“**Module 2: Assignment 1 - Number System”**

Harsh Siddhapura

“Department of Information Technology, Arizona State University”

“IFT 510: Principles of Computer Information and Technology”

“Dr. Dinesh Sthapit”

“August 26, 2023”

“Complete the following tasks:”

“Task-1: Counting and Arithmetic”

1. **“Explain the concept of the decimal or base 10 number system and its origin.”**

   The decimal or base 10 number system is the most often used number system for counting and arithmetic. Base 10 is so-called because it has ten distinct digits, including zero, that may be used to represent numbers. In the decimal system, the value of each digit is determined by its location relative to the decimal point. Each digit to the left of the decimal point is ten times heavier than its right neighbour. This connection applies to any number basis (Wiley, 5th edition, p. 99).

   The origin of the decimal number system can be traced back to ancient civilizations. The earliest forms of counting and arithmetic used by humans were based on their fingers, which naturally provided a convenient and intuitive base 10 system. The decimal system is frequently used because humans have 10 fingers, which makes counting and calculating easier (Wiley, 5th edition, p. 115).

2. **“Compare and contrast the binary (base 2), octal (base 8), and hexadecimal (base 16) number systems. Include examples of each.”**

   Details and comparison of each number system is given below:

   - **Binary (Base 2) Number System:** Binary numbers have only two digits, 0 and 1. It is the most basic number system used in computers, with each digit representing a bit. A subscript of 2 is commonly used to indicate binary numerals. 10102, for example, is the decimal number 10.
   - **Octal (Base 8) Number System:** The octal number system employs eight digits ranging from 0 to 7. Because each octal digit represents three binary digits, it is widely used as a shorthand notation for binary integers. The subscript 8 is commonly used to signify octal numerals. For example, the decimal number 23 is represented as 278.
   - **Hexadecimal (Base 16) Number System:** The hexadecimal number system employs sixteen digits, ranging from 0 to 9 and A to F. Because each hexadecimal digit represents four binary

digits, it is also often used as a shorthand notation for binary integers. A subscript of 16 is commonly used to indicate hexadecimal values. 2A16, for example, represents the decimal value 42 (Wiley, 5th edition, p. 99).

In summary, the binary number system is made up of two digits (0 and 1), the octal number system is made up of eight digits (0 to 7), and the hexadecimal number system is made up of sixteen digits (0 to 9 and A to F). Each method has its own set of benefits and is widely employed in computer programming and digital systems (Wiley, 5th edition, p. 113).

3. **"Discuss the reasons for using the binary number system in early computer design."**

The reasons for using the binary number system in early computer design are as follows:

- **Compatibility with Electronic Components:** Because it corresponds closely with the behaviour of electronic components, the binary number system was selected for early computer architecture. Electronic circuits can readily encode and manipulate binary digits (bits) using on/off states, making data storage and processing efficient and reliable.
- **Simplicity and Consistency:** With only two potential digits (0 and 1), the binary number system has a basic and consistent structure. This simplicity facilitates the design and implementation of the logic circuits and algorithms required for computer operations.
- **Efficient Data Storage:** When compared to other number systems, binary numbers require less physical area to express. When storage capacity was limited and expensive in the early days of computers, this efficiency was critical. Computers could store and analyse more data within their available resources if they used binary representation.
- **Accurate and Reliable Calculations:** In digital circuitry, the binary number system enables exact and error-free computations. Simple logic gates can execute binary arithmetic operations like addition and subtraction, providing reliable results without the need for sophisticated analogue computations.

- **Compatibility with Digital Communication:** Binary representation is compatible with digital communication systems, allowing computers to communicate with other devices and exchange data and instructions. Computers could readily connect and share information by employing the binary number system, allowing the creation of computer networks and the flow of data between various systems.

Overall, because of its compatibility with electrical components, simplicity, efficient data storage, precise computations, and compatibility with digital communication, the binary number system was chosen for early computer architecture.

