EEE 3131 - Digital Electronics

Lecture 1 : Number Systems & Codes

Instructor: Jerry MUWAMBA
Email: jerry.muwamba@unza.zm
jerrymuwamba@yahoo.com

October 2015

References

Our main reference text books in this course are

[1] William Kleitz, Digital Electronics: A Practical Approach with VHDL, 9th Ed., 2012, Prentice Hall ISBN-100131714902

[2] Thomas L. Floyd, Digital Fundamentals with PLD Programming, 9th Ed., 2006, Prentice Hall ISBN-10:0-13-197255-3.

[4] Maini Anil K., Digital Electronics: Principles, Devices and Applications, 2007, John Wiley and Sons Ltd, ISBN 978-0-470-03214-5.

[5] Smith R. J., Dorf R. C., Circuits Devices and Systems, 5th Ed., 2004, John Wiley and Sons Ltd, ISBN 9971-51-172-X.

However, feel free to use some additional text which you might find relevant to our course.

Course Requirements

- ✓ It is an **OBLIGATION** for all students taking this course to attend all lectures and lab sessions.

Prerequisite

- ✓ EEE 3571

Time Allocation

✓ Lectures	**4 hours/week**
✓ Labs	**3 hours/week**

Assessment

✓ Assignments /Quizzes	**5%**
✓ Labs/Mini-Projects	**15%**
✓ Tests	**20%**
✓ Final Exam	**60%**

Introduction

- ✓ To process digital information we use special electronic components that respond to binary signals.
- ✓ To design efficient digital circuits, we also need a special numbering system and a special algebra.
- ✓ In this course, first we examine the binary number system and learn to apply logic theorems to binary relations.
- ✓ Computers consist of large numbers of logic gates and memory elements organized to process data at high speed.
- ✓ Binary data or instructions are stored temporarily in registers. Binary counters are used in calculations and to keep track of computer operations.
- ✓ Instructions and data are stored at specified locations in memory and can be retrieved at will.

Introduction

Objectives of Lectures

- ✓ On completion of the Lectures on Number systems and codes, you should be able to work in the binary number system as well as the decimal system. In addition, you should be able to change a number from one base to another.
- ✓ On completion of the Lectures on Logic fundamentals, you should be able to use basic theorems to simplify and analyse logic functions in common use. Furthermore, you should be able to simplify logic expressions using Karnaugh maps.
- ✓ This therefore, will set a foundation for the material you will study later on combinational logic ckts, sequential logic ckts and finally microprocessors.

Binary Numbers

- ✓ In the decimal number system a quantity is represented by the value and the position of a digit. The number 503.14 means

$$500 \quad +0 \quad +3 \quad +\frac{1}{10} \quad +\frac{4}{100}$$

- ✓ Using powers of 10, this can be rewritten as

$$5\times10^{2}+0\times10^{1}+3\times10^{0}+1\times10^{-1}+4\times10^{-2}$$

- ✓ In other words, 10 is the base and each position to the left or right of the decimal point corresponds to a power of 10.
- ✓ A base 12 or duodecimal system was used by the Babylonians, and we still use 12 in subdividing the foot, the year, and the clock face.
- ✓ In representing data by an **ON-OFF** switch position, there are two possibilities and the corresponding numbers are **1** and **0**.
- ✓ In such a binary system, the base is 2 and the decimal number 10 is written as **1010** since

Binary Numbers

$$1\times 2^3+0\times 2^2+1\times 2^1+0\times 2^0=8+0+2+0=10$$

- ✓ In electronic logic circuits the numbers **1** and **0** usually correspond to two easily distinguished voltage levels specified by the circuit designer. For instance, in TTL (Transistor Transistor Logic), **0** corresponds to a voltage near zero and **1** to a voltage near +5 V.

Number Conversion

- ✓ **Binary-to-decimal conversion**. In a binary number, each position to the right or left of the "binary point" corresponds to a power of 2, and each power of 2 has a decimal equivalent.
- ✓ To convert a binary number to its decimal equivalent, add the decimal equivalents of each position occupied by a **1**.

