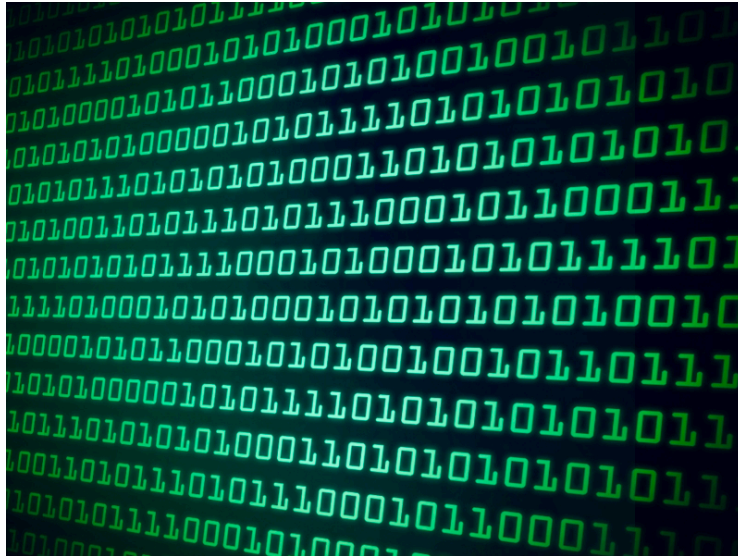


# Data Representation



## Typical Data Sizes

	<b>Data Type</b>	<b>Bytes</b>	
integer types	char	1	
	short	2	
	int	4	
	long	8	
	float	4	"floating-point" (fractional) types
	double	8	

# Numbering Systems

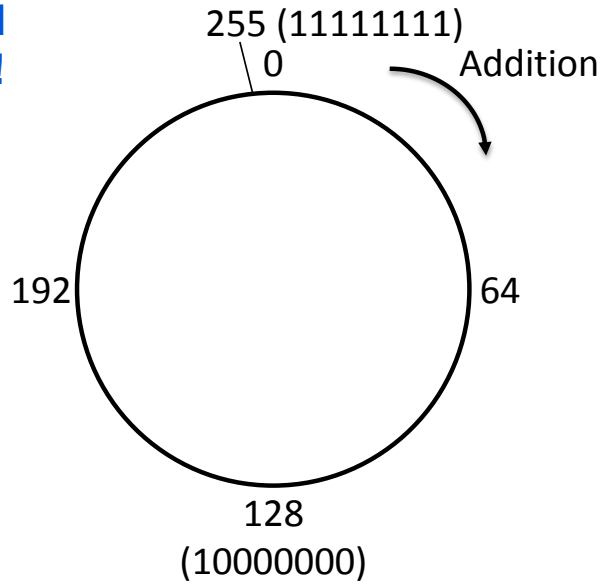
Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

# Binary Arithmetic

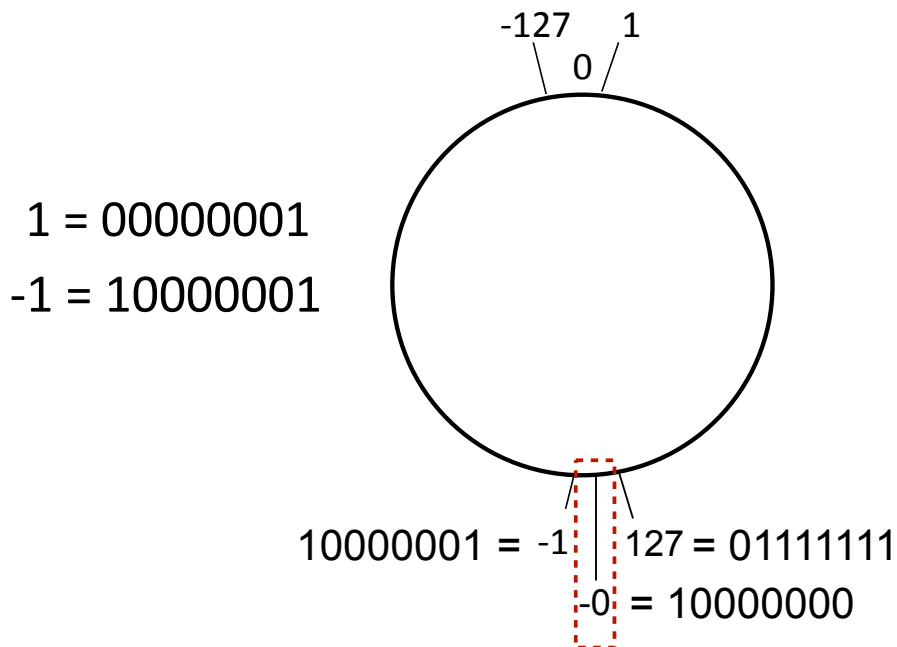
$$\begin{array}{r} 1 \\ 0110 \\ + 0100 \\ \hline 1010 \end{array} \quad \begin{array}{r} 6 \\ + 4 \\ \hline 10 \end{array}$$