“Task-2:Keeping Track of the Bits”

1. **“Explain the concept of bits, bytes, and words in relation to storing and manipulating data.”**

   The bits, bytes and words could be explained as below:

   - **Bits:** Data is stored and processed in computers in the form of binary numbers, which are made up of bits. A bit is the smallest unit of data and can have either a 0 or a 1 value. It is the most fundamental type of information in a computer system.
   - **Bytes:** A byte is an 8-bit group. It is the most basic unit of storage in a computer and can represent a single text character or a tiny integer. Bytes are employed in computer systems to store and process data.
   - **Words:** A word is a type of data unit that is greater than a byte. The size of a word varies according to computer architecture, however it is typically 2, 4, or 8 bytes. Words are used to store and manage bigger datasets like numbers or memory locations. They make it easier to access and process data in a computer system (Wiley, 5th edition, p. 99).

2. **“Discuss how the number of bits used in calculations affects the accuracy of results and the size/range of numbers manipulated by computers.”**

The amount of bits utilised in calculations has a direct impact on the precision of the findings. More exact computations may be conducted with a higher number of bits. This is due to the fact that more bits allow for a wider range of values to be represented, resulting in more accurate calculations. However, with fewer bits, the range of values that may be represented is limited, resulting in less accurate outputs (Wiley, 5th edition, p. 104).

The amount of bits employed in calculations defines the size or range of values that computers can manage. Each bit represents a binary numeral that can be either 0 or 1. The greater the number of bits employed, the greater the range of numbers that may be represented. With 16 bits, for example, the range of integers that may be represented is 0 to 65,536. When additional bits, such as 32 bits, are utilised, the range expands to 0 to 4,294,967,295. As a result, the number of bits has a direct influence on the size and range of numbers that computers can handle (Wiley, 5th edition, p. 99).

“Task-3: Number Systems”

1. **“ Compare and contrast the Roman numeral system with modern positional notation systems (decimal, binary, octal, hexadecimal). Explain the concept of positional notation determining value.”**

- Roman Numeral System: The Roman numeral system is a non-positional notation method in which numbers are represented by letters. It originated in ancient Rome and was widely used in Europe until the decimal system was adopted. Each letter in the Roman numeral system indicates a distinct value, and numerals are constructed by combining these letters. However, the Roman numeral system lacks positional value, which means that the location of a letter has no effect on its value. When compared to positional notation systems, this complicates arithmetic operations and computations.
- Modern Positional Notation Systems: The idea of positional value underpins modern positional notation systems such as decimal, binary, octal, and hexadecimal. The value of a digit in these systems is determined by its location or place value inside the number. The rightmost digit denotes the unit's place,

followed by the tens place, and so on. Each digit is multiplied by the base raised to the power of its location, and the sum of the results determines the total value of the number.

- Positional Notation Determining Value: The value of a number in positional notation systems is defined by the combination of digits and their places. The number of unique digits utilised and the value of each digit are defined by the system's base. In the decimal system, for example, the base is 10 and the digits run from 0 to 9. A digit's value in a certain position is calculated by multiplying the digit by the base raised to the power of its position.

  For instance, in the decimal system, the number 123 can be calculated as (1 * 10^2) + (2 * 10^1) + (3 * 10^0), which equals 100 + 20 + 3, resulting in a value of 123. This concept of positional notation allows for efficient representation, manipulation, and calculation of numbers in various systems, making it easier to perform arithmetic operations and conversions (Wiley, 5th edition, p. 111).

2. **"Provide examples of converting numbers from base 16 (hexadecimal) to base 10 (decimal) using positional notation."**

   To convert numbers from base 16 (hexadecimal) to base 10 (decimal) using positional notation, we can follow a simple process. It is:

➔ Identify the hexadecimal digits in the number.

➔ Assign the corresponding decimal values to each digit.

➔ Multiply each digit by the appropriate power of 16 based on its position.

➔ Sum up the results to obtain the decimal equivalent.

Let's take an example to illustrate this process:

- Example 1: Converting 3D7 from base 16 to base 10
  - The hexadecimal number is 3D7.
  - The decimal value of 3 is 3, D is 13, and 7 is 7.
  - Multiply the first digit (3) by 16^2, the second digit (D) by 16^1, and the third digit (7) by 16^0.

    - Calculate: (3 * 16^2) + (13 * 16^1) + (7 * 16^0) = 3 * 256 + 13 * 16 + 7 = 768 + 208 + 7 = 983.

Therefore, the decimal equivalent of 3D7 in base 16 is 983.