Binary Numbers

- ✓ For instance, $\mathbf{110001} = 2^5 + 2^4 + 0 + 0 + 0 + 2^0 = 32 + 16 + 1 = 49$

$$\mathbf{101.01} = 2^2 + 0 + 2^0 + 0 + 2^{-2} = 4 + 1 + \frac{1}{4} = 5.25$$

- ✓ **Decimal-to-binary conversion**. A decimal number can be converted to its binary equivalent by the inverse process, that is, by expressing the decimal number as a sum of powers of 2.
- ✓ An automatic and more popular method is the double-dabble in which integers and decimals are handled separately.
- ✓ To convert a decimal integer to its equivalent, progressively divide the decimal number by 2, noting the remainders; the remainders taken in reverse order form the binary equivalent.
- ✓ To convert a decimal fraction to its binary equivalent, progressively multiply the fraction by 2, removing and noting the carries; the carries taken in forward order form the binary equivalent.

Binary Numbers

✓ **Example 1**

✓ Convert decimal 28.375 to its binary equivalent.

✓ **Solution**. Using the double-dabble method on the integer,

2	28	remainder
	14	0
	7	0
	3	1
	1	1
	0	1

✓ The binary equivalent is **11100**.

✓ Then converting the fraction

$$0.375 \times 2 = 0.75 \text{ with a carry of } 0$$

$$0.75 \times 2 = 1.50 \text{ with a carry of } 1$$

$$0.5 \times 2 = 1.00 \text{ with a carry of } 1$$

✓ The binary equivalent is **.011**.

✓ Therefore, 28.375 is equivalent to binary **11100.011**.

Binary Numbers

Binary Arithmetic

- ✓ Since the binary system uses the same concept of value and position of the digits as the decimal system, we expected the associated arithmetic to be similar but easier.
- ✓ The binary multiplication table is very short. For addition in binary, we add column by column, carrying where necessary into higher position columns.
- ✓ In subtraction, we subtract column by column, borrowing where necessary from higher position columns.
- ✓ In subtracting a large number from a smaller, we can subtract the smaller from the larger and change the sign just as we do with decimals.

Binary Numbers

Example 2

✓ Convert the numbers in color to the other form and perform the indicated operations.

14	**1110**	13	**1101**	1010	**10**
+11	**+1011**	−10	**−1010**	−1101	**−13**
25	**11001**	3	**0011**	−0011	**−3**

✓ In **multiplication**, we obtain partial products using the binary multiplication table, $(0\times0=0,\ 0\times1=0,\ 1\times0=0,\ 1\times1=1)$ and then add the partial products.

✓ In **division**, we perform repeated subtractions just as in long division of decimals, See Example 3.

Binary Numbers

Example 3

✓ Convert the numbers in color to the other form and perform the indicated operations.

(a)

$$\begin{array}{r} 14.5 \\ \times\ 1.25 \\ \hline 725 \\ 290 \\ 145 \\ \hline 18.125 \end{array}$$

(b)

$$\begin{array}{r} \mathbf{101.1} \\ \mathbf{11.1}\overline{)\mathbf{10011.01}} \\ \mathbf{111} \\ \hline \mathbf{1010} \\ \mathbf{111} \\ \hline \mathbf{111} \\ \mathbf{111} \\ \hline \mathbf{0} \end{array}$$

(a)

$$\begin{array}{r} \mathbf{1110.1} \\ \times\ \mathbf{1.01} \\ \hline \mathbf{11101} \\ \mathbf{00000} \\ \mathbf{11101} \\ \hline \mathbf{10010.001} \end{array}$$

(b)

$$\begin{array}{r} 5.5 \\ 3.5\overline{)19.25} \\ 175 \\ \hline 175 \\ 175 \\ \hline 0 \end{array}$$

Binary Numbers

Bits, Bytes, and Words

- ✓ A single binary digit is called a "bit." All information in digital system is represented by a sequence of bits. An 8-bit sequence is called a "byte"; a 4-bit sequence is a "nibble."
- ✓ The number of bits in the data sequences processed by a given computer is a key characteristic called the "word length". Computers handle data in words of 4 to 64 bits.
- ✓ An 8-bit microprocessor can receive, process, store, and transmit data or instructions in form of bytes. Eight bits can be arranged in $2^8 = 256$ different combinations.

