CS 211 RECITATIONS WEEK 6

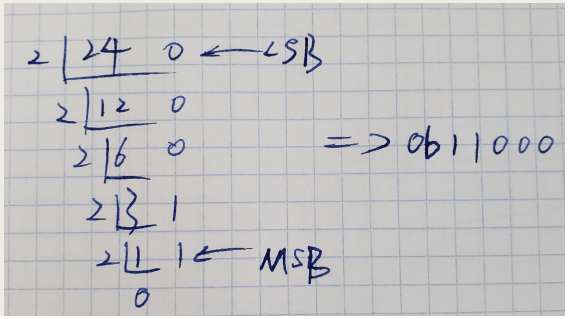
Wenjie Qiu
Teaching Assistant
Office Hour: Thu. noon – 1pm
wenjie.qiu@rutgers.edu

Content

- Number Systems
  - *decimal, binary, hexadecimal, octal*
  - *conversions*
- Representing Integers
  - *signed magnitude*
  - *one's complement*
  - *two's complement*
- Coding
  - *ASCII, Unicode and UTF-8*
- Floating Point Standard

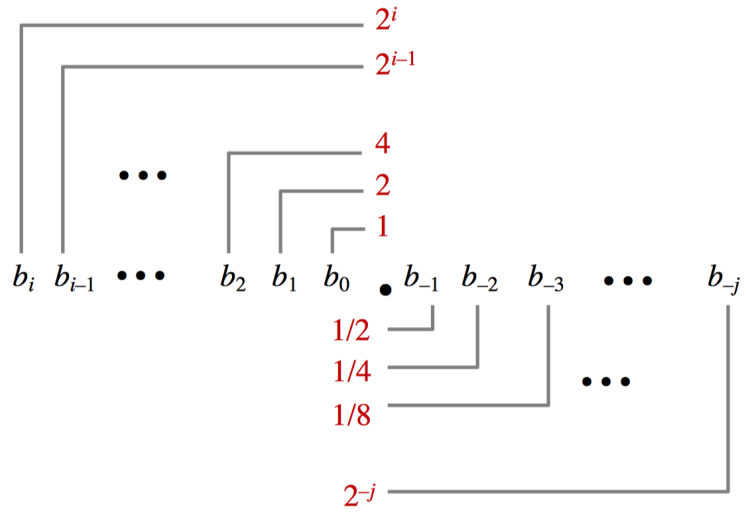
Number Systems

- Different Bases
  - *(10) Decimal (by default)*
  - *(2) Binary (start with 0b, 0b10011, GCC extension)*
  - *(16) Hexadecimal (start with 0x/0X, 0xABCD)*
  - *(8) Octal (start with 0, 012 in octal is 1*8 + 2*1 = 10 in decimal!)*
- Conversion Tips
  - *Hex to Binary: each digit in hex can be represented by FOUR bits*
  - *Octal to Binary: each digit in octal can be represented by THREE bits*
  - *Binary to Hex/Octal: treat each FOUR/THREE bits as a group*
  - *Decimal to Binary*
    - Memorize those bases, 1, 2, 4, 8, 16, 32, 64, etc.
    - Division



Number Systems

- Not just for integers
- 8.625 = 1000.101
  - *0.625 = 1/2 + 1/8 = $2^{-1} + 2^{-3}$=0.101*
  - *So 8.625 is exactly 1000.101 in binary*



- Representation
  - Bits to right of "binary point" represent fractional powers of 2
  - Represents rational number: $\sum_{k=-j}^{i} b_k \cdot 2^k$

Representing Integers

- Computers do not naturally know the concept of “positive/negative”, they store sequence of bits.

- Signed Magnitude
  - *Use the first bit to represent positive(0)/negative(1)*
- Problems
  - *Two zeros?*
  - *Positive + Positive is okay*
  - *Negative + Negative is okay*
  - *Negative + Positive cannot be calculated directly*

000	001	010	011	100	101	110	111
0	1	2	3	4	5	6	7

111	110	101	100	000	001	010	011
-3	-2	-1	-0	0	1	2	3

Representing Integers

- One's complement
  - *The first bit still represents positive (0)/negative(1)*
  - *Flip every bits and get the negative value*

- We still have two zeros!