- Example 2: Converting A3AB from base 16 to base 10
    - The hexadecimal number is A3AB.
    - The decimal value of A is 10, 3 is 3, A is 10, and B is 11.
    - Multiply the first digit (A) by 16^3, the second digit (3) by 16^2, the third digit (A) by 16^1, and the fourth digit (B) by 16^0.
    - Calculate: (10 * 16^3) + (3 * 16^2) + (10 * 16^1) + (11 * 16^0) = 10 * 4096 + 3 * 256 + 10 * 16 + 11 = 40960 + 768 + 160 + 11 = 42099.

Therefore, the decimal equivalent of A3AB in base 16 is 42099.

“Task-4: Counting and Arithmetic in Different Bases”

1. **“Demonstrate how to perform addition and multiplication in the decimal (base 10) and binary (base 2) number systems. Use tables or step-by-step explanations for clarity.”**

- Addition
    - Decimal Addition: Add 247 and 158
        - Step 1: Add the rightmost digits (ones place): 7 + 8 = 15. Write down 5 and carry over 1.
        - Step 2: Add the next digits (tens place) along with the carried-over 1: 1 + 4 + 5 = 10. Write down 0 and carry over 1.
        - Step 3: Add the hundreds digits along with the carried-over 1: 1 + 7 + 1 = 9.

      Result: 247 + 158 = 405

    - Binary Addition:: Add 1011 and 1101
        - Step 1: Add the rightmost bits: 1 + 1 = 2 (in binary, 10). Write down 0 and carry over 1.
        - Step 2: Add the next bits along with the carried-over 1: 1 + 0 + 1 = 2 (in binary, 10). Write down 0 and carry over 1.

        - Step 3: Add the next bits along with the carried-over 1: 1 + 1 + 1 = 3 (in binary, 11). Write down 1 and carry over 1.
        - Step 4: Add the leftmost bits along with the carried-over 1: 1 + 1 + 1 = 3 (in binary, 11). Write down 1.

    Result: 1011 + 1101 = 11000.

- Multiplication
    - Decimal Multiplication: Multiply 24 and 13
        - Step 1: Multiply the rightmost digits: 4 * 3 = 12.
        - Step 2: Multiply the next digits and shift one position left: 2 * 3 = 6.
        - Step 3: Add the two results: 120 + 60 = 180.

    Result: 24 + 13 = 180

    - Binary Multiplication:: Multiply 1101 and 101.
        - Step 1: Multiply the rightmost bit of the second number by each bit of the first number and shift left: 1 * 1101 = 1101.
        - Step 2: Multiply the second rightmost bit and shift two positions left: 0.
        - Step 3: Multiply the third rightmost bit and shift three positions left: 1 * 11010 = 11010.
        - Step 4: Multiply the leftmost bit and shift four positions left: 1 * 110100 = 110100.
        - Step 5: Add the results: 0 + 0 + 1101 + 0 + 11010 + 110100 = 100011.

    Result: 1101 * 101 = 100011

“Task-5: Converting Numbers between Bases”

1. **“Provide step-by-step instructions for converting numbers from base 10 (decimal) to base 2 (binary), base 8 (octal), and base 16 (hexadecimal).”**

- Converting Numbers from Base 10 to Base 2 (Binary):

To convert a number from base 10 to base 2 (binary), follow these steps:

- ➔ Start with the given decimal number.
- ➔ Divide the number by 2 and write down the remainder.
- ➔ Continue dividing the quotient by 2 until the quotient becomes 0.
- ➔ Write down the remainders in reverse order to get the binary representation of the number.

For example, to convert the decimal number 10 to binary:

- ➔ 10 ÷ 2 = 5 with a remainder of 0
- ➔ 5 ÷ 2 = 2 with a remainder of 1
- ➔ 2 ÷ 2 = 1 with a remainder of 0
- ➔ 1 ÷ 2 = 0 with a remainder of 1

So, the binary representation of 10 is 1010 (Wiley, 5th edition, p. 113).