Other Notations

- ✓ The number of years in a century can be written 100D or 100_{10} . In **binary notation** this would be written $0+2^6+2^5+0+0+2^2+0+0=$ **01100100B** or **01100100_2** ; the prefix B or subscript 2 is used whenever necessary to avoid confusion.

Binary Numbers

Other Notations

- ✓ Although 8-bit numbers are easy for computers, they are difficult for humans to deal with. In **octal notation**, a single decimal number from 0 to 7 is used to represent each group of three bits.
- ✓ As an octal number, **01100100B** would be written as **01 100 100 → 144Q = 144_8** . Three-digit octal numbers are easier to remember and easier to check than their 8-bit binary equivalents.
- ✓ In the alternative notation most commonly used in microprocessor work, each group of four bits is represented by **single hexadecimal number.**
- ✓ In "hex," **01100100B** would be written as **0110 0100 → 64H = 64_{16}** . Because four bits can take on sixteen different values, we supplement the ten decimal digits 0 to 9 with letters A, B, C, D, E and F.
- ✓ For example, **11000011_2 → 1100 0011 → $C3_{16}$** and **$255_{10} = 11111111_2$ → 1111 1111 → FF_{16}** . Table 1 shows the various notations discussed thus far.

Binary Numbers

Table 1 Number Systems

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

Signed Magnitudes

- ✓ In binary notation, an *n*-bit data word can represent the first 2^n nonnegative integers.

Binary Numbers

Signed Magnitudes

- ✓ To allow for both positive and negative numbers, the most significant bit (MSB) can be designated as the sign bit (**1** for negative numbers).
- ✓ The lower order bits then represent the magnitude of the number in "straight" binary notation.
- ✓ Although it is used, this arrangement has two disadvantages: the number zero has two different representations, and two different arithmetic circuits are required to process positive and negative numbers.

Two's Complement Notation

- ✓ A better notation for computers, one that is easily implemented in hardware, is based on the fact that adding the complement of a number is equivalent to subtracting the number.
- ✓ A "complement" is that which completes; the "*n*'s complement" of a number *x* is equal to $n - x$.

Binary Numbers

Two's Complement Notation

- ✓ For example, the 10's complement of 3 is 7. To evaluate $9-3$, i.e., to subtract 3 from 9, we can "add the 10's complement" of 3 (i.e., $10-3=7$) to obtain $9+7=16$ to yield 6 after discarding the final carry.
- ✓ In the decimal system, the 10's complement of a multidigit number is easily found by taking the 9's complement of each digit (by inspection) and then adding 1.
- ✓ In general, to subtract a two-digit number B from A we use the relationship

$$A-B=A+\left[100-B\right]-100=A+\left[\left(99-B\right)+1\right]-100 \qquad [1]$$

where $\left(99-B\right)$ is the 9's complement.

- ✓ In the binary system, arithmetic is simplified if negative numbers are in signed 2's complement notation. In this notation, the MSB is the sign bit: **0** for plus, **1** for minus. To form the 2's complement of any number, positive or negative:

Form the 1's *complement by changing* **1**s *to* **0**s *and* **0**s *to* **1**s. *Add* **1**.

Binary Numbers

- ✓ If the result of an arithmetic operation has a 1 sign bit, it is a negative number in 2's complement notation; to obtain the true magnitude, subtract 1 and form the 1's complement.

Example 4

a) Obtain the 10's complement of 15 and 24.

b) Represent -15 and -24 in 8-bit signed 2's complement notation.

c) Perform $24-15$ and $15-24$ directly and by complement notation.

Solution

a) Form the 9's complement of each digit, then add 1:

$$15 \rightarrow 84+1=85 \qquad 24 \rightarrow 75+1=76$$

b) Form the 1's complement of each digit, then add 1.

