Quiz: ¼, show necessary solutions

- Base Conversion
  - Hex 30 to base 3
- How many digits does the representation of the value 64 need on the following:
  - Octal (base 8) number representation
- Subtraction using r's and (r-1)'s complement:
  - $(1001100 - 1010101)_2$
  - $(34009 - 2350)_{10}$

CMSC 130

Lecture 3 – Binary and Alphanumeric Codes, Floating Point Representation

Decimal Codes

Binary Representation of Decimal Numbers

Conversion vs Coding

- Conversion of a decimal number to a binary number and binary coding of a decimal number
  - both result to a series of bits
  - bits from conversion are binary digits
  - bits from coding are combinations of 1's and 0's in some arrangement based on the code used
- Be careful in handling series of 1's and 0's in a digital system

BCD

- A.k.a. binary-coded decimal
- It is a form of coding of a decimal number into binary.
- It is also a form direct conversion, but for values 0 to 9.
- Each of the digits of an unsigned decimal is represented as the 4-bit binary equivalent
- **Unpacked BCD** representation
- **Packed BCD** representation

Unpacked BCD

- This representation contains only one decimal digit per byte.
- The digit is stored in the 4 least significant bits
- The 4 most significant bits are not relevant to the value of the represented number

Examples,

- 7 → 0000 0111
- 11 → 0000 0001 0000 0001

Packed BCD

- This representation packs 2 decimal digits into a single byte.
- 4 bits are used to store each of the digits of the number.

Examples,

- $7 \longrightarrow 0111$
- $11 \longrightarrow 00010001$

Unpacked and Packed BCD

Number	Unpacked	Packed
1	00000001	0001
12	00000001 00000010	0001 0010
123	00000001 00000010 00000011	0001 0010 0011

BCD and Others

Decimal digit	BCD	Excess-3	8 4 -2 -1	2 4 2 1
0	0000	0011	0000	0000
1	0001	0100	0111	0001
2	0010	0101	0110	0010
3	0011	0110	0101	0011
4	0100	0111	0100	0100
5	0101	1000	1011	1011
6	0110	1001	1010	1100
7	0111	1010	1001	1101
8	1000	1011	1000	1110
9	1001	1100	1111	1111

Alphanumeric Codes

ASCII

- American Standard Code for Information Interchange
- It is the standard binary code for alphanumeric characters.
- It uses 7 bits to code **128 characters**
  - Contains **94** printable characters
    - 26 uppercase letters (A – Z), 26 lowercase letters (a – z), 10 numerals (0 – 9), 32 special characters (such as %, *, $)
  - Contains 34 nonprintable characters used for various control functions

ASCII

b3 b2 b1 b0	b6 b5 b4							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Example

- "HI!" $= (0100\ 1000\ 0100\ 1001\ 0010\ 0001)_2$
  $= (48\ 49\ 21)_{16}$

- *NOTE: The b8 position is reserved for error-detection code (either odd or even parity), but in our example, we don't have an error-detection code applied.*

Error-Detection Code

Detection of Bit-Reversal Errors

Error-Detection Code

- Purpose is to detect bit-reversal errors during transmission of binary information
- One of the most common error detection is by means of a parity bit
  - An extra bit included with a message to make the total number of 1's transmitted either odd or even

Error-Detection Code

Character	ASCII (7 bits)	With Even Parity ((7+1) bits)	With Odd Parity ((7+1) bits)
A	100 0001	0100 0001	1100 0001
T	101 0100	1101 0100	0101 0100
9	011 1001	0011 1001	1011 1001
+	010 1011	0010 1011	1010 1011

Error-Detection Code

Message	Parity (P)		Message	Parity (P)	
	Odd	Even		Odd	Even
0000	1	0	1000	0	1
0001	0	1	1001	1	0
0010	0	1	1010	1	0
0011	1	0	1011	0	1
0100	0	1	1100	1	0
0101	1	0	1101	0	1
0110	1	0	1110	0	1
0111	0	1	1111	1	0

Floating Point Representation

Floating Point Representation

- Represents reals in scientific notation
- Addresses some problems in real number representation
  - limit on the range of values – not too large, not too small
  - loss of precision when large numbers are divided

IEEE Standard 754 Floating Point

- Most common representation for real numbers on computers
- IEEE floating point numbers have 3 basic components,
  - the sign
  - the exponent
  - the mantissa

IEEE Standard 754 Floating Point

Precision	Sign	Exponent	Fraction	Bias
Single Precision	1 [31]	8 [30-23]	23 [22-00]	127
Double Precision	1	11	52	1023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	e	e	e	e	e	e	e	e	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f

IEEE Standard 754 Floating Point

- The Sign Bit
  - 0 denotes a positive number
  - 1 denotes a negative number
- The Exponent
  - A bias is added to the actual exponent in order to get the stored exponent
    - This is to represent both positive and negative exponents
  - Single precision: exponent field is 8 bits, bias is 127
  - Double precision: exponent field is 11 bits, bias is 1023

IEEE Standard 754 Floating Point

- The Mantissa – also known as significand
  - Represents precision bits of the number
  - Composed of an implicit leading bit and the fractions bits
- Normalized Form
  - In order to maximize the quantity of representable numbers
  - Basically placing the radix point after the first nonzero digit (from the left)

IEEE Standard 754 Floating Point

- Further,
  - Sign bit: 0-positive, 1-negative
  - Exponent's base is 2
  - Exponent field contains **127 (bias) + true exponent** for single precision, **1023 (bias) + true exponent** for double-precision
  - First bit of the mantissa is typically assumed to be **1.*f***, where ***f*** is the field for fraction bits

IEEE Standard 754 Floating Point

- Examples (single precision):

1. $(10110.11)_2$
2. $(-101.1011101)_2$
3. $(0.00101001)_2$
4. $(31.125)_{10}$
5. $(-4.25)_{10}$