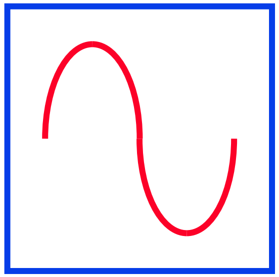
CHAPTER 1: BINARY SYSTEMS

- DIGITAL COMPUTER & DIGITAL SYSTEMS
- BINARY NUMBERS
- NUMBER BASE CONVERSION
- COMPLEMENTS
- SIGNED BINARY NUMBERS
- BINARY CODES
- BINARY STORAGE ELEMENTS

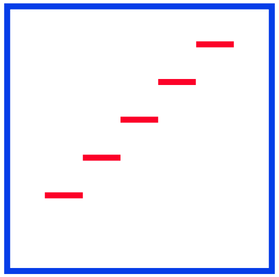
Digital Systems

Discrete Data

- Examples:
  - 26 letters of the alphabet (A, B … etc)
  - 10 decimal digits (0, 1, 2 … etc)
- Combine together
  - Words are made of letters (University … etc)
  - Numbers are made of digits (4241 … etc)



Analog



Discrete

Binary System

- Only ‘0’ and ‘1’ digits
- Can be easily implemented in electronic circuits

Decimal Number System

★ **Base (also called radix) = 10**

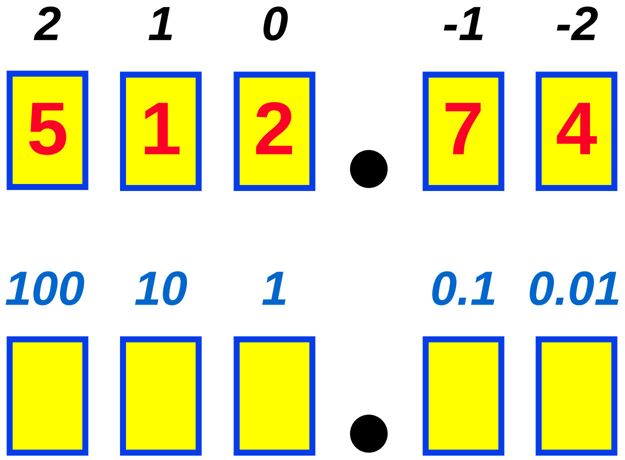
- 10 digits { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

★ **Digit Position**

- Integer & fraction

★ **Digit Weight**

- Weight = (*Base*) $^{\textit{Position}}$



★ **Magnitude**

500 10 2 0.7 0.04

- Sum of "*Digit* x *Weight*"

$d_2*B^2+d_1*B^1+d_0*B^0+d_{-1}*B^{-1}+d_{-2}*B^{-2}$

★ **Formal Notation**

$(512.74)_{10}$

Octal Number System

★ **Base = 8**

- 8 digits { 0, 1, 2, 3, 4, 5, 6, 7 }

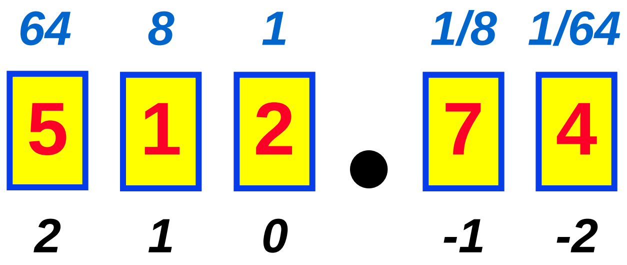
★ **Weights**

- Weight = (*Base*) $^{\textit{Position}}$

★ **Magnitude**

- Sum of "*Digit* x *Weight*"

★ **Formal Notation**



$$5 *8^{2}+1 *8^{1}+2 *8^{0}+7 *8^{-1}+4 *8^{-2}$$

$$=(330.9375)_{10}$$

$$(512.74)_{8}$$

Binary Number System

★ Base = 2

- 2 digits { 0, 1 }, called ***b***inary dig***its*** or "***bits***"

★ Weights

- Weight = (*Base*) $^{\textit{Position}}$

4	*2*	*1*		*1/2*	*1/4*
1	0	1	●	0	1
2	*1*	*0*		*-1*	*-2*

★ Magnitude

- Sum of "*Bit* x *Weight*"

$$1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2}$$

$$= (5.25)_{10}$$

★ Formal Notation

$$(101.01)_2$$

★ Groups of bits

4 bits = *Nibble* 1 0 1 1

8 bits = *Byte* 1 1 0 0 0 1 0 1

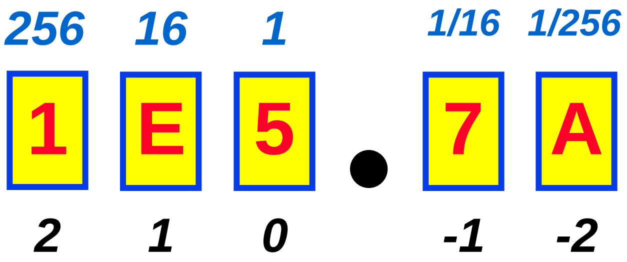
Hexadecimal Number System

★ **Base = 16**

- **16 digits { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }**

★ **Weights**

- **Weight = (*Base*)** $^{\textit{Position}}$



★ **Magnitude**

- **Sum of “*Digit* x *Weight*”**

$$1 * 16^{2} + 14 * 16^{1} + 5 * 16^{0} + 7 * 16^{-1} + 10 * 16^{-2}$$

$$= (485.4765625)_{10}$$

★ **Formal Notation**

$$(1E5.7A)_{16}$$

The Power of 2

n	2^n
0	$2^0=1$
1	$2^1=2$
2	$2^2=4$
3	$2^3=8$
4	$2^4=16$
5	$2^5=32$
6	$2^6=64$
7	$2^7=128$

n	2^n	
8	$2^8=256$	
9	$2^9=512$	
10	$2^{10}=1024$	Kilo
11	$2^{11}=2048$	
12	$2^{12}=4096$	
20	$2^{20}=1M$	Mega
30	$2^{30}=1G$	Giga
40	$2^{40}=1T$	Tera