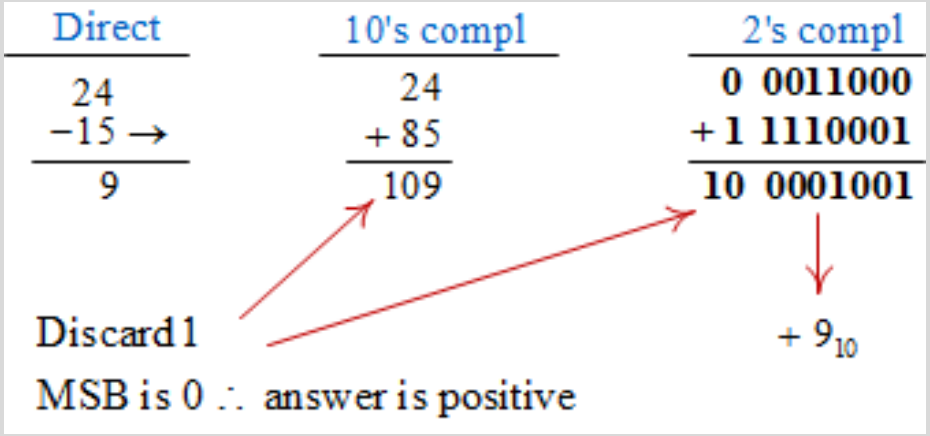
$$-15_{10} \rightarrow -1111 \rightarrow -00001111 \rightarrow 11110000+1 \rightarrow 11110001 \rightarrow 1\ 1110001$$

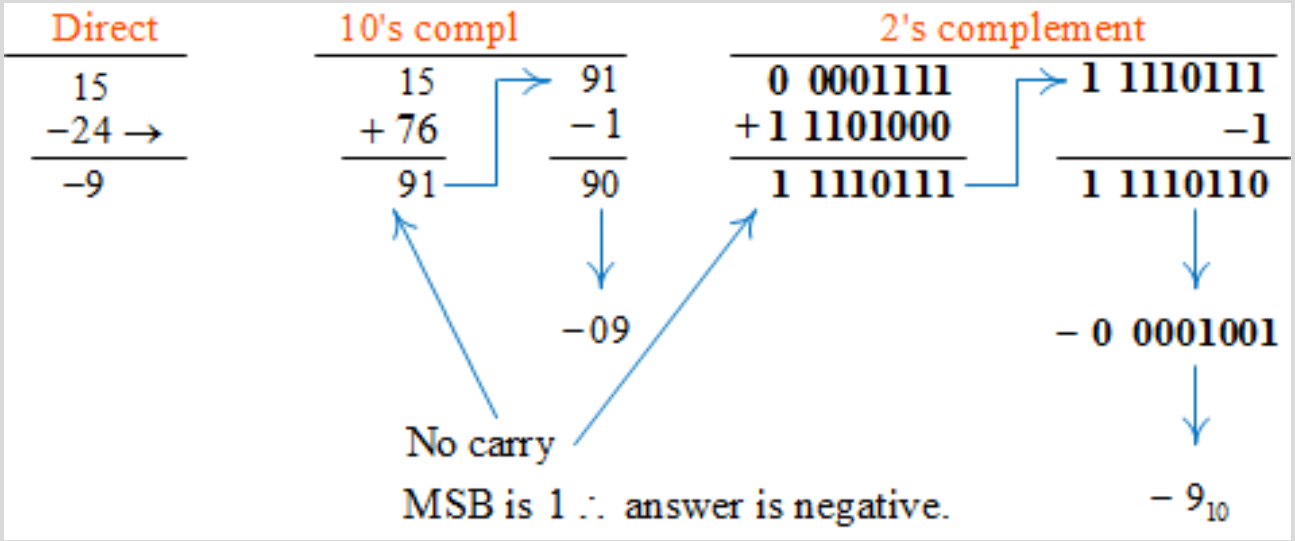
$$-24_{10} \rightarrow -11000 \rightarrow -00011000 \rightarrow 11100111+1 \rightarrow 11101000 \rightarrow 1\ 1101000$$

