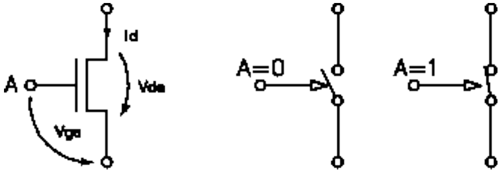
Binary Systems

Outline

- Numbering Systems
- Binary Systems
- Binary Arithmetic
- Binary Coding

Why Binary

- Computer is made of switches.
- Switches only have two states: open or closed.
- How information can be represented by “open” and “closed”?



Numbers

- Let's first look at decimal numbers
  - 134, 45.89, 162.375, …
- What do they mean?
- How are they made of?

- Each number consists of a bunch of digits (0, 1, 2, 3, 4, 5, 6,7, 8, 9):

1	6	2	.	3	7	5	Digits

- Each digit has a weight. The value of the weight depends on the digit's position and they are powers of the base (number of the total digits, which is 10 in this case):

1	6	2	.	3	7	5	Digits
10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	Weights

- To find the decimal value of a number, multiply each digit by its weight and sum the products.

$$(1 \times 10^2) + (6 \times 10^1) + (2 \times 10^0) + (3 \times 10^{-1}) + (7 \times 10^{-2}) + (5 \times 10^{-3}) = 162.375$$

Decimal Number

Numbering Systems

- This numbering system is known as **positional** numbering system. Each number is represented using an alphabet of symbols $\{A_0,A_1,\ldots,A_{B-1}\}$ and the **position** of each symbol indicates the power B is raised to in the expansion.

$$(A_2A_3A_0A_1.A_4)_B=A_2 \times B^3 +A_3 \times B^2 +A_0 \times B^1 +A_1 \times B^0 + A_4 \times B^{-1}$$

- Each decimal number is represented using symbols taken from an alphabet of ten symbols/ digits {0,1,2,3,4,5,6,7,8,9} and a powers of 10 expansion.

Binary Systems

- In the binary numbering systems, numbers are represented using an alphabet of two symbols {0,1} and a powers of two expansion.

1	1	0	1	.	0	1	Binary digits, or bits
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	Weights (in base 2)

$$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2})$$

$$8 + 4 + 0 + 1 + 0 + 0.25 = 13.25$$

Number Conversions

- Converting a decimal number to a number in base-*B* (e.g., binary):
  - Separate the number into an integer part and a fraction part;
  - Convert the integer part by dividing the number with the base *B* and accumulating the remainders;
  - Convert the fraction part by multiplying with B and accumulating the integers.

- **Example**: convert decimal 41.6875 to binary.

Integer part:

Integer	Remainder
41 / 2	1
20 / 2	0
10 / 2	0
5 / 2	1
2 / 2	0
1 / 2	1
0	

Inverse order

$(41)_{10}=(101001)_2$

Fraction part:

	Integer		Fraction
0.6875 x 2	1	+	0.3750
0.3750 x 2	0	+	0.7500
0.7500 x 2	1	+	0.5000
0.5000 x 2	1	+	0.0000

forward order

$(0.6875)_{10}=(0.1011)_2$

Final answer: $(41.6875)_{10}=(101001.1011)_2$

Why Does This Work

41/10=4 rem 1	41/2=20 rem 1
4/10=0 rem 4	20/2=10 rem 0
	10/2=5 rem 0
	5/2=2 rem 1
	2/2=1 rem 0
	1/2=0 rem 1

$4 \times 10^1 + 1 \times 10^0 = 41$

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 41$

MSB (most significant bit)

LSB (least significant bit)

Binary Systems

- A binary system relates closely to the operation of switching circuits which form the basis of computers.
- Switches have two states ‘open’ or ‘closed’ which may be represented by the symbols ‘0’ and ‘1’.
- Since we can represent any symbol using a combination ones and zeros, binary acts as a good description of the inner operation of digital electronic circuitry.

Binary Systems

- However, a binary numbering system does not lend itself to easy comprehension by humans. A string of ones and zeros representing information (decimal numbers, words, etc.) still appears to a human user as a string of ones and zeros.
- The hexadecimal and octal numbering systems allow easier interpretation of binary data.