Addition

★ Decimal Addition

$$\begin{array}{rccc} & 1 & 1 & \\ & & 5 & 5 \\ + & & 5 & 5 \\ \hline & 1 & 1 & 0 \end{array}$$

← **Carry**

= ***Ten*** $\geq$ ***Base***

➔ **Subtract a Base**

Binary Addition

★ Column Addition

$$\begin{array}{cccccccccc}
 & 1 & 1 & 1 & 1 & 1 & 1 & & & \\
 & & 1 & 1 & 1 & 1 & 0 & 1 & & = 61 \\
+ & & & 1 & 0 & 1 & 1 & 1 & & = 23 \\
\hline
 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & & = 84
\end{array}$$

$$\searrow \geq (2)_{10}$$

Binary Subtraction

★ **Borrow a "Base" when needed**

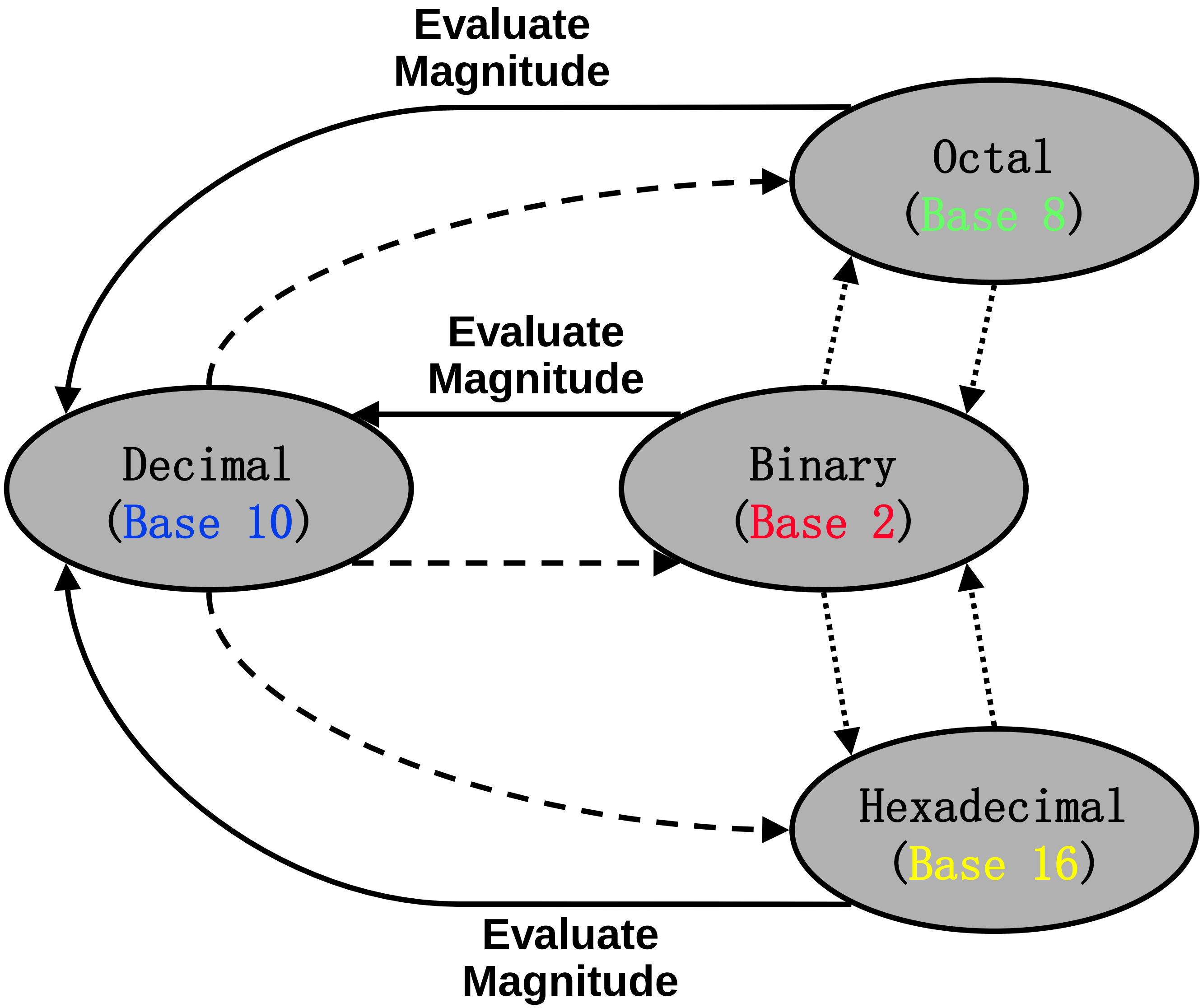
$$
\begin{array}{cccccccc l}
 & & 1 & & & 2 & & & = (10)_2 \\
 & 0 & \cancel{2} & 2 & 0 & 0 & 2 & & \\
 & \cancel{1} & 0 & 0 & \cancel{1} & \cancel{1} & 0 & 1 & = 77 \\
- & & & 1 & 0 & 1 & 1 & 1 & = 23 \\
\hline
 & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & = 54
\end{array}
$$

Binary Multiplication

★ **Bit by bit**

```
                1   0   1   1   1
x                   1   0   1   0
---------------------------------
                0   0   0   0   0
            1   0   1   1   1
        0   0   0   0   0
    1   0   1   1   1
---------------------------------
    1   1   1   0   0   1   1   0
```

Number Base Conversions



Decimal (*Integer*) to Binary Conversion

★ Divide the number by the ‘Base’ (=2)

★ Take the remainder (either 0 or 1) as a coefficient

★ Take the quotient and repeat the division

Example: $(13)_{10}$

	Quotient	Remainder	Coefficient
13 / 2 =	6	1	$a_0 = 1$
6 / 2 =	3	0	$a_1 = 0$
3 / 2 =	1	1	$a_2 = 1$
1 / 2 =	0	1	$a_3 = 1$

Answer: $(13)_{10} = (a_3\, a_2\, a_1\, a_0)_2 = (1101)_2$

MSB LSB

Decimal (*Fraction*) to Binary Conversion

★ Multiply the number by the ‘Base’ (=2)