# Modular Arithmetic

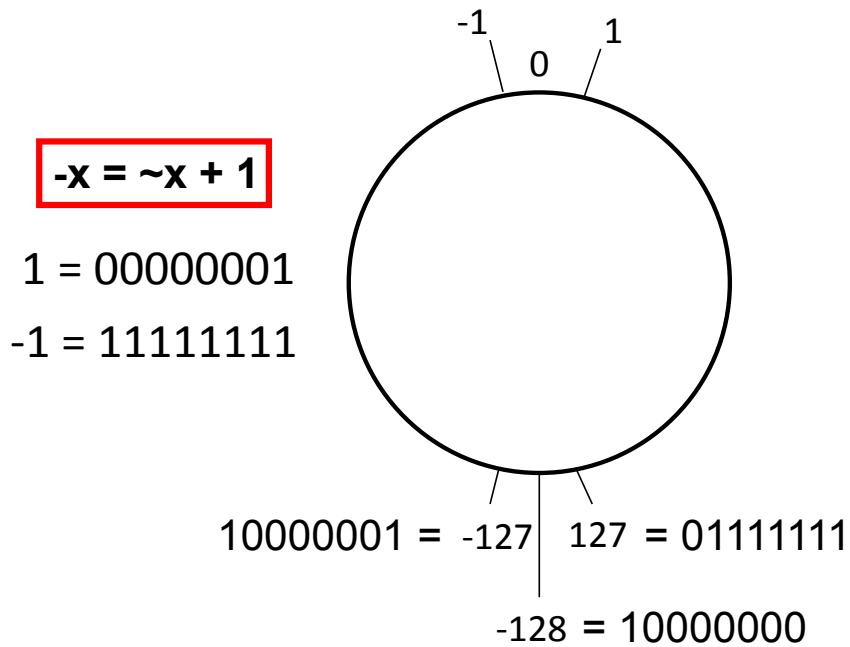
Unsigned numbers!



# Signed Magnitude



# Two's Complement



# Signed vs. Unsigned

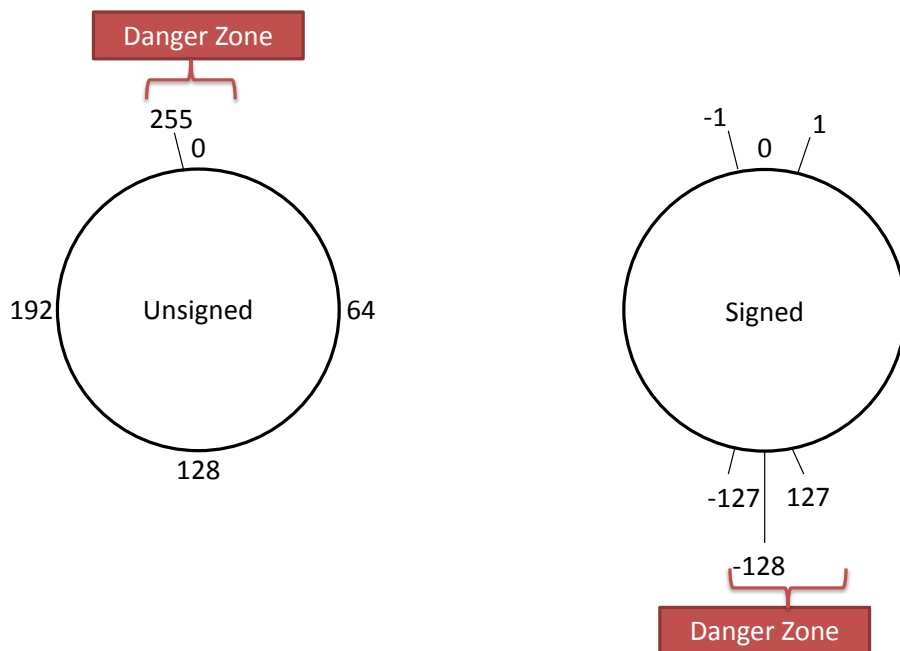
Bits	Signed		Unsigned
0000	0	↔ = ↔	0
0001	1		1
0010	2		2
0011	3		3
0100	4		4
0101	5		5
0110	6		6
0111	7		7
1000	-8	↔ +/- 16 ↔	8
1001	-7		9
1010	-6		10
1011	-5		11
1100	-4		12
1101	-3		13
1110	-2		14
1111	-1		15

