Syllabus (Section A)

**Information & Digital Systems**
- Introduction to digital systems, number systems, weighted and non-weighted codes, code conversion, binary addition and subtraction, 2’s complement method.

**Boolean Algebra & Combinational Logic Circuits**
- Digital logic, Boolean algebra, Boolean function, canonical forms, Karnaugh maps, minimization of Boolean functions, logic gates and their truth tables, design methodologies, combinational logic circuit design, arithmetic and data handling logic circuits, decoder, encoder, multiplexer, demultiplexer.

**Sequential Logic Circuits**
- SR, JK, T, and D flip-flops, master-slave JK flip-flop, timing diagrams of different flip-flops, edge-triggered and level-triggered timing diagrams, counters, registers, memory, finite state machine, asynchronous & synchronous sequential systems, reliable design and fault diagnosis.
Reference Books


Outline

Information & Digital Systems

- Introductory Concept
  - Digital & Analog Systems
  - Advantages & Limitations of Digital Techniques
  - Digital Circuits & Logic Circuits
  - Digital & Parallel Transmission

- Number Systems and Codes
  - Decimal Number System
  - Binary Number System
  - Octal Number System
  - Hexadecimal Number System
  - BCD, ASCII, and Gray Code
  - Conversion from One Number System to Another
  - Parity Method for Error Detection
Introductory Concept

- **Digital System:**
  - A digital system is a combination of devices designed to manipulate logical information or physical quantities that are represented in digital form; that is, the quantities can take on only discrete values. **Example:** Digital computers and calculators, digital telephone system.

- **Analog System:**
  - An analog system contains devices that manipulate physical quantities that are represented in analog/continuous form. **Example:** Audio amplifiers, magnetic tape recording and playback equipment.

**Figure 1.** (a) Analog signal; (b) digital signal.
Introductory Concept

Advantages of Digital Techniques:

- Digital systems are generally easier to design.
- Information storage is easy. Information can be retrieved easily as well.
- Numeric information can be represented digitally with greater precision and range than it can with analog signals. So, digital information are more accurate than the analog.
- In general, digital techniques offer more flexibility than do analog techniques in that they can be more easily programmed to perform any desired algorithm.
- Digital circuits are less affected by noise as digital techniques allow the use of built-in error detection and correction mechanisms.
- Digital circuits provide more powerful processing capabilities in terms of speed.
- More digital circuitry can be fabricated on IC chips.
Introductory Concept

Limitations of Digital Techniques:

• There is really only one major drawback when using digital techniques: “The real world is mainly analog.”

• Most physical quantities are analog in nature, and it is these quantities that are often the inputs and outputs that are being monitored, operated on, and controlled by a system.

• To take advantage of digital techniques, when dealing with analog inputs and outputs, these steps must be followed:
  1) Convert the physical variable to an electrical signal (analog)
  2) Convert the electrical (analog) signal to digital form
  3) Process (operate on) the digital information
  4) Convert the digital outputs back to real-world analog form

• Figure 2 shows a block diagram of this for a typical temperature control system.
**Figure 2.** Diagram of a precision digital temperature control system.

- A user pushes up or down buttons to set the desired temperature. A temperature sensor in the heated space converts the measured temperature to a proportional voltage. The analog voltage is converted to digital quantity by an **analog-to-digital converter (ADC)**.
Introductory Concept

- This value is then compared to the desired value and used to determine a digital value of how much heat is needed.
- The digital value is converted to an analog quantity (voltage) by a digital-to-analog converter (DAC). This voltage is applied to a heating element, which will produce heat that is related to the voltage applied and will effect the temperature of the space.

Digital Signal & Timing Diagram:
- Figure 3 demonstrates the logic levels **HIGH (1)** and **LOW (0)** for 5-volt logic systems that were based on bipolar transistor. The input voltage must be less than 0.8 V to recognize a ‘0’ and between 2 V and 5 V to recognize a ‘1’, as shown in Figure 3(a).
- Figure 3(b) represents a typical digital waveform for the voltage ranges defined in Figure 3(a). Notice that the **HIGH** voltage level between $t_1$ and $t_2$ is 4 V.
Figure 3. Logic levels and timing. (a) typical voltage ranges for a given technology of digital circuits; (b) a graph of signal levels changing over time.

- In digital systems, exact value of a voltage is not important. A **HIGH** voltage of 3.7 V or 4.3 V would represent the exact same information.
Digital Circuits:

- Digital circuits are designed to produce output voltages that fall within the prescribed 0 and 1 voltage ranges such as those defined in Figure 3.

- A digital circuit will respond in the same way to all input voltages that fall within the allowed 0 range; similarly it will not distinguish between input voltages that lie within the allowed 1 range.

- Figure 4 represents a typical digital circuit with input $V_i$ and output $V_0$. The output is shown for two different input signal waveforms. Note that, $V_0$ is the same for both cases because the two input waveforms, while differing in their exact voltage levels, are at the same binary level.
Introductory Concept

Digital Circuits:

Figure 4. A digital circuit responds to an input’s binary level (0 or 1) and not to its actual voltage.
Introductory Concept

Logic Circuits:
• The manner in which a digital circuit responds to an input is referred to as the circuit’s logic. Each type of digital circuits obeys a certain sets of logic rules. For this reason, digital circuits are also called logic circuits.

Serial and Parallel Transmission:
• Tone of the common operations that occur in any digital system is the transmission of information from one place to another. The information that is transmitted is in binary form and is generally represented as voltages at the outputs of a sending circuit that are connected to the inputs of a receiving circuit.
• Figure 5 illustrates the two basic methods for digital information transmission: parallel and serial. Figure 5(a) shows how the binary number 01101001 is transmitted from the computer to a printer using parallel transmission.
Introduction Concept

Figure 5. (a) Parallel transmission uses one connecting line per bit and all bits are transmitted simultaneously; (b) serial transmission uses only one signal line, and the individual bits are transmitted serially one at a time.  