★ Take the integer (either 0 or 1) as a coefficient

★ Take the resultant fraction and repeat the division

Example: $(0.625)_{10}$

		Integer		Fraction	Coefficient
0.625	* 2 =	1	.	25	$a_{-1} = 1$
0.25	* 2 =	0	.	5	$a_{-2} = 0$
0.5	* 2 =	1	.	0	$a_{-3} = 1$

Answer: $(0.625)_{10} = (0.a_{-1}\, a_{-2}\, a_{-3})_2 = (0.101)_2$

MSB **LSB**

Decimal to Octal Conversion

Example: $(175)_{10}$

	Quotient	Remainder	Coefficient
175 / 8 =	21	7	$a_0 = 7$
21 / 8 =	2	5	$a_1 = 5$
2 / 8 =	0	2	$a_2 = 2$

Answer: $(175)_{10} = (a_2\ a_1\ a_0)_8 = (257)_8$

Example: $(0.3125)_{10}$

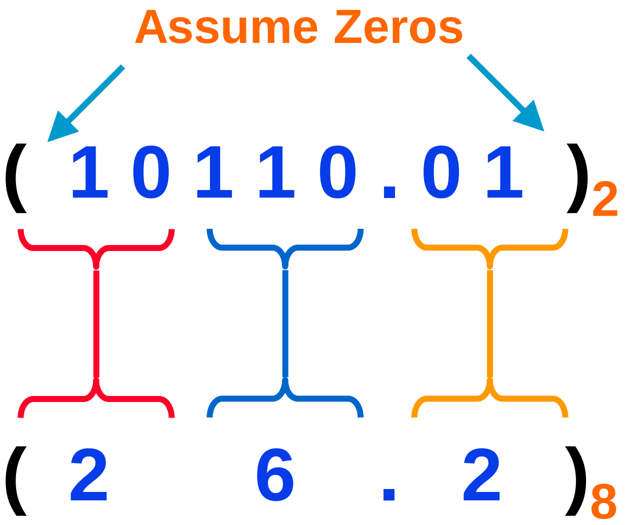
	Integer		Fraction	Coefficient
0.3125 * 8 =	2	.	5	$a_{-1} = 2$
0.5 * 8 =	4	.	0	$a_{-2} = 4$

Answer: $(0.3125)_{10} = (0.a_{-1}\ a_{-2}\ a_{-3})_8 = (0.24)_8$

Binary − Octal Conversion

- ★ $8 = 2^3$
- ★ Each group of 3 bits represents an octal digit

Example:



Assume Zeros

$(\; 1\,0\,1\,1\,0\,.\,0\,1 \;)_2$

$(\; 2 \quad 6 \; . \; 2 \;)_8$

Octal	Binary
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

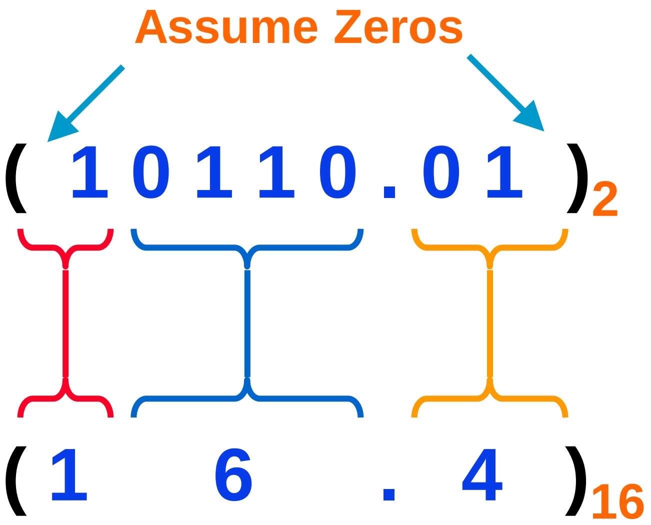
Works both ways (*Binary* to *Octal* & *Octal* to *Binary*)

Binary – Hexadecimal Conversion

★ 16 = 2^4

★ Each group of 4 bits represents a hexadecimal digit

Example:



Assume Zeros

$(\ 1\,0\,1\,1\,0\,.\,0\,1\ )_2$

$(1\quad 6\quad .\quad 4)_{16}$

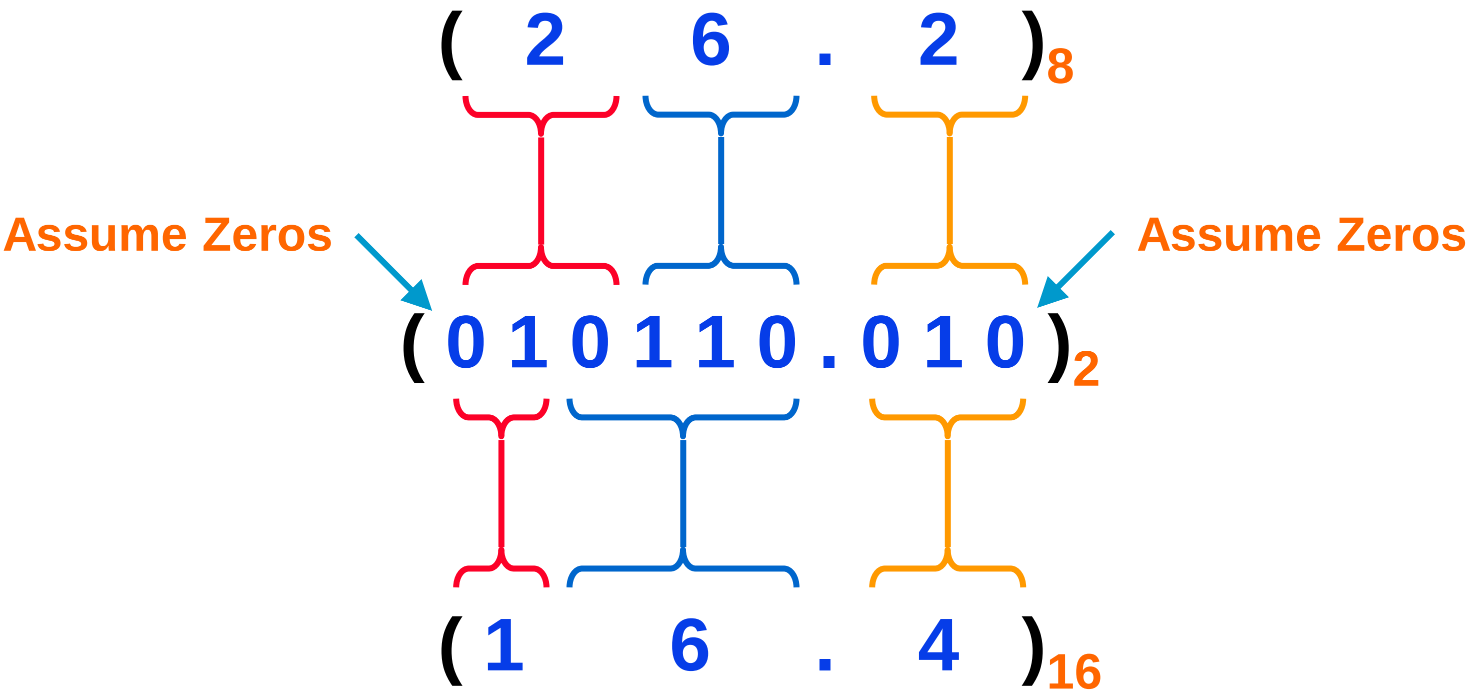
Hex	Binary
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