Binary Numbers

Example 4

c) Solution





Binary Numbers

Binary-Coded Decimals (BCD)

- ✓ For the convenience of humans, computer input/output devices may accept/provide decimals on the human side and binaries on the computer side.
- ✓ The conversion is simplified by coding each decimal digit, that is, replacing it by the 4-bit binary representation of the digit.
- ✓ For example, in the 8421 BCD code $6_{10} \rightarrow \mathbf{0110}$, $3_{10} \rightarrow \mathbf{0011}$, and $363_{10} \rightarrow \mathbf{0011\ 0110\ 0011}$.
- ✓ **BCD-to-Binary Conversion**. A given BCD number can be converted into an equivalent binary number by first writing its decimal equivalent and then converting it into its binary equivalent.
- ✓ **Binary-to-BCD Conversion**. The process is the same as the process of BCD-to-binary conversion executed in reverse order.

Binary Numbers

Higher-Density BCD Encoding

- ✓ In regular BCD encoding of decimal numbers, the number of bits needed to represent a given decimal number is greater than that required for straight binary encoding of the same.
- ✓ For instance, a three-digit decimal number requires 12 bits for representation in conventional BCD format.
- ✓ However, since $2^{10} > 10^3$, if these three decimal digits are encoded together, only 10 bits would be needed to do that.
- ✓ Two such encoding schemes are Chen-Ho encoding and the densely packed decimal.
- ✓ The latter has the advantage that subsets of encoding encode two digits in the optimal seven bits and one digit in the four bits like regular BCD.

Binary Numbers

Packed and Unpacked BCD Numbers

- ✓ In unpacked BCD, each four-bit BCD group corresponding to a decimal digit is stored in a separate register inside the machine. In this case, if registers are eight bits or wider, the register space is wasted.
- ✓ For packed BCD numbers, two BCD digits are stored in a single eight-bit register. Thus, to store two BCD digits in one eight-bit register, the number in the upper register is shifted to the left 4 times and then added to the number in the lower register.
- ✓ The illustration is shown for storage of decimals 5 and 7.
  - ❑ Decimal digit 5 is initially stored in the 8-bit register as: **0000 0101**.
  - ❑ Decimal digit 7 is initially stored in the 8-bit register as: **0000 0111**.
  - ❑ After shifting to the left 4 times, the digit 5 register reads: **0101 0000**.
  - ❑ The addition of the contents of the digit 5 and 7 registers now reads: **0101 0111**.

Binary Numbers

Excess-3 Code

- ✓ Is another important BCD code. It is particularly significant for arithmetic operations as it overcomes the shortcomings encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9.
- ✓ The excess-3 code has no such limitation, and considerably simplifies arithmetic operations.
- ✓ The excess-3 code for a given decimal number is determined by adding '3' to each decimal digit in the given number and then replacing each digit of the newly found decimal number by its four -bit binary equivalent.
- ✓ If the addition of '3' to a digit produces a carry , as is the case with digits 7, 8, and 9, that carry should not be taken forward. As an example, let us find the excess-3 code for the decimal number 597:
  - ❑ The addition of '3' to each digit yields numbers '8', '12' and '10',
  - ❑ The corresponding four-bit binary equivalents are **1000**, **1100** and **1010.**
  - ❑ The excess-3 code for 597 is therefore, **100011001010**.

Binary Numbers

Converting Excess-3 code to decimal

✓ Table2 lists the excess-3 code for the decimal numbers 0-9.

Table 2 Excess-3 code equivalent of decimal numbers.

Decimal	Excess-3 code	Decimal	Excess-3 code
0	**0011**	5	**1000**
1	**0100**	6	**1001**
2	**0101**	7	**1010**
3	**0110**	8	**1011**
4	**0111**	9	**1100**

✓ It is worth noting that, given excess-3 code, the equivalent decimal number can be determined by splitting the number into 4-bit groups, starting from the radix point, then subtracting **0011** from each 4-bit group.