- Figure 5(b) shows that there is only connection between the computer and printer when serial transmission is used. One bit is transmitted serially per time interval. The LSB is transmitted first for serial transmission.
Number Systems and Codes

Digital Number Systems:

- Many number systems are in use in digital technology. The most common are the decimal, binary, octal, and hexadecimal systems.

Decimal Number System:

- The decimal system is composed of 10 number or symbols. These 10 symbols are: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9; using these symbols as digits of a number, we can express any quantity. Decimal system, also called base-10 system has 10 digits.

- Decimal system is a positional-value system in which the value of a digit depends on its position. More rigorously, the various positions relative to the decimal point carry weights that can be expressed as powers of 10, as illustrated in Figure 6. The places to the left of decimal point are positive powers of 2 and places to the right are negative powers of 2. The number 2745.214 is thus equal to:

\[
2745.214 = (2 \times 10^3) + (7 \times 10^2) + (4 \times 10^1) + (5 \times 10^0) + (2 \times 10^{-1}) + (1 \times 10^{-2}) + (4 \times 10^{-3})
\]
Digital Number Systems:

**Decimal Number System:**

- Figure 6. Decimal position values as powers of 10.

- In general, any number is simply the sum of the products of each digit value and its positional value. The leftmost digit is the **Most Significant Digit (MSD)** & rightmost digit is the **Least Significant Digit (LSD)**. The decimal point separates the positive powers of 10 from the negative powers.
Number Systems and Codes

☐ Digital Number Systems:

✓ Decimal Number System:-

• In general, with N places or digits we can count through 10^N different numbers, starting with and including zero. The largest number will always be 10^N – 1.

✓ Binary Number System:-

• In the binary system there are only two symbols or possible digit values, 0 and 1. Almost every digital systems uses the binary (base-2) number system as the basic number system of its operations, although other systems are often used in conjunction with binary.

• Even so, this base-2 system can be used to represent any quantity that can be represented in decimal or other number systems.

• Binary system is also a positional-value system, wherein each binary digit has its own value or weight expressed as a power of 2.
Number Systems and Codes

- **Digital Number Systems:**
  - **Binary Number System:**
    - Figure 7 represents the binary number system where places to the left of the binary point (counter part of the decimal point) are positive powers of 2 & places to the right are negative powers of 2.

![Figure 7. Binary position values as powers of 2.](image-url)
Number Systems and Codes

❑ Digital Number Systems:

✓ Binary Number System:-

• The number 1011.101 is represented in Figure 7. To find its equivalent in the decimal system are simply the sum of the products of each digit value (0 or 1) and its positional value:

\[
1011.101_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) \\
= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125 = 11.625_{10}
\]

✓ In binary system, the term binary digit is often abbreviated to the term bit, which we will use from now on. The Most Significant Bit (MSB) is the leftmost bit (largest weight) and the Least Significant Bit (LSB) is the right most bit (smallest weight).

✓ Binary Counting:-

• Let’s use 4-bit binary numbers to illustrate the method for counting in binary. Let’s use 4-bit binary numbers to illustrate the method for counting in binary. The sequence, shown in Figure 8, begins with all bits at 0; it is called the zero count.
Digital Number Systems:

✓ Binary Counting:

**Figure 8.** Binary counting sequence.
Digital Number Systems:

Binary Counting:

- For each successive count, the units ($2^0$) position toggles; that is, it changes from one binary value to the other. Each time the units bit changes from a 1 to 0, the twos ($2^1$) position will toggle (change states).

- Each time the twos position changes from a 1 to a 0, the fours ($2^2$) position will toggle (change states). Like-wise, each time the fours position goes from a 1 to 0, the eights ($2^3$) position toggles. This same process would be continued for the higher order bit positions if the binary number had more than 4 bits.

- For binary system, by using $N$ bits or places, we can go through $2^N$ counts. For example, with two bits we can go through $2^2 = 4$ counts (00$_2$ through 11$_2$). The last count will always be all 1s and is equal to $2^N - 1$ in the decimal system. For example, using 4 bits, the last count is 1111$_2 = 2^4 - 1 = 15_{10}$. 
Number Systems and Codes

- **Digital Number Systems:**

  - **Octal Number System:**

    The octal number system is often used in digital computer work. The octal number system has a base of eight, meaning that it has eight possible digits: 0, 1, 2, 3, 4, 5, 6, and 7. The digit positions in an octal number have weights as follows:

    ![Octal position values (weights) as powers of 8.](image)

    **Figure 9.** Octal position values (weights) as powers of 8.
Number Systems and Codes

- Digital Number Systems:
  - Octal Number System:
    - To find the equivalent of octal system in the decimal system, we simply take the sum of the products of each digit value and its positional value:
      \[ 2745.214_{8} = 2 \times (8^3) + 7 \times (8^2) + 4 \times (8^1) + 5 \times (8^0) + 2 \times (8^{-1}) + 1 \times (8^{-2}) + 4 \times (8^{-3}) = 1,509.2734375_{10} \]
  - Hexadecimal Number System:
    - The hexadecimal number system uses base-16. Thus, it has 16 possible digit symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The digit position in a hexadecimal number value weights as shown in Figure 10. It is important to note that, hexadecimal digits A through F are equivalent to the decimal values 10 through 15.
    - To find the equivalent of hexadecimal system in the decimal system, we simply take the sum of the products of each digit value and its positional value:
      \[(B6F.5)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 + 5 \times 16^{-1} = (46687.0625)_{10} \]
Digital Number Systems:

Hexadecimal Number System:

Figure 10. Hexadecimal position values (weights) as power of 16.