Works both ways (*Binary* to *Hex* & *Hex* to *Binary*)

Octal − Hexadecimal Conversion

★ **Convert to Binary as an intermediate step**

Example:



$(\,2\;\;6\;.\;2\,)_8$

Assume Zeros

Assume Zeros

$(\,0\,1\,0\,1\,1\,0\,.\,0\,1\,0\,)_2$

$(\,1\;\;6\;.\;4\,)_{16}$

Works both ways (*Octal* to *Hex* & *Hex* to *Octal*)

Decimal, Binary, Octal and Hexadecimal

Decimal	Binary	Octal	Hex
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Complements

★ 1's Complement (*Diminished Radix* Complement)

- All '0's become '1's
- All '1's become '0's

Example $(10110000)_2$

⇨ $(01001111)_2$

If you add a number and its 1's complement …

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

Complements

★ 2's Complement (*Radix* Complement)

- Take 1's complement then add 1

OR

- Toggle all bits to the left of the first '1' from the right

Example:

Number:	1 0 1 1 0 0 0 0	1 0 1 1 0 0 0 0
1's Comp.:	0 1 0 0 1 1 1 1	
+	1	
	0 1 0 1 0 0 0 0	0 1 0 1 0 0 0 0

Negative Numbers

★ **Computers Represent Information in ‘0’s and ‘1’s**

- **‘+’ and ‘−’ signs have to be represented in ‘0’s and ‘1’s**

★ **3 Systems**

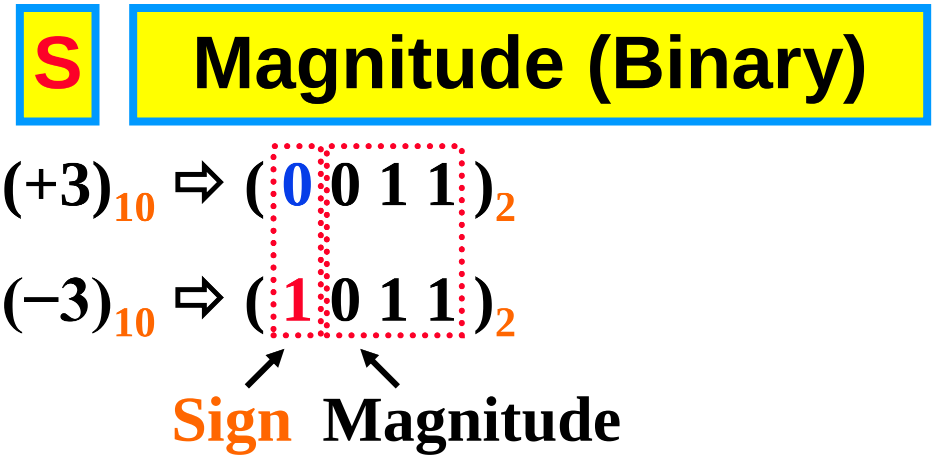
- **Signed Magnitude**
- **1’s Complement**
- **2’s Complement**

All three use the *left-most bit* to represent the sign:

- ♦ **‘0’ ⇨ positive**
- ♦ **‘1’ ⇨ negative**

Signed Magnitude Representation

★ Magnitude is magnitude, *does not change with sign*



S

Magnitude (Binary)

$(+3)_{10}$ ⇨ $(0\ 0\ 1\ 1)_2$

$(-3)_{10}$ ⇨ $(1\ 0\ 1\ 1)_2$

Sign Magnitude

★ Can’t include the *sign bit* in ‘Addition’

$$\begin{array}{rl} 0\ 0\ 1\ 1 & ⇨ (+3)_{10} \\ +\quad 1\ 0\ 1\ 1 & ⇨ (-3)_{10} \\ \hline 1\ 1\ 1\ 0 & ⇨ (-6)_{10} \end{array}$$

1’s Complement Representation

★ Positive numbers are represented in “Binary”

0	Magnitude (Binary)

★ Negative numbers are represented in “1’s Comp.”

1	Code (1’s Comp.)

$(+3)_{10} \Rightarrow (0\ 011)_{2}$

$(-3)_{10} \Rightarrow (1\ 100)_{2}$

★ There are 2 representations for ‘0’

$(+0)_{10} \Rightarrow (0\ 000)_{2}$

$(-0)_{10} \Rightarrow (1\ 111)_{2}$

1’s Complement Range

★ **4-Bit Representation**

$2^4 = 16$ Combinations

$-7 \le \text{Number} \le +7$

$-2^3+1 \le \text{Number} \le +2^3-1$

★ **n-Bit Representation**

$-2^{n-1}+1 \le \text{Number} \le +2^{n-1}-1$

Decimal	1’s Comp.
+ 7	0 1 1 1
+ 6	0 1 1 0
+ 5	0 1 0 1
+ 4	0 1 0 0
+ 3	0 0 1 1
+ 2	0 0 1 0
+ 1	0 0 0 1
+ 0	0 0 0 0
− 0	1 1 1 1
− 1	1 1 1 0
− 2	1 1 0 1
− 3	1 1 0 0
− 4	1 0 1 1
− 5	1 0 1 0
− 6	1 0 0 1
− 7	1 0 0 0

2’s Complement Representation

★ Positive numbers are represented in “Binary”

0	Magnitude (Binary)

★ Negative numbers are represented in “2’s Comp.”

