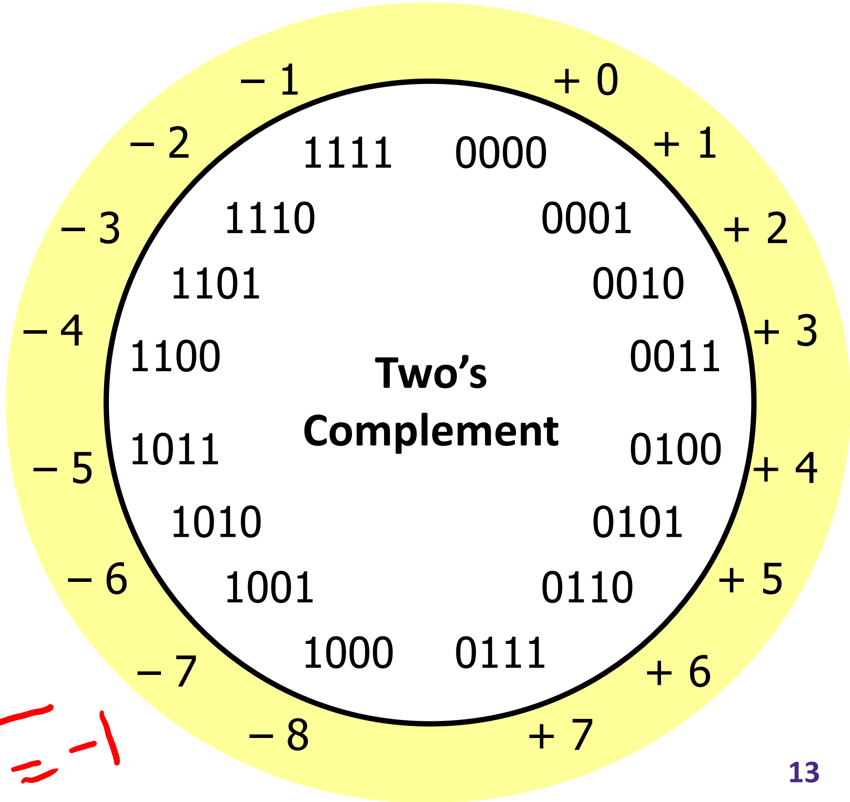
Why Two's Complement is So Great

- Roughly same number of (+) and (–) numbers
- Positive number encodings match unsigned
- Single zero
- All zeros encoding = 0

- Simple negation procedure:
  - Get negative representation of any integer by taking bitwise complement and then adding one!

```
( ~x + 1 == -x )
```

1= 0001
~ → 1110
\+ 1
1111 = -1



Review Question (Individual)

- Compute the value of `signed char sc = 0xF0;` (Two's Complement)

Polling Question

- Take the 4-bit number encoding **x = 0b1011**
- Which of the following numbers is NOT a valid interpretation of x using any of the number representation schemes discussed today?
  - Unsigned, Sign and Magnitude, Two's Complement
  - Vote in Ed Lessons

A. -4

✓ **B.** -5

✓ **C.** 11

✓ **D.** -3

E. We're lost…

unsigned → $x = 8+2+1 = 11$

sign + mag. $x = -(2+1) = -3$

two's compl. $x = -8 + 2+1 = -5$

Summary

- Integers represented using unsigned and two's complement representations
  - Limited by fixed bit width
  - Two's Complement encoding solves problems of Sign+Magnitude and aligns with Unsigned
  - We'll examine arithmetic operations next lecture