Table 1 shows the relationship among hexadecimal, decimal, octal, and binary number systems.
## Number Systems and Codes

### Digital Number Systems:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Hexadecimal</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>02</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>3</td>
<td>03</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>4</td>
<td>04</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>5</td>
<td>05</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>6</td>
<td>06</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>7</td>
<td>07</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>A</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>B</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>C</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>D</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>E</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>F</td>
<td>17</td>
</tr>
</tbody>
</table>
Number Systems and Codes

Binary Arithmetic:

- Binary Addition:
  - Only four cases can occur in adding two binary digits (bits) in any position. They are: $0 + 0 = 0$; $1 + 0 = 1$; $1 + 1 = 10 = 0 + \text{carry of 1 into next position}$; $1 + 1 + 1 = 11 = 1 + \text{carry of 1 into next position}$.
  - **Example:** Add $11.011_2$ with $10.110_2$.

\[
\begin{array}{cccccc}
1 & 1 & 1 & & & \\
1 & 1 & . & 0 & 1 & 1 & \text{(3.375) Augend} \\
+ & 1 & 0 & . & 1 & 1 & 0 & \text{(2.750) Addend} \\
\hline
1 & 1 & 0 & . & 0 & 0 & 1 & \text{(6.125) Sum} \\
\end{array}
\]
Number Systems and Codes

- **Binary Arithmetic:**

  - **Binary Subtraction:**
    - Binary subtraction may be thought of as the inverse of binary addition. The rules of binary subtraction are: $1 - 0 = 1$; $1 - 1 = 0$; $0 - 0 = 0$; $0 - 1 = 1$ with a borrow of 1; or $10 - 1 = 1$.
    - The last rule shows that if a 1 bit is subtracted from a 0 bit, then a 1 must be borrowed from the next most significant column. Borrows propagate to the left from column to column, as illustrated next.

  - **Example:** Subtract $10111_2$ from $1001101_2$.

    | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
    |---|---|---|---|---|---|---|
    |   |   |   |   |   |   | 10|
    |   |   |   |   |   | 10| 0 |
    |   |   |   |   | 10| 0 | 10|
    |   | 10| 0 | 1 | 1 | 0 | 1 |
    |   |   |   |   |   |   | 0 |
    |   |   |   |   |   | 1 | 1 |

    Minuend: $1001101_2$
    Subtrahend: $10111_2$
    Minus: 10111
    Answer: 110010
Number Systems and Codes

- **Binary Arithmetic:**
  - **Binary Subtraction:**
    - In this example, a borrow is first encountered in column 1. The borrow is taken from column 2, resulting in a 10 in column 1 and a 0 in column 2. The 0 now present in column 2 necessitates a borrow from column 3. No other borrows are necessary until column 4.
    - In this case, there is no 1 in column 5 to borrow. Hence we must first borrow the 1 from column 6, which results in 0 in column 6 and 10 in column 5. Now column 4 borrows a 1 from column 5, leaving 1 in column 5 \((10 - 1 = 1)\) and 10 in column 4. This sequence of borrows is shown in the minuend terms of the example.
Number Systems and Codes

❑ Binary Arithmetic:

✓ Binary Multiplication:

• The multiplication of binary numbers is done in the same manner as the multiplication of decimal numbers. The following example illustrates for unsigned binary numbers:

• Example: Multiply 10111\textsubscript{2} with 1011\textsubscript{2}.

\[
\begin{array}{c}
1 & 0 & 1 & 1 & 1 \\
\times & 1 & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 \\
\underline{1 & 0 & 1 & 1 & 1} \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
\]

Multiplicand \((23\textsubscript{10})\)

Multiplier \((11\textsubscript{10})\)

Final Product \((253\textsubscript{10})\)
Number Systems and Codes

- **Binary Arithmetic:**

  - **Binary Division:**
    - The process of dividing one binary number (the dividend) by another number (the divisor) is the same as that which is followed for decimal numbers, that which we usually refer to as “long division.” Rules are: \(0 \div 1 = 0; \ 1 \div 1 = 1\)
    
    - **Example:** Divide \(11011_2\) by \(100_2\).

      \[
      \begin{array}{c|c|c}
      & 1 & 1 0 . 1 1 \\
      1 0 0 & 1 & 1 0 1 1 \\
      \hline
      & 1 \ 0 \ 0 \\
      1 0 0 & 1 & 0 1 \\
      \hline
      & 1 \ 0 0 \\
      1 0 0 & 1 & 1 0 \\
      \hline
      & 1 \ 0 0 \\
      1 0 0 & 1 & 0 0 \\
      \hline
      & 0 0 0
      \end{array}
      \]

      \[\therefore \text{The resulting quotient} = 110.11_2\]
Number Systems and Codes

- **Binary to Decimal Conversion:**
  - Any binary number can be converted to its decimal equivalent simply by summing together the weights of the various positions in the binary number which contain a 1.
  - **Example:** Convert $1101.011_2$ to its corresponding decimal value.
    \[
    1101.011_2 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3})
    \]
    \[
    = 8 + 4 + 0 + 1 + 0 + 0.25 + 0.125 = 13.375_{10}
    \]

- **Decimal to Binary Conversion:**
  - (a) **Integers:**
    - Such conversion can be achieved by using the so-called double-dabble method. This is also known as divide-by-two method. In this method, we progressively divide the given decimal number by 2 and write down the remainders after each division.
    - This remainders taken in the reverse order (i.e. from bottom-to-top) form the required binary number.
Number Systems and Codes

Decimal to Binary Conversion:

- (a) Integers:

**Example:** Convert $25_{10}$ into its binary equivalent.

<table>
<thead>
<tr>
<th>Successive Divisions</th>
<th>Quotients</th>
<th>Remainders</th>
</tr>
</thead>
<tbody>
<tr>
<td>$25 \div 2$</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>$12 \div 2$</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>$6 \div 2$</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$3 \div 2$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$1 \div 2$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$\therefore 25_{10} = 11001_2$
Number Systems and Codes

Decimal to Binary Conversion:

(a) Fractions:

- In this case, multiply-by-two rule is used i.e. we multiply each number by 2 and recorded the carry in the integer position. These carries taken in the forward (top-to-bottom) direction give the required binary fraction.