1	Code (2’s Comp.)

$(+3)_{10}$ ⇨ $(0\ 011)_2$

$(-3)_{10}$ ⇨ $(1\ 101)_2$

★ There is 1 representation for ‘0’

$(+0)_{10}$ ⇨ $(0\ 000)_2$

$(-0)_{10}$ ⇨ $(0\ 000)_2$

1’s Comp. 1 1 1 1

$$\begin{array}{rr} & 1\,1\,1\,1 \\ + & 1 \\ \hline 1 & \boxed{0\,0\,0\,0} \end{array}$$

2’s Complement Range

★ **4-Bit Representation**

$2^4 = 16$ Combinations

$$-8 \leq \text{Number} \leq +7$$

$$-2^3 \leq \text{Number} \leq +2^3 - 1$$

★ **n-Bit Representation**

$$-2^{n-1} \leq \text{Number} \leq +2^{n-1} - 1$$

Decimal	2’s Comp.
+ 7	0 1 1 1
+ 6	0 1 1 0
+ 5	0 1 0 1
+ 4	0 1 0 0
+ 3	0 0 1 1
+ 2	0 0 1 0
+ 1	0 0 0 1
+ 0	0 0 0 0
− 1	1 1 1 1
− 2	1 1 1 0
− 3	1 1 0 1
− 4	1 1 0 0
− 5	1 0 1 1
− 6	1 0 1 0
− 7	1 0 0 1
− 8	1 0 0 0

Number Representations

★ 4-Bit Example

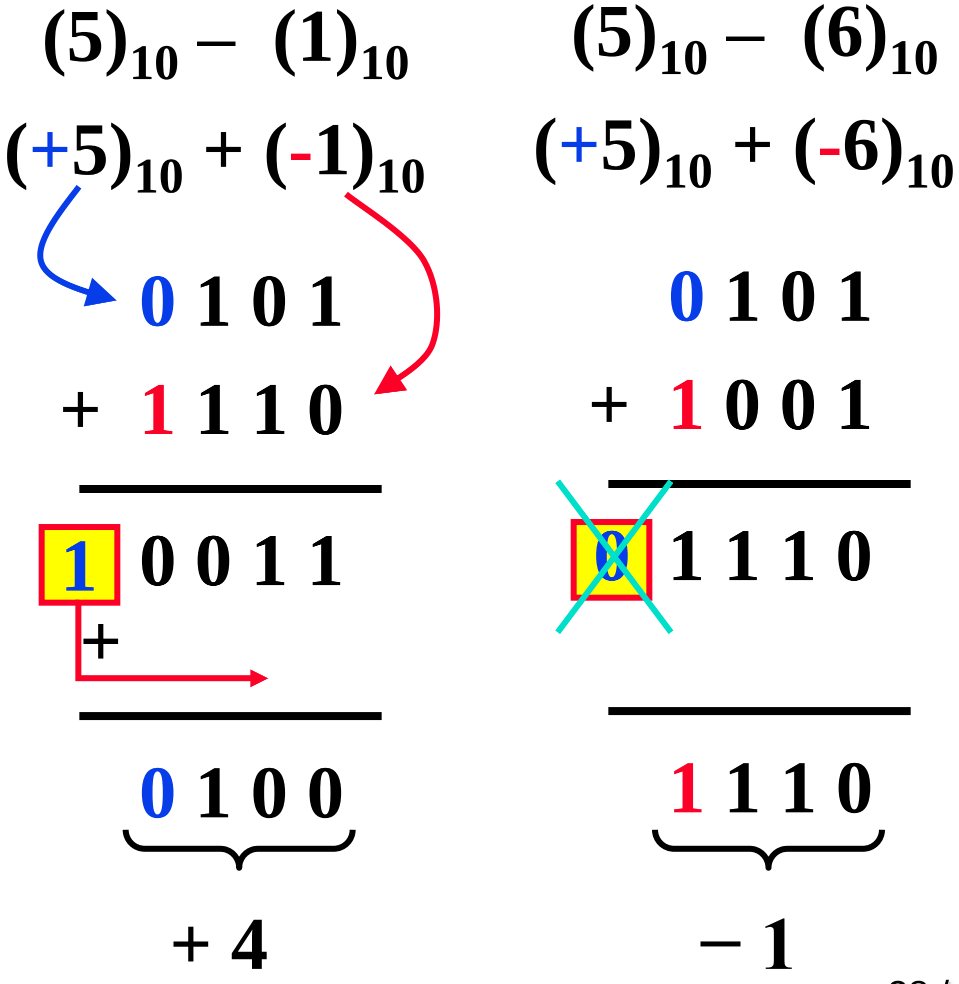
	Unsigned Binary	Signed Magnitude	1's Comp.	2's Comp.
Range	0 ≤ N ≤ 15	-7 ≤ N ≤ +7	-7 ≤ N ≤ +7	-8 ≤ N ≤ +7
Positive	□ Binary	0 □ □ □ Binary	0 □ □ □ Binary	0 □ □ □ Binary
Negative	X	1 □ □ □ Binary	1 □ □ □ 1's Comp.	1 □ □ □ 2's Comp.