- Converting Numbers from Base 10 to Base 8 (Octal):

To convert a number from base 10 to base 8 (octal), follow these steps:

- ➔ Start with the given decimal number.
- ➔ Divide the decimal number by 8, and note down the quotient and remainder.
- ➔ The remainder will be the least significant digit (rightmost digit) of the octal number.
- ➔ Divide the quotient obtained in step 2 by 8 again, and note down the new quotient and remainder.
- ➔ Repeat step 4 until the quotient becomes zero.
- ➔ The remainders obtained in each step, from bottom to top, will form the octal number.
- ➔ Write down the octal number obtained.

For example, to convert the decimal number 6026 to octal:

- ➔ Step 1: Start with the decimal number 6026.

- ➔ Step 2: Divide 6026 by 8. The quotient is 753 and the remainder is 2.
- ➔ Step 3: The remainder 2 is the least significant digit of the octal number.
- ➔ Step 4: Divide 753 by 8. The quotient is 94 and the remainder is 1.
- ➔ Step 5: Divide 94 by 8. The quotient is 11 and the remainder is 6.
- ➔ Step 6: Divide 11 by 8. The quotient is 1 and the remainder is 3.
- ➔ Step 7: The remainders obtained in each step, from bottom to top, are 3, 6, 1, and 2.
- ➔ Step 8: Write down the octal number obtained: 3612.

- Converting Numbers from Base 10 to Base 16 (Hexadecimal):

To convert a number from base 10 to Base 16 (Hexadecimal), follow these steps:

- ➔ Start with the given decimal number.
- ➔ Divide the decimal number by 16.
- ➔ Write down the remainder as the rightmost digit of the hexadecimal number.
- ➔ Divide the quotient obtained in step 2 by 16 again.
- ➔ Write down the remainder as the next digit to the left of the previous digit.
- ➔ Repeat steps 4 and 5 until the quotient becomes zero.
- ➔ The resulting digits, from right to left, form the hexadecimal representation of the original decimal number.

For example, let's convert the decimal number 12345 to hexadecimal:

- ➔ Divide 12345 by 16: 12345 ÷ 16 = 771 remainder 9. Write down 9 as the rightmost digit.
- ➔ Divide 771 by 16: 771 ÷ 16 = 48 remainder 3. Write down 3 as the next digit to the left.
- ➔ Divide 48 by 16: 48 ÷ 16 = 3 remainder 0. Write down 0 as the next digit.
- ➔ Divide 3 by 16: 3 ÷ 16 = 0 remainder 3. Write down 3 as the leftmost digit.
- ➔ The resulting hexadecimal representation is 3039.

Therefore, the decimal number 12345 is equivalent to 3039 in base 16 (hexadecimal).

2. **“Explain the quicker algorithms for converting numbers to binary, octal, and hexadecimal.”**

   Quicker Algorithms for Converting Numbers to Binary, Octal, and Hexadecimal are explained below:

   - **Binary Conversion:** The division technique may be used to convert integers to binary. Begin by dividing the decimal value by two and writing down the residue. Continue dividing the quotient by 2 until it reaches zero. By arranging the remainders in reverse order, the binary representation is achieved. This algorithm is simple and effective.
   - **Octal Conversion:** The division technique may also be used to convert numbers to octal. Take the remainder of the decimal number and divide it by 8. Continue dividing the quotient by 8 until it reaches zero. By putting the remainders in reverse order, the octal representation is achieved. The binary conversion algorithm is identical to this one.
   - **Hexadecimal Conversion:** The division technique may also be used to convert numbers to hexadecimal. Take the remainder of the decimal number and divide it by 16. Continue dividing the quotient by 16 until it reaches zero. By substituting remainders higher than 9 with matching letters (A-F) and arranging them in reverse order, the hexadecimal representation is achieved. The binary and octal conversion techniques are likewise comparable to this one.

   These methods convert integers to binary, octal, and hexadecimal representations in a rapid and efficient manner. They are based on the division technique and are simple to implement in a variety of computer languages (Wiley, 5th edition, p. 113).

“Task-6: Fractions and Mixed Numbers”

1. **“Discuss the representation of fractions in different number bases and the effect of moving the number point.”**

   Fractions are expressed in multiple number bases by grouping the digits in the smaller base and converting each group independently. To convert a fraction from base 2 to base 8, for example, the digits are grouped by threes (because 23 = 8) and each group is transformed as usual. It's worth noting that fractional zeros are added to the right of the fraction.

