NUMBER SYSTEM

Introduction to Numbering Systems

- We are all familiar with the decimal number system (Base 10). Some other number systems that we will work with are

  - Binary → Base 2
  - Octal → Base 8
  - Hexadecimal → Base 16

Characteristics of Numbering Systems

1) The digits are consecutive.
2) The number of digits is equal to the size of the base.
3) Zero is always the first digit.
4) The base number is never a digit.
5) When 1 is added to the largest digit, a sum of zero and a carry of one results.
6) Numeric values determined by the have implicit positional values of the digits.

Significant Digits

Binary: 11101101

Most significant bit

Least significant bit

Hexadecimal: 1D63A7A

Most significant digit

Least significant digit

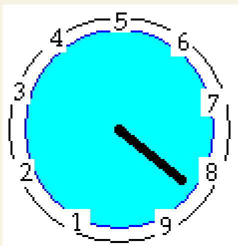
Data Representation?

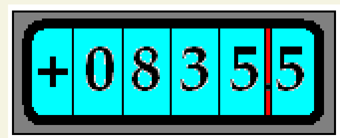
1) Representation = Measurement
2) Most things in the "Real World" actually exist as a single, continuously varying quantity ***Mass, Volume, Speed, Pressure, Temperature***
3) Easy to measure by "representing" it using a different thing that varies in the same way Eg. Pressure as the height of column of mercury or as voltage produced by a pressure transducer
4) These are ANALOG measurements

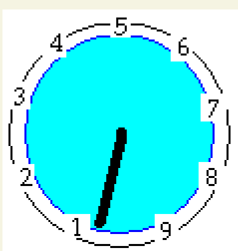
Digital Representation

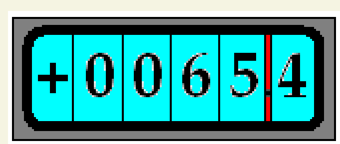
- Convert ANALOG to DIGITAL measurement by using a scale of units
- DIGITAL measurements
  - In units – a set of symbolic values – **digits**
  - Values larger than any symbol in the set use sequence of digits – **Units, Tens, Hundreds…**
  - Measured in discrete or whole units
  - Difficult to measure something that is not a multiple of units in size. **Eg Fractions**

Analog vs. Digital representation









Data Representation

- Computers use digital representation
- Based on a binary system
  (uses on/off states to represent 2 digits).
- Many different types of data.
- Examples?
- ALL data (no matter how complex)
  must be represented in memory as binary digits (bits).

Number systems and computers

- Computers store all data as binary digits, but we may need to convert this to a number system we are familiar with.

- Computer programs and data are often represented (outside the computer) using octal and hexadecimal number systems because they are a short hand way of representing binary numbers.

Number Systems - Decimal

- The decimal system is a base-10 system.
- There are 10 distinct digits (0 to 9) to represent any quantity.
- For an n-digit number, the value that each digit represents depends on its weight or position.
- The weights are based on powers of 10.

$1024 = 1*10^3 + 0*10^2 + 2*10^1 + 4*10^0 = 1000 + 20 + 4$

Binary Number System

- Also called the "Base 2 system"
- The binary number system is used to model the series of electrical signals computers use to represent information
  - 0 represents the no voltage or an off state
  - 1 represents the presence of voltage or an on state

Binary Addition

4 Possible Binary Addition Combinations:

(1)
$$\begin{array}{r} 0 \\ +0 \\ \hline 00 \end{array}$$

Carry Sum

(2)
$$\begin{array}{r} 0 \\ +1 \\ \hline 01 \end{array}$$

(3)
$$\begin{array}{r} 1 \\ +0 \\ \hline 01 \end{array}$$

(4)
$$\begin{array}{r} 1 \\ +1 \\ \hline 10 \end{array}$$

Note that leading zeroes are frequently dropped.