Binary Subtraction Using 1’s Comp. Addition

- ★ Change “*Subtraction*” to “*Addition*”

- ★ If “*Carry*” = 1 then add it to the LSB, and the result is positive (in *Binary*)

- ★ If “*Carry*” = 0 then the result is negative (in *1’s Comp.*)



$(5)_{10} - (1)_{10}$

$(+5)_{10} + (-1)_{10}$

$$\begin{array}{rl} & 0\,1\,0\,1 \\ + & 1\,1\,1\,0 \\ \hline \boxed{1} & 0\,0\,1\,1 \\ + & \\ \hline & \underbrace{0\,1\,0\,0}_{+\,4} \end{array}$$

$(5)_{10} - (6)_{10}$

$(+5)_{10} + (-6)_{10}$

$$\begin{array}{rl} & 0\,1\,0\,1 \\ + & 1\,0\,0\,1 \\ \hline \cancel{\boxed{0}} & 1\,1\,1\,0 \\ \hline & \underbrace{1\,1\,1\,0}_{-\,1} \end{array}$$

Binary Subtraction Using 2’s Comp. Addition

- ★ Change “*Subtraction*” to “*Addition*”
- ★ If “*Carry*” = 1 ignore it, and the result is positive (in *Binary*)
- ★ If “*Carry*” = 0 then the result is negative (in *2’s Comp*.)

$(5)_{10} - (1)_{10}$

$(+5)_{10} + (-1)_{10}$

$$\begin{array}{r} 0\ 1\ 0\ 1 \\ +\ \ 1\ 1\ 1\ 1 \\ \hline \cancel{\boxed{1}}\ \ \underbrace{0\ 1\ 0\ 0}_{+4} \end{array}$$

$(5)_{10} - (6)_{10}$

$(+5)_{10} + (-6)_{10}$

$$\begin{array}{r} 0\ 1\ 0\ 1 \\ +\ \ 1\ 0\ 1\ 0 \\ \hline \cancel{\boxed{0}}\ \ \underbrace{1\ 1\ 1\ 1}_{-1} \end{array}$$

Binary Codes

★ Group of *n* bits

- Up to 2^n combinations
- Each *combination* represents *an element* of information

★ Binary Coded Decimal (BCD)

- Each Decimal Digit is represented by 4 bits
- (0 – 9) ⇨ Valid combinations
- (10 – 15) ⇨ Invalid combinations

Decimal	BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

BCD Addition

★ **One decimal digit + one decimal digit**

- **If the result is 1 decimal digit (≤ 9), then it is a simple binary addition**

***Example*:**

$$\begin{array}{r} 5 \\ +\ \ 3 \\ \hline \boxed{8} \end{array} \quad \Longleftrightarrow \quad \begin{array}{r} 0\,1\,0\,1 \\ +\ \ 0\,0\,1\,1 \\ \hline \boxed{1\,0\,0\,0} \end{array}$$

- **If the result is two decimal digits (≥ 10), then binary addition gives invalid combinations**

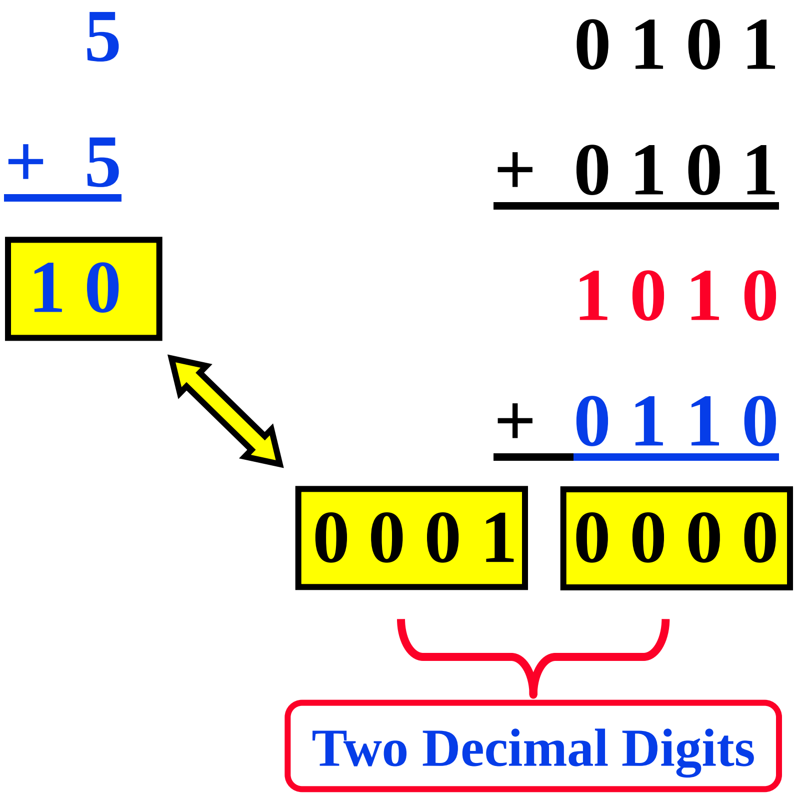
***Example*:**

$$\begin{array}{r} 5 \\ +\ \ 5 \\ \hline \end{array} \qquad \begin{array}{r} 0\,1\,0\,1 \\ +\ \ 0\,1\,0\,1 \\ \hline \end{array}$$

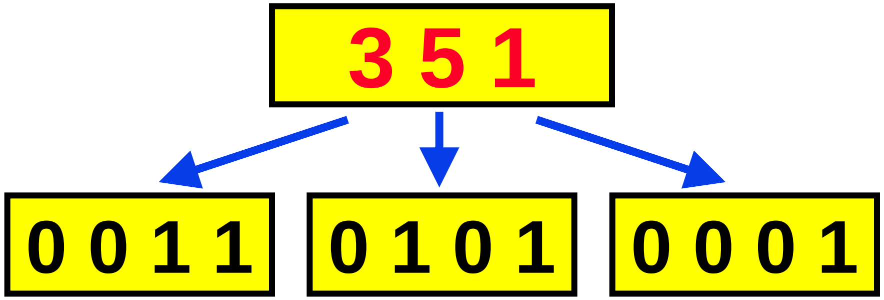
$$\boxed{0\,0\,0\,1}\ \boxed{0\,0\,0\,0} \quad \Longleftrightarrow \quad \boxed{1\,0} \qquad 1\,0\,1\,0$$