# Sign Extension

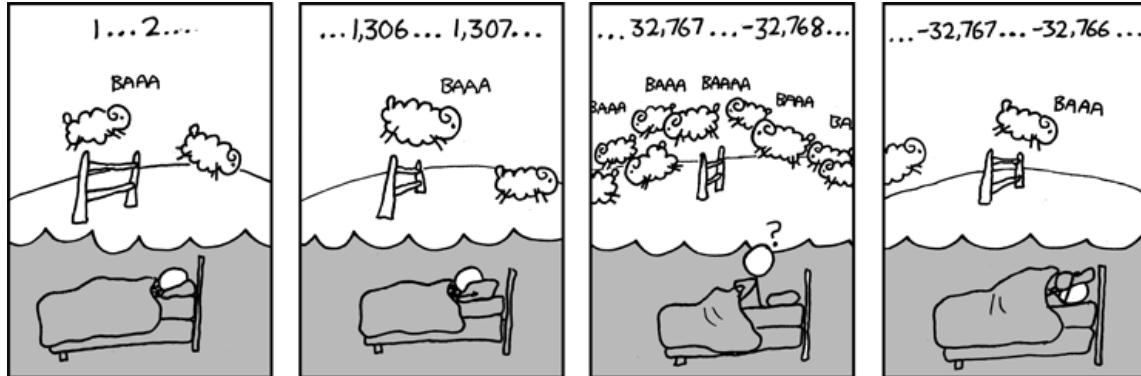
(5)     0101      $\rightarrow$      0000101

(-4)     1100      $\rightarrow$      11111100

# Overflow



# “Can’t Sleep”



(credit: xkcd.com)

# Unsigned Bugs...

```
float sum_elements(float a[], unsigned length) {  
    float sum = 0;  
    for (int i = 0; i <= length - 1; i++) {  
        sum += a[i];  
    }  
    return sum;  
}
```

# IEEE Floating Point (IEEE 754)

$$\text{value} = (-1)^s \times M \times 2^E$$

sign (s), significand (M), exponent (E)



len: 1    k = 8 or 11    23 or 52

## Normalized

$$E = \text{exp (unsigned)} - \text{bias}$$

$$\text{bias} = 2^{k-1} - 1$$

$$M = 0b1.\text{frac (binary)}$$

## Denormalized

when exp = 00...00

$$E = 1 - \text{bias}$$

$$M = 0b0.\text{frac (binary)}$$

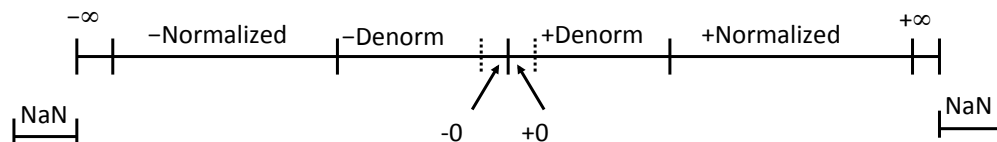
## Special

when exp = 11...11

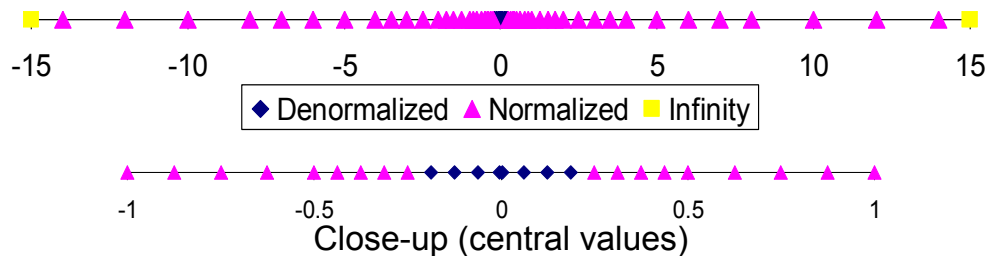
(a) frac = 00...00: **Infinity**

(b) frac  $\neq$  00...00: **NaN**

# Floating Point Visualization



6-bit values (3 exp, 2 frac)



# Representing Chars

## ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	