Decimal to Binary Conversion

- The easiest way to convert a decimal number to its binary equivalent is to use the **Division Algorithm**
- This method repeatedly divides a decimal number by 2 and records the quotient and remainder
  - The remainder digits (a sequence of zeros and ones) form the binary equivalent in least significant to most significant digit sequence

Division Algorithm

Convert 67 to its binary equivalent:

$\mathbf{67_{10} = x_2}$

Step 1: 67 / 2 = 33 R 1 *Divide 67 by 2. Record quotient in next row*

Step 2: 33 / 2 = 16 R 1 ***Again divide by 2; record quotient in next row***

Step 3: 16 / 2 = 8 R 0 ***Repeat again***

Step 4: 8 / 2 = 4 R 0 ***Repeat again***

Step 5: 4 / 2 = 2 R 0 ***Repeat again***

Step 6: 2 / 2 = 1 R 0 ***Repeat again***

Step 7: 1 / 2 = 0 R 1 **STOP *when quotient equals 0***

$\boxed{1\ 0\ 0\ 0\ 0\ 1\ 1_2}$

Binary to Decimal Conversion

- The easiest method for converting a binary number to its decimal equivalent is to use the **Multiplication Algorithm**
- Multiply the binary digits by increasing powers of two, starting from the right
- Then, to find the decimal number equivalent, sum those products

Multiplication Algorithm

Convert $(10101101)_2$ to its decimal equivalent:

Binary	1	0	1	0	1	1	0	1
	x	x	x	x	x	x	x	x
Positional Values	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Products	128 +		32 +		8 +	4 +		1

$$\boxed{173_{10}}$$

Octal Number System

- Also known as the **Base 8 System**
- Uses digits **0 – 7**
- Readily converts to binary
- Groups of three (binary) digits can be used to represent each octal digit
- Also uses multiplication and division algorithms for conversion to and from base 10

Decimal to Octal Conversion

Convert 427_{10} to its octal equivalent:

427 / 8 = 53 R3	Divide by 8; R is LSD
53 / 8 = 6 R5	Divide Q by 8; R is next digit
6 / 8 = 0 R6	Repeat until Q = 0

$\boxed{653_8}$

Octal to Decimal Conversion

Convert 653_8 to its decimal equivalent:

Octal Digits →	6	5	3
	x	x	x
Positional Values →	8^2	8^1	8^0
Products →	384 +	40 +	3

$$\boxed{427_{10}}$$

Octal to Binary Conversion

Each octal number converts to 3 binary digits

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

To convert 653_8 to binary, just substitute code:

$$\begin{array}{ccc} 6 & 5 & 3 \\ \downarrow & \downarrow & \downarrow \\ 110 & 101 & 011 \end{array}$$

Hexadecimal Number System

- Base 16 system
- Uses digits 0-9 & letters A,B,C,D,E,F
- Groups of four bits represent each base 16 digit

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Decimal to Hexadecimal Conversion

Convert 830_{10} to its hexadecimal equivalent:

830 / 16 = 51 R14 = E in Hex

51 / 16 = 3 R3

3 / 16 = 0 R3

$\boxed{33E_{16}}$

Hexadecimal to Decimal Conversion

Convert $3B4F_{16}$ to its decimal equivalent:

	3	B	4	F
Hex Digits →	3	B	4	F
	x	x	x	x
Positional Values →	16^3	16^2	16^1	16^0

Products → 12288 +2816 + 64 +15

$\boxed{15{,}183_{10}}$

Binary to Hexadecimal Conversion

- The easiest method for converting binary to hexadecimal is to use a **substitution code**
- Each hex number converts to 4 binary digits

Substitution Code

0000 = 0	0100 = 4	1000 = 8	1100 = C
0001 = 1	0101 = 5	1001 = 9	1101 = D
0010 = 2	0110 = 6	1010 = A	1110 = E
0011 = 3	0111 = 7	1011 = B	1111 = F

Substitution Code

Convert $010101101010111001101010_2$ to hex using the 4-bit substitution code :

0101 0110 1010 1110 0110 1010

0101	0110	1010	1110	0110	1010
5	6	A	E	6	A

$\boxed{56AE6A_{16}}$

Substitution Code

Substitution code can also be used to convert binary to octal by using 3-bit groupings:

010	101	101	010	111	001	101	010
2	5	5	2	7	1	5	2

$$\boxed{25527152_8}$$

Complementary Arithmetic

- 1's complement
  - Switch all 0's to 1's and 1's to 0's

Binary # → 10110011

1's complement → 01001100

Complementary Arithmetic

- 2's complement
  - Step 1: Find 1's complement of the number

    Binary # → 11000110

    1's complement → 00111001
  - Step 2: Add 1 to the 1's complement

$$\begin{array}{r} 00111001 \\ +\quad 00000001 \\ \hline 00111010 \end{array}$$

Negative Integers – One's (1's) Complement

- Computers generally use a system called "complementary representation" to store negative integers.
- Two basic types - ones and twos complement, of which 2's complement is the most widely used.
- The number range is split into two halves, to represent the positive and negative numbers.
- Negative numbers begin with 1, positive with 0.