BCD Addition

- If the binary result is greater than 9, correct the result by adding 6



Multiple Decimal Digits



Gray Code

- One bit changes from one code to the next code
- Different than Binary

Decimal	Gray
00	0000
01	0001
02	0011
03	0010
04	0110
05	0111
06	0101
07	0100
08	1100
09	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Binary
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

ASCII Code

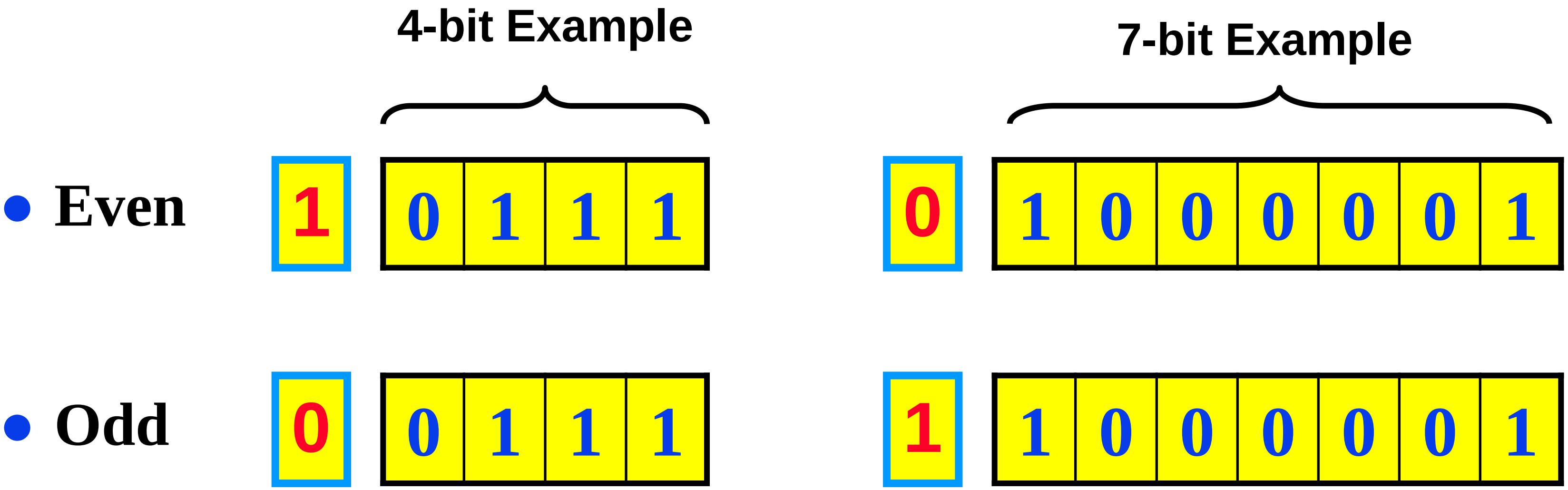
American Standard Code for Information Interchange

Info	7-bit Code
A	1000001
B	1000010
• • •	• • •
Z	1011010
a	1100001
b	1100010
• • •	• • •
z	1111010
@	1000000
?	0111111
+	0101011

Error Detecting Codes

★ **Parity**

One bit added to a group of bits to make the total number of ‘1’s (including the parity bit) *even* or *odd*



		4-bit Example		7-bit Example
• Even	1	0 1 1 1	0	1 0 0 0 0 0 1
• Odd	0	0 1 1 1	1	1 0 0 0 0 0 1