Octal: partitioning the binary number into groups of three digits each

$$(10\ 110\ 001\ 101\ 011\ .\ 111\ 100\ 000\ 110)_2 = (26153.7406)_8$$

2 6 1 5 3 7 4 0 6

Hexadecimal: partitioning the binary number into groups of four digits each

$$(10\ 1100\ 0110\ 1011\ .\ 1111\ 0000\ 0110)_2 = (2C6B.F06)_{16}$$

2 C 6 B F 0 6

Binary Systems

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	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

Binary Systems

- Each group of four bits can be represented using just one hexadecimal symbol. As a result a byte (8 bits) of information can be described by two hexadecimal symbols. System engineers can easily read the contents of a memory register.

- e.g. The first four bytes contain the data

  FF-D8-FF-E1

  the binary equivalent is

  11111111-11011000-11111111-11100001

Binary Systems

0000	FF	D8	FF	E1	1D	FE	45	78	69	66	00	00	49	49	2A	00
0010	08	00	00	00	09	00	0F	01	02	00	06	00	00	00	7A	00
0020	00	00	10	01	02	00	14	00	00	00	80	00	00	00	12	01
0030	03	00	01	00	00	00	01	00	00	00	1A	01	05	00	01	00
0040	00	00	A0	00	00	00	1B	01	05	00	01	00	00	00	A8	00
0050	00	00	28	01	03	00	01	00	00	00	02	00	00	00	32	01
0060	02	00	14	00	00	00	B0	00	00	00	13	02	03	00	01	00
0070	00	00	01	00	00	00	69	87	04	00	01	00	00	00	C4	00
0080	00	00	3A	06	00	00	43	61	6E	6F	6E	00	43	61	6E	6F
0090	6E	20	50	6F	77	65	72	53	68	6F	74	20	41	36	30	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	B4	00	00	00
00B0	01	00	00	00	B4	00	00	00	01	00	00	00	32	30	30	34
00C0	3A	30	36	3A	32	35	20	31	32	3A	33	30	3A	32	35	00
00D0	1F	00	9A	82	05	00	01	00	00	00	86	03	00	00	9D	82
00E0	05	00	01	00	00	00	8E	03	00	00	00	90	07	00	04	00

We could convert decimal to binary first, then group the bits into 3 (or 4) and convert to octal (or hexadecimal). Or we could convert them directly using the dividing/ multiplying method that we learned earlier.

- **Example**: convert decimal 41.6875 to octal.

Integer part:

Integer	Remainder
41 / 8	1
5 / 8	5
0	

Inverse order

$(41)_{10}=(51)_8$

Fraction part:

	Integer		Fraction
0.6875 x 8	5	+	0.5000
0.5000 x 8	4	+	0.0000

$(0.6875)_{10}=(0.54)_8$

Final answer: $(41.6875)_{10}=(51.54)_8$

Binary Arithmetic

- As with the decimal system it is possible to perform arithmetic using binary numbers.
- The binary addition and subtraction follows the same rules as decimal addition and subtraction, except now only the symbols {0,1} are used.

$$\begin{array}{rr} & 101011 \\ + & 011011 \\ \hline & 1000110 \end{array}$$

$$\begin{array}{rr} & 1000110 \\ - & 101011 \\ \hline & 011011 \end{array}$$

Binary Arithmetic

- By applying standard arithmetic rules binary multiplication is straightforward.

$$
\begin{array}{rr}
 & 111 \\
\times & 101 \\
\hline
 & 111 \\
 & 0000 \\
+ & 11100 \\
\hline
 & 100011
\end{array}
$$

Binary Arithmetic

- Binary division is also straight forward.

$$
\begin{array}{r@{\,}l}
 & 111 \\ \cline{2-2}
101 & \big)\,100011 \\
 & \underline{101} \\
 & 111 \\
 & \underline{101} \\
 & 101 \\
 & \underline{101} \\
 & 0
\end{array}
$$

Negative Numbers

- Positive binary numbers are easily stored in digital computers using just ones and zeros.
- Negative binary numbers however are not readily represented using just ones and zeros.
- The notation we are used to distinguishes between positive and negative numbers by using the symbols ‘+’ and ‘–’. In a digital computer we only use the symbols ‘0’ and ‘1’.

Negative Numbers

- A simple method of distinguishing between positive and negative numbers is to use the leftmost bit as a sign bit.
- The convention is that when the sign bit is zero the number is positive and when the sign bit is one the number is negative.
  - e.g. the decimal number 5 is represented in binary as 101. The number +5 is represented in this system as 0101 and the number –5 is represented as 1101.
- This notation is known as the **signed magnitude** representation.