Example: Convert 0.8125\textsubscript{10} into its binary equivalent.

<table>
<thead>
<tr>
<th>Successive Multiplications</th>
<th>Fractions</th>
<th>Carries</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8125 \times 2 = 1.625</td>
<td>0.625</td>
<td>1</td>
</tr>
<tr>
<td>0.625 \times 2 = 1.25</td>
<td>0.25</td>
<td>1</td>
</tr>
<tr>
<td>0.25 \times 2 = 0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>0.5 \times 2 = 1.0</td>
<td>0.0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \therefore \quad 0.8125\textsubscript{10} = 0.1101\textsubscript{2} \]

\[ \therefore \quad 25.8125\textsubscript{10} = 11001.1101\textsubscript{2} \]
Octal to Decimal Conversion:

- Any octal number can be converted to its decimal equivalent simply by summing together the products of each digit value of the octal number and its positional value.

- **Example:** Convert $372.6_8$ into its decimal equivalent.

  $372.6_8 = (3 \times 8^2) + (7 \times 8^1) + (2 \times 8^0) + (6 \times 8^{-1}) = 192 + 56 + 2 + 0.75 = 250.75_{10}$

Decimal to Octal Conversion:

- The double-dabble method is used with 8 acting as the multiplying factor for fractions and the dividing factor for integers.

- **Example:** Convert $175.15_{10}$ into its octal equivalent.

  - (a) Integer:-

    | Successive Divisions | Quotients | Remainders |
    |----------------------|-----------|------------|
    | 175 ÷ 8              | 21        | 7          |
    | 21 ÷ 8               | 2         | 5          |
    | 2 ÷ 8                | 0         | 2          |

    $\therefore 175_{10} = 257_8$
Number Systems and Codes

Decimal to Octal Conversion:

Example (continued): Convert $175.15_{10}$ into its octal equivalent.

(a) Fraction:

<table>
<thead>
<tr>
<th>Successive Multiplications</th>
<th>Fractions</th>
<th>Carries</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.15 \times 8 = 1.20$</td>
<td>0.20</td>
<td>1</td>
</tr>
<tr>
<td>$0.20 \times 8 = 1.60$</td>
<td>0.60</td>
<td>1</td>
</tr>
<tr>
<td>$0.60 \times 8 = 4.80$</td>
<td>0.80</td>
<td>4</td>
</tr>
<tr>
<td>$0.80 \times 8 = 6.40$</td>
<td>0.40</td>
<td>6</td>
</tr>
<tr>
<td>$0.40 \times 8 = 3.20$</td>
<td>0.20</td>
<td>3</td>
</tr>
</tbody>
</table>

We can stop here but the answer would be approximate.

$0.15_{10} \approx 0.11463_8$

$175.15_{10} \approx 257.11463_8$
Number Systems and Codes

- Hexadecimal to DecimalConversion:
  - Any Hexadecimal number can be converted to its decimal equivalent simply by summing together the products of hexadecimal digits and their weights to get the decimal equivalent.
  - **Example:** Convert F6D9.1\textsubscript{16} into its decimal equivalent.

\[
F6D9.1\textsubscript{16} = (F \times 16^3) + (6 \times 16^2) + (D \times 16^1) + (9 \times 16^0) + (1 \times 16^{-1}) \\
= (15 \times 16^3) + (6 \times 16^2) + (13 \times 16^1) + (9 \times 16^0) + (1 \times 16^{-1}) \\
= 61440 + 1536 + 208 + 9 + 0.0625 = 63193.0625\textsubscript{10}
\]
Number Systems and Codes

Decimal to Hexadecimal Conversion:

- The double-dabble method is used with 16 acting as the multiplying factor for fractions and dividing factor for integers.

- **Example:** Convert $2803.08203125_{10}$ into its hexadecimal equivalent.

(a) Integer:-

<table>
<thead>
<tr>
<th>Successive Divisions</th>
<th>Quotients</th>
<th>Remainders</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$2803 \div 16$</td>
<td>175</td>
<td>3</td>
<td>Top</td>
</tr>
<tr>
<td>$175 \div 16$</td>
<td>10</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$10 \div 16$</td>
<td>0</td>
<td>A</td>
<td>Bottom</td>
</tr>
</tbody>
</table>

∴ $2803_{10} = AF3_{16}$
### Decimal to Hexadecimal Conversion:

- **Example (Continued):** Convert $2803.08203125_{10}$ into its hexadecimal equivalent.

- **(a) Fractions:**

<table>
<thead>
<tr>
<th>Successive Multiplications</th>
<th>Fractions</th>
<th>Carries</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.08203125 \times 16 = 1.3125$</td>
<td>$0.3125$</td>
<td>$1$ Top</td>
</tr>
<tr>
<td>$0.3125 \times 16 = 5.00$</td>
<td>$0.00$</td>
<td>$5$ Bottom</td>
</tr>
</tbody>
</table>

\[ \therefore 0.08203125_{10} = 0.15_{16} \]

\[ \therefore 2803.08203125_{10} = AF3.15_{16} \]
Number Systems and Codes

Octal to Binary Conversion:

- The conversion from octal to binary is performed by converting each octal digit to its three-bit binary equivalent. The eight possible digits are converted as indicated in Table 2.

Table 2. Octal to Binary Conversion.