★ **Good for checking single-bit errors**

Binary Logic

★ Operators

- **NOT**

  If ‘x’ = 0 then NOT ‘x’ = 1

  If ‘x’ = 1 then NOT ‘x’ = 0

- **AND**

  If ‘x’ = 1 AND ‘y’ = 1 then ‘z’ = 1

  Otherwise ‘z’ = 0

- **OR**

  If ‘x’ = 1 OR ‘y’ = 1 then ‘z’ = 1

  Otherwise ‘z’ = 0

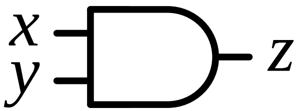
Binary Logic

★ Truth Tables, Boolean Expressions, and Logic Gates

AND

x	*y*	*z*
0	0	0
0	1	0
1	0	0
1	1	1

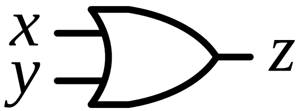
$$z = x \bullet y = x\, y$$



OR

x	*y*	*z*
0	0	0
0	1	1
1	0	1
1	1	1

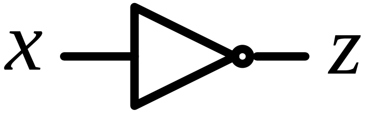
$$z = x + y$$



NOT

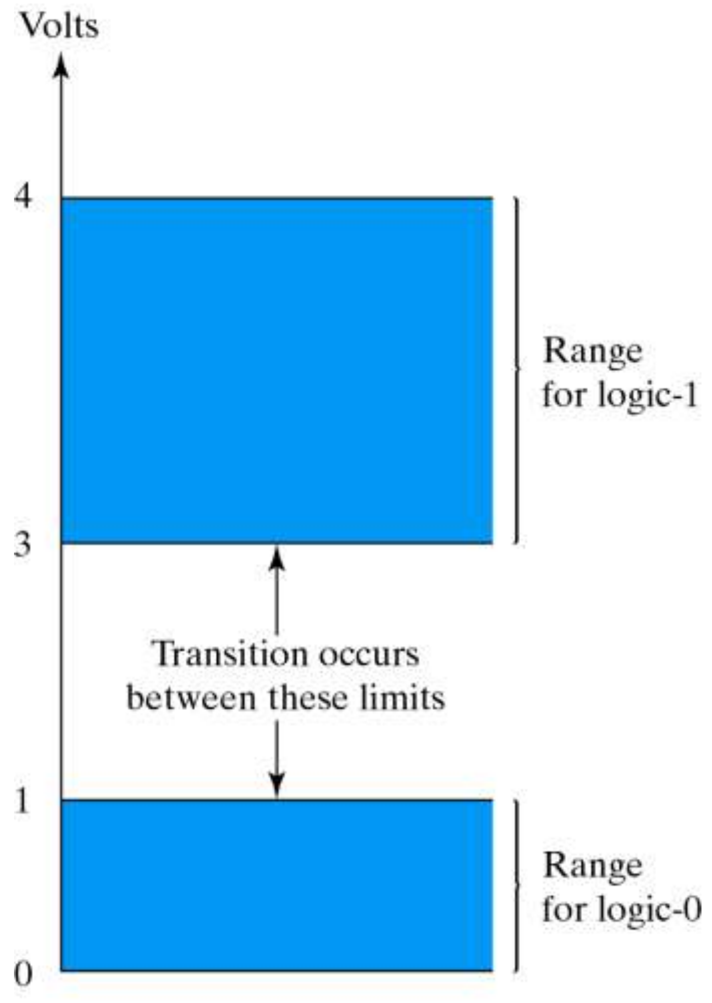
x	*z*
0	1
1	0

$$z = \overline{x} = x'$$



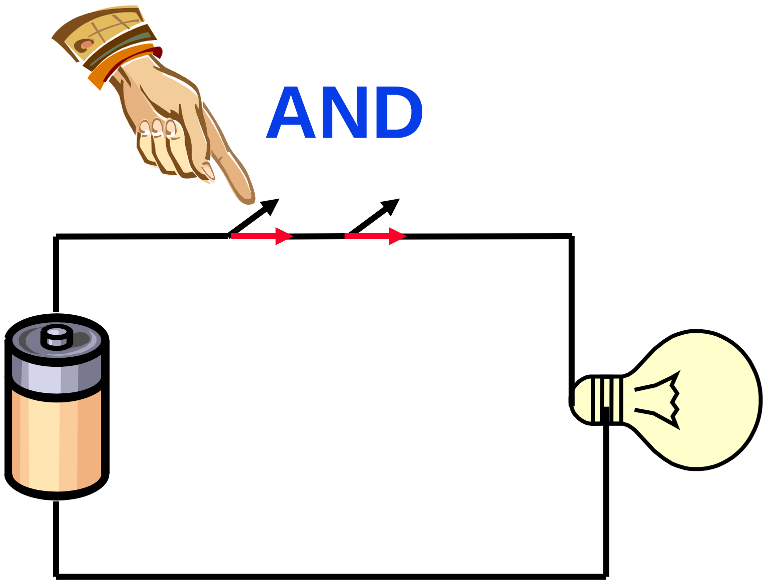
Logic Signals

- ★ **Binary '0' is represented by a "*low*" voltage (range of voltages)**
- ★ **Binary '1' is represented by a "*high*" voltage (range of voltages)**
- ★ **The "voltage ranges" guard against noise**

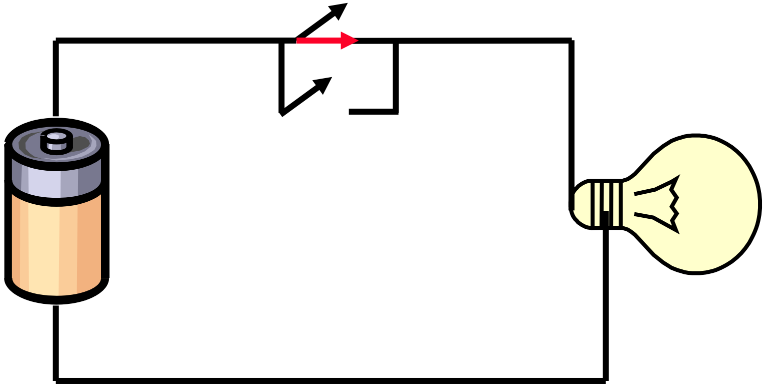


Example of binary signals

Switching Circuits



AND



OR

Homework

- Mano
  - Chapter 1
    - 1-2
    - 1-7
    - 1-9
    - 1-10
    - 1-11
    - 1-16
    - 1-18
    - 1-20
    - 1-24(a)
    - 1-29

- Write your family name in ASCII with odd parity

- Decode the following ASCII string (with MSB = parity):

  11000011 01101111 11101101
  11110000 11000000 01010000
  01010011 01010101 11010100

  Is the parity *even* or *odd*?

Homework

★ Mano

1-2 What is the exact number of bytes in a system that contains (a) 32K byte, (b) 64M byte, and (c) 6.4G byte?

1-7 Express the following numbers in decimal: $(10110.0101)_2$, $(16.5)_{16}$, and $(26.24)_8$.

1-9 Convert the hexadecimal number 68BE to binary and then from binary convert it to octal.

1-10 Convert the decimal number 345 to binary in two ways: (a) convert directly to binary, (b) convert first to hexadecimal, then from hexadecimal to binary. Which method is faster?

Homework

1-11 **Do the following conversion problems:**

(a) Convert decimal 34.4375 to binary.

(b) Calculate the binary equivalent of 1/3 out to 8 places. Then convert from binary to decimal. How close is the result to 1/3?

(c) Convert the binary result in (b) into hexadecimal. Then convert the result to decimal. Is the answer the same?

1-16 **Obtain the 1's and 2's complements of the following binary numbers:**

(a) 11101010 (b) 01111110 (c) 00000001 (d) 10000000 (e) 00000000

Homework

1-18 **Perform subtraction on the following unsigned binary numbers using the 2's-complement of the subtrahend. Where the result should be negative, 2's complement it and affix a minus sign.**

(a) 11011 – 11001 (b) 110100 – 10101 (c) 1011 – 110000
(d) 101010 – 101011

1-20 **Convert decimal +61 and +27 to binary using the signed-2's complement representation and enough digits to accommodate the numbers. Then perform the binary equivalent of (+27) + (– 61), (–27) + (+61) and (–27) + (–61). Convert the answers back to decimal and verify that they are correct.**

Homework

1-24 Represent decimal number 6027 in (a) BCD

1-29 The following is a string of ASCII characters whose bit patterns have been converted into hexadecimal for compactness: 4A EF 68 6E 20 C4 EF E5. Of the 8 bits in each pair digit, the leftmost is a parity bit. The remaining bits are the ASCII code.

(a) Convert to bit form and decode the ASCII

(b) Determine the parity used: odd or even.