A number's value is multiplied by the base when the number point is moved one position to the right. Moving the number point one position to the left, on the other hand, halves its value by the base. This is true for both integer and fractional numbers. When converting between bases, the option of multiplication or division operation for the fractional component is reversed. It is also worth noting that moving a number right one digit results in the loss of any fractional values.

These ideas are applicable to several number bases and may be used to translate fractions between bases and perform arithmetic operations on mixed numbers (Wiley, 5th edition, p. 120).

2. **"Explain the challenges in converting fractions between base 10 and base 2 and the limitations of exact conversion."**

There is a challenge in converting fractions between base 10 and base 2 because of the changes in the multipliers associated with each digit, converting fractions between base 10 and base 2 can be difficult. The multipliers to the left of the radix point are integers and have a direct link with the bases. However, the multipliers to the right of the radix point are fractional and may not have a reasonable connection between the bases. This makes finding an accurate representation of a fraction in base 2 difficult when it is exactly represented in base 10.

Limitations of exact conversions are that, for fractions, exact conversion between base 10 and base 2 is not always achievable. Many base 10 fractions result in an infinite number of base 2 fractions, implying that no combination of bits adds up exactly to the fraction. The decimal fraction 0.1 in base 10 cannot, for example, be expressed accurately in binary form. It contains a binary repeating fraction of 0.0001100110011... Because there is no universal link between fractions of types 1/10k and 1/2k, it is impossible to assume that a number that can be represented in base 10 can also be represented in base 2.

3. **“Describe the process of converting mixed numbers from one base to another, including the handling of the integer and fractional parts.”**

   The integer and fractional components of mixed numbers must be treated independently when converting them from one base to another.

   The standard arithmetic rules apply to the integer portion. The radix points must line up while adding, subtracting, multiplying, or dividing mixed numbers. The radix point is determined in the same way as the base 10 point is. In base 8, for example, the number of digits to the right of the radix point in the result equals the total of the radix digits in the multiplier and the multiplicand.

   The technique for the fractional component is identical to converting fractions from base 10 to another base. The radix point is moved to the left, and the number is repeatedly multiplied by the base value. Values that shift to the left of the radix point are saved and then discarded. This procedure is continued until the appropriate number of digits of precision is reached or the amount being multiplied equals zero.

   Converting mixed numbers from one base to another entails treating the integer and fractional components separately and using the proper conversion procedures to each (Wiley, 5th edition, p. 120).

“Report”

“Number Systems and Conversions”

This report delves into the intricate world of number systems, shedding light on the significance of the decimal (base 10) system, the distinctions between binary (base 2), octal (base 8), and hexadecimal (base 16) systems, and the complexities of number conversions. The report also delves into the realm of arithmetic in various bases, the role of bits, bytes, and words in data representation, and the challenges and limitations of converting fractions and mixed numbers. Number systems are crucial to mathematics, computer science, and a variety of engineering disciplines. They provide the foundation for representing and manipulating data, allowing for more efficient computations and operations. This paper goes into number systems, conversions, and the relevance of positional notation. It covers decimal, binary, octal, and hexadecimal systems, as well as fraction and mixed number representation in several bases.

The decimal number system, based on ten distinct digits (0-9), forms the bedrock of modern counting and arithmetic practices. Its roots trace back to the earliest human civilizations that counted using their fingers. In the decimal system, the position of each digit relative to the decimal point determines its value. This positioning mechanism, coupled with the familiarity of human beings with ten fingers, renders the decimal system highly intuitive and practical (Wiley, 5th edition, p. 99). Because of its ten different digits (0 to 9), the decimal or base 10 number system is commonly used for counting and mathematics. Its origins may be traced back to ancient cultures, when fingers were used as a natural basis for counting. “The decimal system is designed such that the value of each digit is determined by its location relative to the decimal point. Because humans have 10 fingers, this approach simplifies mathematics and arithmetic” (Wiley, 5th edition, p. 99).