<table>
<thead>
<tr>
<th>Octal Digit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>000</td>
<td>001</td>
<td>010</td>
<td>011</td>
<td>100</td>
<td>101</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

- Example: Convert 673.124₈ into its binary equivalent.

\[
673.124₈ = (110\underline{111} 011 . \underline{001} \underline{010} 100)_2
\]

\[
\begin{array}{ccccccc}
6 & 7 & 3 & . & 1 & 2 & 4 \\
\end{array}
\]
Number Systems and Codes

- **Binary to Octal Conversion:**

  - Converting from binary to octal is simply the reverse of the foregoing process. The conversion from binary to octal is easily accomplished by partitioning the binary number into groups of three digits each, starting from the binary point and proceeding to the left and to the right.

  - **Example:** Convert \(110110001101011.111100000110_2\) into its octal equivalent.

  \[
  (110 \ 110 \ 001 \ 101 \ 011 \ . \ 111 \ 100 \ 000 \ 110)_2 = 66153.7406_8
  \]

  - Sometimes the binary number will not have even groups of three bits. For those cases, we can one or two 0s to the left of the MSB of the binary number to fill out the last group (for the left part from the binary point). For the right part from the binary point, we can one or two 0s to the right of the LSB of the binary number to fill out the last group.
Number Systems and Codes

- **Hexadecimal to Binary Conversion:**
  - The conversion from hexadecimal to binary is performed by converting each hex digit to its four-bit binary equivalent. The sixteen possible digits are converted as indicated in **Table 3:**

  **Table 3. Hexadecimal to Binary Conversion.**

<table>
<thead>
<tr>
<th>Hexadecimal Digit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Equivalent</td>
<td>0000</td>
<td>0001</td>
<td>0010</td>
<td>0011</td>
<td>0100</td>
<td>0101</td>
<td>0110</td>
<td>0111</td>
</tr>
<tr>
<td>Hexadecimal Digit</td>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>Binary Equivalent</td>
<td>1000</td>
<td>1001</td>
<td>1010</td>
<td>1011</td>
<td>1100</td>
<td>1101</td>
<td>1110</td>
<td>1111</td>
</tr>
</tbody>
</table>

- **Example:** Convert \(306.D_{16}\) into its binary equivalent.

\[
306.D_{16} = (0011\ 0000\ 0110\ .\ 1101)_2
\]

\[
3 \quad 0 \quad 6 \quad D
\]
Number Systems and Codes

- **Binary to Hexadecimal Conversion:**
  - Conversion from binary to hexadecimal is just the reverse of the process of converting from hexadecimal to binary number. The binary number is grouped into groups of four bits, and each group is converted to its equivalent hex digit. The groups are done by starting from the binary point and proceeding to the left and to the right. 0s are added, if needed, to complete a four-bit group.
  - **Example:** Convert $10110001101011.1111001_2$ into its hexadecimal equivalent.

\[
(0010 \ 1100 \ 0110 \ 1011 \ . \ 1111 \ 0010)_2 = 2C6B.F2_{16}
\]

- Note that two 0s are placed to the left of the MSB and one 0 is placed to the right of the LSB of the binary number to produce even groups of four bits.
Hexadecimal to Octal Conversion:

- The easiest way to convert a hexadecimal number to its corresponding octal number is to convert the hexadecimal number to binary number first, then to octal number.

- **Example:** Convert \( AF.C_{16} \) into its octal equivalent.

At first we will convert the hexadecimal number to binary number.

\[
AF.C_{16} = (1010 \ 1111 \ . \ 0001 \ 0110 \ 1100)_2
\]

Now we will convert the binary number to octal number.

\[
(010 \ 101 \ 111.000 \ 101 \ 101 \ 100)_2 = 257.0554_8
\]

\[\therefore \ AF.C_{16} = 257.0554_8\]
Number Systems and Codes

Octal to Hexadecimal Conversion:

- The easiest way to convert an octal number to its corresponding hexadecimal number is to convert the octal number to binary number first, then to hexadecimal number.

- **Example:** Convert $5457.0554_8$ into its hexadecimal equivalent.

At first we will convert the octal number to binary number.

$$5457.0554_8 = (101\ 100\ 101\ 111\ .\ 000\ 101\ 101\ 100)_2$$

$$\begin{align*} 5 & \ 4 & \ 5 & \ 7 & \ 0 & \ 5 & \ 5 & \ 4 \\
\end{align*}$$

Now we will convert the binary number to hexadecimal number.

$$\begin{align*} (1011\ 0010\ 1111\ .\ 0001\ 0110\ 1100)_2 &= B2F.16C_{16} \\
B & \ 2 & \ F & \ 1 & \ 6 & \ C \\
\end{align*}$$

$\therefore \ 5457.0554_8 = B2F.16C_{16}$
Number Systems and Codes

Code & Binary Coding:

• When numbers, letters, or words are represented by a special group of symbols, we say that they are being encoded, and the group of symbols is called a code.

• We have seen that any decimal can be represented by an equivalent binary number. The groups of 0s and 1s in the binary number can be thought of as a code representing the decimal number.

• When a decimal number is represented by its equivalent binary number, we call it straight binary coding.

• We have seen that the conversions between any binary can become long and complicated for large numbers. For this reason, a means of encoding decimal numbers that combines some features of both the decimal and the binary system is used in certain solution.
Binary Coded Decimal (BCD) Code:

- If each digit of a decimal number is represented by its binary equivalent, the result is a code called **Binary Coded Decimal (BCD)**; since a decimal digit can be as large as 9, four bits are required to code each digit (the binary code for 9 is 1001).

- To illustrate the BCD code, take a decimal number such as 874. Each digit is changed to its binary equivalent as follows:

  \[
  \begin{array}{ccc}
    8 & 7 & 4 \\
    1000 & 0111 & 0100
  \end{array}
  \]

- As another example, let us change 943 to its BCD code representation.

  \[
  \begin{array}{ccc}
    9 & 4 & 3 \\
    1001 & 0100 & 0011
  \end{array}
  \]
Number Systems and Codes

- **Binary Coded Decimal (BCD) Code:**
  - The BCD code, then represents each digit of the decimal number by a four-bit binary number. Clearly, only the four-bit binary numbers from 0000 through 1001 are used.
  - The BCD code doesn’t use the numbers 1010, 1011, 1100, 1101, 1110, and 1111. If any of the “forbidden” four-bit numbers ever occurs in a machine using the BCD code, it is usually an indication that an error has occurred.