- But 110 + 001 now can be calculated directly!
  - *110 + 001 = 111 (negative zero)*
  - *But this is correct anyway*

000	001	010	011	100	101	110	111
0	1	2	3	4	5	6	7

100	101	110	111	000	001	010	011
-3	-2	-1	-0	0	1	2	3

Representing Integers

- Two's complement
  - *One's complement plus one (in binary) in both directions*
  - *The first bit still represents positive(0)/negative(1)*

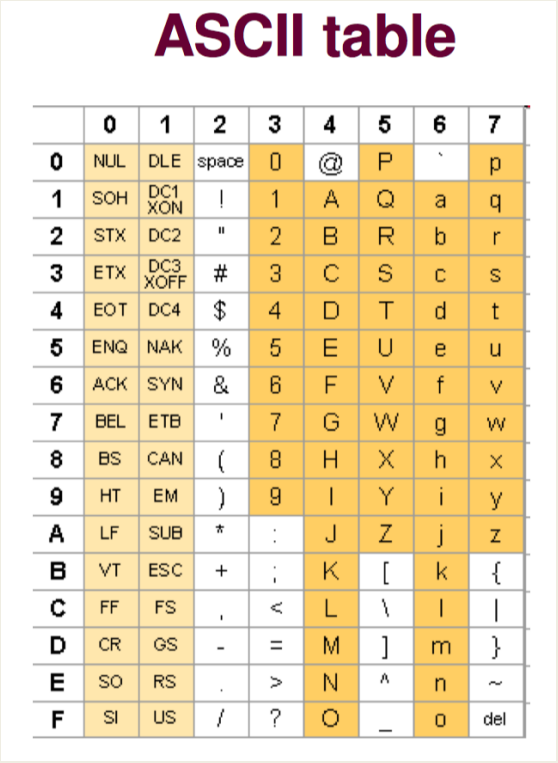
000	001	010	011	100	101	110	111
0	1	2	3	4	5	6	7

100	101	110	111	000	001	010	011
-4	-3	-2	-1	0	1	2	3

- One more negative number (only one zero!)
  - *Very convenient to use*
  - *Used in most computers today*

Coding

- ASCII
  - *A character is stored as one byte according to the ASCII standard.*
  - *Previously, 7 bits -> 128 characters, now 8 bits -> 256 characters.*
  - *char in C programming language.*



ASCII table

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

Coding

- Unicode is a standard that defines more than 100k characters in many languages.
- Unicode can be implemented by different character encodings.
- Most common: UTF-8
  - *Variable length: 1-4 Bytes for each character*
- Others:
  - *UFT-16*
  - *GBK*
  - *and many more*

Floating Point Standard

Numerical form:

$$V_{10} = (-1)^{s} * M * 2^{E}$$

- Sign bit ***s*** determines whether number is negative or positive
- Significand (mantissa) ***M*** normally a fractional value in range **[1.0, 2.0)**
- Exponent ***E*** weights value by a (possibly negative) power of two

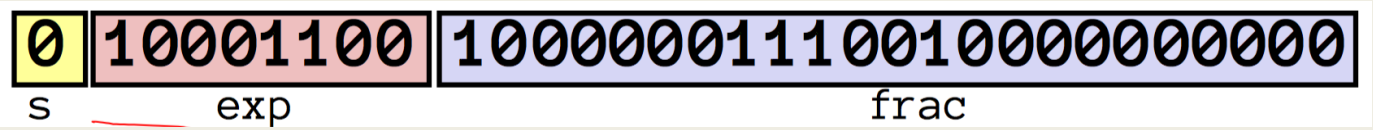
Representation in memory:

- MSB `s` is sign bit ***s***
- `exp` field encodes ***E*** (but is *not equal* to E)
- `frac` field encodes ***M*** (but is *not equal* to M)

Floating Point Standard

- 12345 in decimal is 11000000111001 in binary
  - *1.1000000111001 * 2^{13} (scientific notations in binary)*
- Mantissa
  - *M = 1. 1000000111001*
  - *frac = 1000000111001 followed by 0*
- Exponent
  - *E = 13 = exp – Bias*
  - *Bias = 2^{k-1} – 1 = (127 in 32 bits / 1023 in 64 bits)*
  - *exp = E + Bias = 140 in decimal, so 10001100 in binary*

$$V_{10} = (-1)^{S} * M * 2^{E}$$



Floating Point Standard

32 bits single precision (type float)

- 1 bit for sign, 8 bits for exponent, 23 bits for mantissa
  - Sign bit: 1 = negative numbers, 0 = positive numbers
  - Exponent is power of 2
- Have 2 zero's
- Range is approximately -10^{38} to 10^{38}

64 bits double precision (type double)

- 1 bit for sign, 11 bits for exponent, 52 bits for mantissa
- Majority of new bits for mantissa → higher precision
- Range is -10^{308} to $+10^{308}$