Negative Integers – One's (1's) Complement

- To perform 1's complement operation on a binary number, replace 1's with 0's and 0's with 1's (ie Complement it!)

+6 represented by: 00000110

-6 represented by: 11111001

- Advantages: arithmetic is easier (cheaper/faster electronics)
- Fairly straightforward addition
  - Add any carry from the Most Significant (left-most) Bit to Least Significant (right-most) Bit of the result
- For subtraction
  - form 1's complement of number to be subtracted and then add
- Disadvantages : still two representations for zero

00000000 and 11111111 (in 8-bit representation)

Negative Integers – Two's (2's) Complement

- To perform the 2's complement operation on a binary number
  - replace 1's with 0's and 0's with 1's (i.e. the one's complement of the number)
  - add 1

+6 represented by: 00000110

-6 represented by: 11111010

- Advantages:
  - Arithmetic is very straightforward
  - End Around Carry is ignored
- only one representation for zero (00000000)

Negative Integers – Two's (2's) Complement

Two's Complement

- To convert an integer to 2's complement
  - » Take the binary form of the number

00000110 (6 as an 8-bit representation)

  - » Flip the bits: (Find 1's Complement)

11111001

  - » Add 1

11111001

+1

11111010 (2's complement of 6)

- Justification of representation: 6+(-6)=0?

00000110 (6)

+11111010 (2's complement of 6)

100000000 (0)

Negative Integers – Two's (2's) Complement

- Interpretation of Negative Results

 00000101 (5)
+11110100 (-12)
 11111001 (   )

   - Result is negative
   - Negative what?

Take the 2's comp of the result to find out since the 2's comp of a 2's comp is the original number

   - Negative 7

the 2's complement of 11111001 is 00000111 or 7_{10}

- Problem: word size is fixed, but addition can produce a result that is too large to fit in the number of bits available.
  This is called overflow.
- If two numbers of the same sign are added, but the result has the opposite sign then overflow has occurred
- Overflow can occur whether or not there is a carry
- Examples:
  01000000 (+64) 10000000 (-128) 01000001 (+65)
  11000000 (-64) 10000001 (-127) 01000000 (+64)

Binary Coded Decimal

- Scheme whereby each decimal digit is represented by its 4-bit binary code
- 7 = 0111
- 246 = 001001000110
- Many CPUs provide arithmetic instructions for operating directly on BCD. However, calculations slower and more difficult.

Data Representation

- Computers store everything as binary digits. So, how can we encode numbers, images, sound, text ??
- We need standard encoding systems for each type of data.
- Some standards evolve from proprietary products which became very popular.
- Other standards are created by official industry bodies where none previously existed.
  - Some example encoding standards are ?

Alphanumeric Data

- Alphanumeric data such as names and addresses are represented by assigning a unique binary code or sequence of bits to represent each character.
- As each character is entered from a keyboard (or other input device) it is converted into a binary code.
- Character code sets contain two types of characters:
  - Printable (normal characters)
  - Non-printable. Characters used as control codes.
    - CTRL G (beep)
    - CTRL Z (end of file)

Alphanumeric Codes

- There are 3 main coding methods in use:
- ASCII
- EBCDIC

ASCII

- 7-bit code (128 characters)
- has an extended 8-bit version
- used on PC's and non-IBM mainframes
- widely used to transfer data from one computer to another

EBCDIC

- An 8-bit code (256 characters)
- Different collating sequence to ASCII
- used on mainframe IBM machine
- Both ASCII and EBCDIC are 8 bit codes inadequate for representing all international characters
- Some European characters
- Most non-Alphabetic languages
  eg : Mandarin, Kanji, Arabic, etc…