✓ The new number is the 8421 BCD equivalent of the given excess-3 code, which can subsequently be converted into the equivalent decimal number.

Binary Numbers

Gray Code

- ✓ The Gray code was designed by Frank Gray at Bell Labs and patented in 1953. It is an unweighted binary code in which two successive values differ only by 1 bit.
- ✓ Owing to this feature, the maximum error that can creep into a system using Gray code to encode data is much less than the worst-case error encountered in the case of straight binary encoding. Table 3 list Gray code equivalents of decimal numbers 0 – 15.

Table 3 Gray code.

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

Binary Numbers

Gray Code

- ✓ There are various ways by which Gray codes with a given number of bits can be remembered. One such a way is to remember that the least significant bit follows a repetitive pattern of '2' (11, 00, 11, …), the next higher adjacent bit follows a pattern of '4' (1111, 0000, 1111, …) and so forth.
- ✓ **Binary to Gray Code Conversion steps**

  Step 1. Begin with the most significant bit (MSB). The MSB of the Gray code equivalent is the same as the MSB of the given binary number.

  Step 2. The second MSB in the Gray code number is obtained by adding the MSB and the second MSB in the binary number and ignoring the carry, if any.

  Step 3. The third MSB in the Gray code number is obtained by adding the second MSB and the third MSB in the binary number, ignore carry.

  Step 4. The process continues until we obtain the LSB of Gray code number.

Binary Numbers

Example 5

- ✓ Convert the Binary number **1011** into its Gray code equivalent.
- ✓ **Solution**

Binary	1011
Gray code	**1- - -**
Binary	1011
Gray code	**11**
Binary	1011
Gray code	**111**
Binary	1011
Gray code	**1110**

Binary Numbers

Gray Code

- ✓ **Gray Code to Binary Conversion steps**

Step 1. Begin with the most significant bit (MSB). The MSB of the binary number is the same as the MSB of the Gray code number.

Step 2. The bit next to the MSB (second MSB) in the binary number is obtained by adding the MSB in the binary number to the second MSB in the Gray code number and disregarding the carry, if any.

Step 3. The third MSB in the binary number is obtained by adding the second MSB in the binary number to the third MSB in the Gray code number. Ignored carry.

Step 4. The process continues until we obtain the LSB of binary number.

Binary Numbers

Example 6

- ✓ Convert the Gray code number **1110** into its binary equivalent.
- ✓ **Solution**

Gray code	1110
Binary	**1- - -**
Gray code	1110
Binary	**10- -**
Gray code	1110
Binary	**101**
Gray code	1110
Binary	**1011**

Binary Numbers

Applications of Gray codes

- ✓ The Gray code is used in the transmission of digital signals as it minimizes the occurrence of errors.
- ✓ The Gray code is preferred over the straight binary code in angle-measuring devices.
- ✓ The Gray code is used for labeling Karnaugh maps, a graphical technique used for minimizing of Boolean expressions.
- ✓ The use of Gray codes to address program memory in computers minimizes power consumption. This is due to fewer address lines changing state with advances in the program counter.
- ✓ Gray codes are also useful in genetic algorithms.

Binary Numbers

Alphanumeric Codes

- ✓ When a computer is to handle letters as well as numbers, an alphanumeric code is used. In the American Standard Code for Information Interchange (ASCII), seven bits are used to represent all the characters and punctuation marks on a teletypewriter keyboard plus some control signals.
- ✓ Note that, $2^7 = 128$ combinations of 7 bits. An eighth bit, the MSB, is a parity bit used in error detection. In even parity convention, the MSB is set so that the **1**s in each ASCII character is even; the presence of an odd number of **1**s indicates an error.
- ✓ In reference [2], the ASCII code is listed in a table starting from page 28.
- ✓ The EBCDIC (Extended Binary Coded Decimal Interchange Code) is another widely used alphanumeric code, mainly popular with large systems.
- ✓ It is an 8-bit code and thus can accommodate up to 256 characters. Reference [2] has the details.

End of Lecture 1

Thank you for your attention!