- **Comparison of BCD and Binary:**
  - It is important to realize that BCD is not another number system like binary, octal, decimal, and hexadecimal. It is in fact, the decimal system with each digit encoded in its binary equivalent.
  - It is also important to understand that, a BCD number is not the same as a straight binary number. A straight binary code takes the complete decimal number and represents it in binary; the BCD code converts each decimal digit to binary individually.
Comparison of BCD and Binary:

- To illustrate, take the number 137 and compare its straight binary and BCD codes:

  \[137_{10} = 10001001_2 \quad \text{(Binary)}\]
  \[137_{10} = 0001\ 0011\ 0111 \quad \text{(BCD)}\]

- The BCD code requires 12 bits while straight binary code requires only eight bits to represent \(137_{10}\).

- The main advantage of the BCD code is the relative ease of converting to and from decimal. Only the 4-bit code groups for the decimal digits 0 to 9 need to be remembered.

- This ease of conversion is especially important from a hardware standpoint, because in a digital system it is the logic circuits that perform the conversions to and from decimal.
Alphanumeric Codes:

- In addition to numerical data, a computer must be able to handle non-numerical information.
- In other words, a computer should recognize codes that represent letters of the alphabet, punctuation marks, and other special characters as well as numbers. These codes are called Alphanumeric Codes.
- A complete alphanumeric code would be the 26 lowercase letters, 26 uppercase letters, 10 numeric digits, 7 punctuation marks, and anywhere from 20 to 40 other characters, such as +, /, #, ÷, ., and so on.
- We can say that an alphanumeric code represents all of the various characters and functions that are found on a computer keyboard.
Alphanumeric Codes:

- **ASCII Code:**
  
  - The most widely used alphanumeric code is the American Standard Code for Information Interchange (ASCII).
  
  - The ASCII (askee) code is 7-bit code, and so it has $2^7 = 128$ possible code groups. This is more than enough to represent all of the standard keyboard characters as well as control functions such as the `<RETURN>` and `<LINE FEED>` functions.
  
  - **Table 4** shows a partial listing of the ASCII code.
Number Systems and Codes

- **Alphanumeric Codes:**
  - **ASCII Code:**

Table 4. Standard ASCII codes.