Negative Numbers

- Another negative number notation is known as **1's complement**.
- For –*N*, an *n* bit word: $\overline{N} = (2^n - 1) - N$
- In 1's complement notation the representation of a positive binary number remains unchanged. A negative number is represented by the complement of the binary number.
  - e.g. the decimal number 5 is represented in binary as 101. The number +5 is represented in this system as 0101 and the number –5 is represented as 1010.
  - For a binary number, simply change 1's to 0's and 0's to 1's to obtain its 1's complement .

Negative Numbers

- A related (and more popular notation) is **2's complement**.
- For –*N*, an *n* bit word: $N^* = 2^n - N$
- In 2's complement notation the representation of a positive binary number remains unchanged. Now however a negative number is represented by the complement of the binary number plus 1.
  - e.g. the decimal number 5 is represented in binary as 101. The number +5 is represented in this system as 0101 and the number –5 is represented as 1010+1=1011.
  - For a binary number, the 2's complement can be formed by leaving all least significant 0's and the first 1 unchanged and replacing 1's with 0's and 0's with 1's in all other higher significant.

Negative Numbers

Decimal	Signed Magnitude	1's Complement	2's Complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

Negative Numbers

- The reasoning behind the use of 1's complement and 2's complement becomes apparent when arithmetic is considered.
- For signed magnitude numbers addition becomes more difficult since extra logic circuitry is necessary to determine whether the number is positive or negative.
- This extra level of logic is not necessary for the complementary systems.

Negative Numbers

e.g. In our familiar decimal system we have

$$-2 \quad + \quad +7 \quad = \quad +5$$

in binary this becomes

$$-010 \quad + \quad 111 \quad = \quad +101$$

and so the summation becomes a difference.
Extra circuitry is needed to determine whether or not to switch between logic designed for addition or logic designed for subtraction.

Negative Numbers

Using 1's complement the expression

$$-2 \quad + \quad +7 \quad = \quad +5$$

becomes

$$1101 \quad + \quad 0111 \quad = \quad 10100$$

the fifth bit of the sum is removed and added to the remainder,

i.e.

$$10100 \rightarrow 0100 + 0001 = 0101$$

which is the 1's complement representation for +5.

Negative Numbers

Similarly using 2's complement the expression

$$-2 \quad + \quad +7 \quad = \quad +5$$

becomes

$$1110 \quad + \quad 0111 \quad = \quad 10101$$

the fifth bit of the sum is removed, i.e.

$$10101 \longrightarrow 0101$$

which is the 2's complement representation for +5.

Negative Numbers

- Both of the complementary numbering systems allow numbers to be added without determining whether or not the numbers are positive or negative.
- For 1's complement the most significant bit is removed from the sum and added to the remainder. 2's complement has the additional advantage that the most significant bit need only be removed.
- In both the signed magnitude system and 1's complement system there are two representations of the number zero, i.e. +0 and -0. This confusion does not arise with 2's complement where only one representation for zero exists.

Binary Codes

- Digital systems represent and manipulate not only binary numbers but also many other discrete elements of information.
- Any discrete element of information that is distinct among a group of quantities can be be represented with a binary code.
- An *n*-bit binary code is a group of *n* bits that assumes up to 2^n distinct combinations of 1's and 0's, with each combination representing one element of the set that is being coded.

BCD Code

- **BCD: Binary-coded decimal**
- **Represent the decimal digits by means of a code that contains 1's and 0's.**
- **4-bit binary code for 10 decimal digits.**

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

$$(185)_{10} = (0001\ 1000\ 0101)_{BCD}$$

Gray Code

- In Gray coding adjacent values only differ in one bit location. An example is useful for illustration purposes.
  - A digital system represents 3 as the binary number 0011. If the third bit is corrupted 0011→0111 the number becomes 7.
  - If instead 3 is represented by 0010 upon corruption of the third bit the code becomes 0010 →0110 which is the Gray code for 4.
- By using Gray code a one bit error results in a less dramatic error.

Gray code	decimal equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

One-hot Code

- In one-hot coding each bit is used in isolation to represent a value. Using one-hot coding 3 is represented as 00000100 whereas 7 is represented by 01000000.

one-hot code	decimal equivalent
00000001	1
00000010	2
00000100	3
00001000	4
00010000	5
00100000	6
01000000	7
10000000	8