Binary, octal, and hexadecimal systems, each with unique attributes, play pivotal roles in computer science and digital technology. Binary, utilising only 0 and 1 digits, forms the foundation of digital data representation. “Octal, with digits ranging from 0 to 7, often serves as a concise representation for binary values. Hexadecimal, boasting digits 0-9 and A-F, is instrumental in addressing and transmitting binary patterns” (Wiley, 5th edition, p. 99). Binary arithmetic is a cornerstone of computer operations, facilitated by electronic components' compatibility with binary digits. Adding and multiplying binary numbers necessitates systematic carrying over and logic gate operations. “Binary arithmetic's simplicity and accuracy align harmoniously with the underlying electronic nature of computers, facilitating data manipulation and processing” (Wiley, 5th edition, p. 99).

Binary (base 2), octal (base 8), and hexadecimal (base 16) are positional notation systems that use various sets of symbols to represent values. Because of its compatibility with electrical components, binary is vital for computers. “Octal and hexadecimal are efficient binary integer shorthand notations, with each digit encoding a certain number of binary digits. The adaptability of binary, octal, and hexadecimal representations is essential in computer programming” (Wiley, 5th edition, p. 113). “Bits are the most basic data units, with values of 0 or 1. Bytes, which are made up of 8 bits, are the most basic storage units. Words, which are bigger than bytes, differ by computer architecture and may hold larger datasets. Bit size has an impact on computation accuracy and the range of values that can be represented. More bits provide for more precision and a wider range of values” (Wiley, 5th edition, p. 99, p. 104).

Converting numbers between bases necessitates the use of efficient methods. “The division approach is used for binary, octal, and hexadecimal values. This is accomplished by dividing the decimal number by the base, noting the remainders, and generating the resultant representation. These procedures are crucial for effective conversions and are based on the amount of digits necessary in each base” (Wiley, 5th edition, p. 113). “Converting fractions between bases and shifting the number point are difficult tasks. It is critical to group digits and transform them individually. Moving the number point to the right doubles the value and moving it to the left divides it. Because fractional portions do not always have a clear association across bases, exact conversion of fractions is limited" (Wiley, 5th edition, p. 120).

Converting numbers between different bases involves algorithms that rely on division as the foundational operation. For instance, converting base 10 to binary entails repeated division by 2, with remainders forming the binary representation in reverse order. "This systematic process ensures efficient and precise conversion across various programming languages and platforms" (Wiley, 5th edition, p. 113). "Representing fractions in diverse bases demands a distinct approach, involving grouping of digits and judicious placement of the radix point. Moving the number point to the right multiplies the value by the base, while shifting left divides it. However, the complexity arises when dealing with fractional components due to differing multipliers and potential loss of precision" (Wiley, 5th edition, p. 120).

The conversion of fractions between base 10 and base 2 introduces challenges owing to the incongruity of multipliers. "While some fractions possess exact counterparts in both bases, others, such as 0.1, do not possess precise binary representations due to recurring fractions. This limitation highlights the intricate relationship between base systems and their implications for accurate representation" (Wiley, 5th edition, p. 120). "The process of converting mixed numbers necessitates the separation of integer and fractional components. Standard arithmetic rules apply to integers, while fractional conversion relies on multiplication by the base and strategic placement of the radix point. The alignment of radix points is pivotal to ensure accuracy in conversions and calculations" (Wiley, 5th edition, p. 120). When converting mixed numbers, the integer and fractional components must be handled independently. Standard arithmetic methods apply to integers, and the radix points align throughout operations. The fractional component operates on the same principles as fraction conversion, relocating the number point and multiplying or dividing by the base value.

In essence, understanding number systems and their conversions unveils the intricate mechanics that underpin digital technology, computer operations, and data representation. The various systems' unique attributes and challenges offer insights into their applications and limitations. The report's exploration of number systems and conversions is vital for any technologically inclined individual, fostering a deeper understanding of the foundation upon which modern digital systems are built.

Understanding number systems, conversions, and their ramifications is essential for a wide range of applications, particularly in computer science and engineering. The ease and consistency of the decimal system with human counting, as well as the efficiency of binary, octal, and hexadecimal systems, all contribute to their relevance. Manipulation of integers, fractions, and mixed numbers between bases necessitates exact algorithms as well as a complete understanding of positional notation rules. This understanding is required for accurate data representation, storage, and manipulation.

References

The Architecture of Computer Hardware, Systems Software, and Networking: An Information Technology Approach, 6th Edition

Publisher: Wiley, Authors: Irv Englander, Wilson Wong