<table>
<thead>
<tr>
<th>Character</th>
<th>Hex</th>
<th>Decimal</th>
<th>Character</th>
<th>Hex</th>
<th>Decimal</th>
<th>Character</th>
<th>Hex</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL (null)</td>
<td>0</td>
<td>0</td>
<td>Space</td>
<td>20</td>
<td>32</td>
<td>@</td>
<td>40</td>
<td>64</td>
</tr>
<tr>
<td>Start Heading</td>
<td>1</td>
<td>1</td>
<td>!</td>
<td>21</td>
<td>33</td>
<td>a</td>
<td>61</td>
<td>97</td>
</tr>
<tr>
<td>Start Text</td>
<td>2</td>
<td>2</td>
<td>&quot;</td>
<td>22</td>
<td>34</td>
<td>b</td>
<td>62</td>
<td>98</td>
</tr>
<tr>
<td>End Text</td>
<td>3</td>
<td>3</td>
<td>#</td>
<td>23</td>
<td>35</td>
<td>c</td>
<td>63</td>
<td>99</td>
</tr>
<tr>
<td>End Transmit.</td>
<td>4</td>
<td>4</td>
<td>$</td>
<td>24</td>
<td>36</td>
<td>d</td>
<td>64</td>
<td>100</td>
</tr>
<tr>
<td>Enquiry</td>
<td>5</td>
<td>5</td>
<td>%</td>
<td>25</td>
<td>37</td>
<td>e</td>
<td>65</td>
<td>101</td>
</tr>
<tr>
<td>Acknowlege</td>
<td>6</td>
<td>6</td>
<td>&amp;</td>
<td>26</td>
<td>38</td>
<td>f</td>
<td>66</td>
<td>102</td>
</tr>
<tr>
<td>Bell</td>
<td>7</td>
<td>7</td>
<td>'</td>
<td>27</td>
<td>39</td>
<td>g</td>
<td>67</td>
<td>103</td>
</tr>
<tr>
<td>Backspace</td>
<td>8</td>
<td>8</td>
<td>(</td>
<td>28</td>
<td>40</td>
<td>h</td>
<td>68</td>
<td>104</td>
</tr>
<tr>
<td>Horiz. Tab</td>
<td>9</td>
<td>9</td>
<td>)</td>
<td>29</td>
<td>41</td>
<td>i</td>
<td>69</td>
<td>105</td>
</tr>
<tr>
<td>Line Feed</td>
<td>A</td>
<td>10</td>
<td>'</td>
<td>2A</td>
<td>42</td>
<td>j</td>
<td>6A</td>
<td>106</td>
</tr>
<tr>
<td>Vert. Tab</td>
<td>B</td>
<td>11</td>
<td>+</td>
<td>2B</td>
<td>43</td>
<td>k</td>
<td>6B</td>
<td>107</td>
</tr>
<tr>
<td>Form Feed</td>
<td>C</td>
<td>12</td>
<td>,</td>
<td>2C</td>
<td>44</td>
<td>l</td>
<td>6C</td>
<td>108</td>
</tr>
<tr>
<td>Carriage Return</td>
<td>D</td>
<td>13</td>
<td>-</td>
<td>2D</td>
<td>45</td>
<td>m</td>
<td>6D</td>
<td>109</td>
</tr>
<tr>
<td>Shift Out</td>
<td>E</td>
<td>14</td>
<td>.</td>
<td>2E</td>
<td>46</td>
<td>n</td>
<td>6E</td>
<td>110</td>
</tr>
<tr>
<td>Shift In</td>
<td>F</td>
<td>15</td>
<td>/</td>
<td>2F</td>
<td>47</td>
<td>o</td>
<td>6F</td>
<td>111</td>
</tr>
<tr>
<td>Data Link Esc</td>
<td>10</td>
<td>16</td>
<td>0</td>
<td>30</td>
<td>48</td>
<td>p</td>
<td>70</td>
<td>112</td>
</tr>
<tr>
<td>Direct Control 1</td>
<td>11</td>
<td>17</td>
<td>1</td>
<td>31</td>
<td>49</td>
<td>q</td>
<td>71</td>
<td>113</td>
</tr>
<tr>
<td>Direct Control 2</td>
<td>12</td>
<td>18</td>
<td>2</td>
<td>32</td>
<td>50</td>
<td>r</td>
<td>72</td>
<td>114</td>
</tr>
<tr>
<td>Direct Control 3</td>
<td>13</td>
<td>19</td>
<td>3</td>
<td>33</td>
<td>51</td>
<td>s</td>
<td>73</td>
<td>115</td>
</tr>
<tr>
<td>Direct Control 4</td>
<td>14</td>
<td>20</td>
<td>4</td>
<td>34</td>
<td>52</td>
<td>t</td>
<td>74</td>
<td>116</td>
</tr>
<tr>
<td>Negative ACK</td>
<td>15</td>
<td>21</td>
<td>5</td>
<td>35</td>
<td>53</td>
<td>u</td>
<td>75</td>
<td>117</td>
</tr>
<tr>
<td>Synch Idle</td>
<td>16</td>
<td>22</td>
<td>6</td>
<td>36</td>
<td>54</td>
<td>v</td>
<td>76</td>
<td>118</td>
</tr>
<tr>
<td>End Trans Block</td>
<td>17</td>
<td>23</td>
<td>7</td>
<td>37</td>
<td>55</td>
<td>w</td>
<td>77</td>
<td>119</td>
</tr>
<tr>
<td>Cancel</td>
<td>18</td>
<td>24</td>
<td>8</td>
<td>38</td>
<td>56</td>
<td>x</td>
<td>78</td>
<td>120</td>
</tr>
<tr>
<td>End of Medium</td>
<td>19</td>
<td>25</td>
<td>9</td>
<td>39</td>
<td>57</td>
<td>y</td>
<td>79</td>
<td>121</td>
</tr>
<tr>
<td>Substitute</td>
<td>1A</td>
<td>26</td>
<td>;</td>
<td>3A</td>
<td>58</td>
<td>z</td>
<td>7A</td>
<td>122</td>
</tr>
<tr>
<td>Escape</td>
<td>1B</td>
<td>27</td>
<td>:</td>
<td>3B</td>
<td>59</td>
<td>{</td>
<td>7B</td>
<td>123</td>
</tr>
<tr>
<td>Form Separator</td>
<td>1C</td>
<td>28</td>
<td>&lt;</td>
<td>3C</td>
<td>60</td>
<td></td>
<td></td>
<td>7C</td>
</tr>
<tr>
<td>Group Separator</td>
<td>1D</td>
<td>29</td>
<td>=</td>
<td>3D</td>
<td>61</td>
<td>}</td>
<td>7D</td>
<td>125</td>
</tr>
<tr>
<td>Record Separator</td>
<td>1E</td>
<td>30</td>
<td>&gt;</td>
<td>3E</td>
<td>62</td>
<td>^</td>
<td>7E</td>
<td>126</td>
</tr>
<tr>
<td>Unit Separator</td>
<td>1F</td>
<td>31</td>
<td>?</td>
<td>3F</td>
<td>63</td>
<td>_</td>
<td>5F</td>
<td>95</td>
</tr>
</tbody>
</table>
Number Systems and Codes

- **Alphanumeric Codes:**
  - **Cyclic Code:**
    - A cyclic code may be defined as any code in which, for any code word, a circular shift produces another code word.
  - **Gray Code:**
    - The Gray Code is one of the most common types of cyclic codes & has the characteristic that the code words for two consecutive numbers differ in only 1 bit. Thus, the distance between the two code words is 1.

- **Define a Gray Code for Encoding the Decimal Numbers 0 to 15:**
  - Four bits are needed to represent all the numbers, and the necessary code can be constructed by assigning bit i of the code word to be 0 if bits i and i + 1 of the corresponding binary number are the same and 1 otherwise. The most significant bit of the number must always be compared with 0 when using this technique. The resulting code is given in Table 5.
Gray Code for Encoding the Decimal Numbers 0 to 15:

Table 5. Gray Code for Decimal Numbers 0 to 15.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Gray</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0011</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>0010</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0110</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>0111</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>0101</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>0100</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1100</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>1101</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>1111</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>1110</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>1010</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>1011</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>1001</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>1000</td>
</tr>
</tbody>
</table>
Number Systems and Codes

- **Alphanumeric Codes:**
  - **Binary to Gray Code Conversion:**
    - **Example:** Convert a binary 1011 to Gray Code.
    - **Process:**
      - (A) Record the most significant bit.
      - (B) Add this bit to the next position, recording the sum and neglecting any carry.
      - (C) Record successive sums until completed.
    - First record the 8’s (2³) bit. Then add the 8’s bit to the 4’s bit (2²) (1 + 0 = 1). Record it. Then add the 4’s bit to the 2’s bit (2¹) (0 + 1 = 1). Then add the 2’s bit to the 1’s bit (2⁰) (1 + 1 = 10; ignore carry).
    - The result is 1110.
Number Systems and Codes

Alphanumeric Codes:

Gray Code to Binary Conversion:

- **Example:** Convert the Gray Code 1011 to binary.

- **Process:**
  - (A) Record the most significant bit (MSB).
  - (B) Add the binary MSB to the next significant bit of the Gray Code.
  - (C) Record the result, ignoring carries.

First, record the MSB. Then add the MSB to the 4’s bit \(2^2\) of the Gray number \((1 + 0 = 1)\). Then add the 4’s bit of the binary number with the 2’s bit \(2^1\) of the Gray Code \((1 + 1 = 10\); ignore carry). Record the 0. Then add the 2’s bit \(2^1\) of the binary code to the 1’s bit \(2^0\) of the Gray Code \((0 + 1 = 1)\) and record the sum.

- **The result is 1101.**
Number Systems and Codes

Convert Base-A to Base-B:

• To convert a number N from Base-A to Base-B, first convert the number of Base-A to decimal then convert the decimal number to Base-B.

• Example: Convert \((18.6)_9\) to \((?)_{11}\).

Converting \((18.6)_9\) to base-10 via series substitution yields:

\[
18.6_9 = 1 \times 9^1 + 8 \times 9^0 + 6 \times 9^{-1} = 9 + 8 + 0.666\ldots = 17.666\ldots_{10}
\]

Converting the decimal number \((17.666\ldots)_{10}\) to Base-11:

(a) Integer:-

<table>
<thead>
<tr>
<th>Successive Divisions</th>
<th>Quotients</th>
<th>Remainders</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 ÷ 11</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1 ÷ 11</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[\therefore 17_{10} = 16_{11}\]
### Convert Base-A to Base-B:

- **Example (continued):** Convert \((18.6)_9\) to \((?)_{11}\).

#### (b) Fraction:-

<table>
<thead>
<tr>
<th>Successive Multiplications</th>
<th>Fractions</th>
<th>Carries</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.666 \times 11 = 7.326)</td>
<td>(0.326)</td>
<td>(7)</td>
</tr>
<tr>
<td>(0.326 \times 11 = 3.586)</td>
<td>(0.586)</td>
<td>(3)</td>
</tr>
<tr>
<td>(0.586 \times 11 = 6.446)</td>
<td>(0.446)</td>
<td>(6)</td>
</tr>
</tbody>
</table>

\[\therefore 0.666\ldots_{10} = 0.736_{11}\]

\[\therefore (17.666\ldots)_{10} = (16.736\ldots)_{11}\]

\[\therefore (18.69)_9 = (16.736\ldots)_{11}\]
Number Systems and Codes

Parity Method for Error Detection:

- The movement of binary data and codes from one location to another is the most frequent operation performed in digital systems. Here are just a few examples.
  - (1) The transmission of a digitized voice over a microwave link
  - (2) The storage of data in and retrieval of data from external memory devices such as magnetic or disk
  - (3) Transmission of digital data from a computer to a remote computer over telephone lines (i.e. using a modem)

- Whenever information is transmitted from one device (the transmitter) to another device (the receiver), there is a possibility that errors can occur such that the receiver does not receive the identical information that was sent by the transmitter.

- The major cause of any transmission errors is electrical noise, which consists of spurious fluctuations in voltage or current that are present in all electronic systems to varying degrees. Occasionally, the noise is large enough in amplitude that it will alter the logic level of the signal. When this occurs, the receiver may incorrectly interpret that bit (0 in place of 1 or vice versa).
Parity Method for Error Detection:

- Many digital systems employ same method for detection (and sometimes correction) of errors. One of the simplest and most widely used schemes for error detection is the **parity method**.

Parity Bit:-

- A parity bit is an extra bit that is attached to a code group that is being transferred from one location to another. The parity bit is made either 0 or 1, depending on the number of 1s that are contained in the code group. Two different methods are used:
  
  1. **Odd Parity**
  2. **Even Parity**

<table>
<thead>
<tr>
<th>Parity Bit</th>
<th>P</th>
<th>Information Bits</th>
</tr>
</thead>
</table>

**Figure 11.** Parity-coded information.
Parity Method for Error Detection:

Parity Bit:-

- For odd parity, this parity bit is set to a 1 or a 0 at the transmitter such that the sum of the 1 bits in the entire word is odd. The following are examples:

<table>
<thead>
<tr>
<th>Parity</th>
<th>Data</th>
<th>Total 1s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1010110</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1010111</td>
<td>5</td>
</tr>
</tbody>
</table>

- Note that in each case, the total number of 1 bits in the word (including the parity bit) is odd. One parity and seven data bits form the 8-bit word that is transmitted to the receiving location.

- At the receiving end, each 8-bit word that is received is checked to see that the word contains an odd number of 1’s. If a word is received that has an even number of 1’s, then the receiver detects an error and will request a retransmission.
Number Systems and Codes

Parity Method for Error Detection:

Parity Bit:-

- Even parity is also used in many systems. For even parity, this parity bit is set to a 1 or 0 at the transmitter such that the sum of the 1 bits in the entire word is even. The following are examples:

<table>
<thead>
<tr>
<th>Parity</th>
<th>Data</th>
<th>Total 1s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1010111</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>1010000</td>
<td>2</td>
</tr>
</tbody>
</table>

- When the parity method is being used, the transmitter and the receiver must have agreement, in advance, as to whether odd or even parity is being used. There is no advantage of one over the other although even parity seems to be used more often.

- Regardless of whether even parity or odd parity is used, the parity bit becomes an actual part of the code word.
**Number Systems and Codes**

- **Parity Method for Error Detection:**
  - **Parity Bit:-**
    - The parity bit is issued to detect any single-bit errors that occur during the transmission of a code from one location to another. This parity method wouldn’t work if two bits were in error, because two errors would not change the “oddness” or “evenness” of the number of 1’s in the code.
    - In practice, the parity method is used only in situations where the probability of a single error is very low and the probability of double errors is essentially zero.
References