**Chapter 2**

**Number Systems and Codes**

*Lesson 2.1:* Number Systems

*Lesson 2.2:* Binary Arithmetic

*Lesson 2.3:* Codes

****

***Lesson 2.1***

***Number Systems***

**2.1.0 Objectives**

*On completion of this lesson you will know:*

* *Basic concepts of different number systems*
* *Characteristics of decimal, binary, octal and hexadecimal numbers*
* *Conversion of numbers*

**2.1.1 Number Systems**

Arithmetic operations using decimal numbers are quite common to us. Digital systems like calculators and computers do not understand the words and numbers we use. In logical design, however, it is necessary to perform manipulations in the binary system of numbers because of the on-off nature of the physical devices used. The popular number systems are: decimal, binary, octal and hexadecimal. Basic characteristics of these number systems are given below

**Decimal Number System**

It is the most commonly used number system in real life. It has the following features:

* Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (total ten digits)
* Base (radix): 10 (ten)
* Weights: 1, 10, 100, 1000, … (powers of base 10, this is, )

A positive integer can be expressed as a sum of power of 10. For example, the decimal number 1962 can be expressed as:



Here base 10 indicates that the number is a decimal number. In general, a number with a decimal point is represented by a series of coefficients:



And a number expressed in a base  system, where  is the base or radix of the number, has coefficients multiplied by power of 



For example, the decimal number 1962.22 can be expressed as:



**Binary System**

Digital computers use binary numbers for internal operations. It has the following features:

* Digits: 0,1 (total two digits), also called binary digits.
* Base (radix): 2
* Weights: 1, 2, 4, 8, 16, … (powers of base 2, this is, )

For example, the binary 101011 can be expressed as:



**Octal System**

Octal numbers are used in programming languages. It has the following features:

* Digits: 0, 1, 2, 3, 4, 5, 6, 7 (total eight digits)
* Base (radix): 8
* Weights:1, 8, 64, 512, … (powers of base 8, this is, )

**Hexadecimal System**

Hexadecimal (also called hex in short) is a propositional numeral system. Hexadecimal is commonly used to represent computer memory addresses. It has the following features:

* Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (total sixteen digits)
* Base (radix): 16
* Weights:1, 16, 256, … (powers of base 16, this is, )

**2.1.2 Converting from Decimal to base  system**

To convert a decimal number to its other equivalent numbers, the remainder method can be used. (This method can be used to convert a decimal number into any other base.) The remainder method involves the following four steps:

1. Divide the decimal number by the base (in the case of binary, divide by 2).
2. Indicate the remainder to the right.
3. Continue dividing into each quotient (and indicating the remainder) until the divide operation produces a zero quotient.
4. The base 2 number is the numeric remainder reading from the last division to the first.

**Example 2.1.1:** Convert 4710 to its binary equivalent.



**Example 2.1.2:** Convert 4710 to its octal equivalent.



**Example 2.1.3:** Convert 4710 to its hexadecimal equivalent.



**Binary to Octal Conversion: A shortcut method**

The rule is as follows:

1. Starting from the right of the given binary stream into group of three, if leftmost group has fewer bits, attach the required number of leading 0s to complete the group.
2. Determine equivalent octal digit for each group.

**Example 2.1.4:** Find the octal number of 1001101112

Step 1: 100 110 111

Step 2: 1002= (1×22) + (0×21) + (0×20)=48; 1102= (1×22) + (1×21) + (0×20)=68; 1012= (1×22) + (1×21) + (1×20)=78

Thus the octal number of 1001101112 is 4678

**Octal to Binary Conversion: A shortcut method**

The rule is as follows:

1. Find the equivalent binary group of 3 digits for each octal digit.
2. Find the results by combining binary groups.

**Example 2.1.5:** Find the binary number of 4678

Step 1: 48=1002; 68=1102; 78=1112

Step 2: 100 110 111 is the binary equivalent of 4678

Thus the binary number of 4678 is 1001101112

**Binary to Hexadecimal Conversion: A shortcut method**

The rule is as follows:

1. Starting from the right of the given binary stream into group of four, if leftmost group has fewer bits, attach the required number of leading 0s to complete the group.
2. Determine equivalent one hexadecimal digit for each group.

**Example 2.1.6:** Find the hexadecimal equivalent number of 100110112

Step 1: 1001 1011

Step 2: 10012= (1×23) + (0×22) + (0×21) + (1×20) = 916; 10112= (1×23) +(0×22) + (1×21) + (1×20) = B16

Thus the octal number of 100110112 is 9B16

**Hexadecimal to Binary Conversion: A shortcut method**

The rule is as follows:

1. Find the equivalent binary group of 4 digits for each hexadecimal digit.
2. Find the results by combining binary groups.

**Example 2.1.7:** Find the binary equivalent of 9B 16

Step 1: 916=10012; B16=10112

Step 2: 10011011 is the binary equivalent of 9B16

Thus the binary number of 9B16 is 100110112

**2.1.3 Key points**

* The popular number systems are: decimal, binary, octal and hexadecimal.
* Decimal number has digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (total ten digits), base (radix): 10, weights: 1, 10, 100, 1000, … (powers of base 10)
* Binary number has digits: 0,1 (total two digits), base (radix): 2, weights: 1, 2, 4, 8, 16, … (powers of base 2)
* Octal has digits: 0, 1, 2, 3, 4, 5, 6, 7 (total eight digits), base (radix): 8, weights:1, 8, 64, 512, … (powers of base 8)
* Hexadecimal has digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (total sixteen digits), base (radix): 16, weights:1, 16, 256, … (powers of base 16)

**2.1.4 Practice Set**

**Multiple Choice Questions**

1. The decimal equivalent of 11102 is \_\_\_\_\_\_\_\_\_\_\_.
   1. 810
   2. 1210
   3. 1010
   4. 1410
2. The binary equivalent of 10910 is \_\_\_\_\_\_\_\_\_\_.
   1. 1001102
   2. 11011012
   3. 11110012
   4. 099192
3. The decimal equivalent of 73D516 is \_\_\_\_\_\_\_\_\_\_
   1. 2965310
   2. 2955310
   3. 1965310
   4. 1955310

**Review Questions**

1. Name four different number systems and mention their base.
2. What are the bases of four different number systems?
3. Give the binary value of A16 1210 316, and 68

**Analytical Questions**

1. Convert the binary number to decimal (a) 101012 (b) 101.012
2. Convert the decimal number to binary (a) 654.4310 (b) 33310
3. Convert the hexadecimal number to binary (a) 73A16 (b) 12916
4. Convert the binary number to hexadecimal (a) 1111110012 (b) 1010.1102

****

***Lesson 2.2***

***Binary Arithmetic***

**2.2.0 Objectives**

*On completion of this lesson you will know:*

* *Basic concepts of binary arithmetic*
* *Steps for binary addition, subtraction, multiplication and division.*
* *Additive method of binary subtraction.*

**2.2.1 Binary Number**

We use the decimal numbers in our daily practice. The digits 0-9 are combined to get any numbers like 104, 4561 etc. We also perform decimal arithmetic, which involves addition, subtraction, multiplication and division of numbers. For example, a chocolate costs Tk 2 and we want to buy two chocolates; the total cost of the two chocolates will be Tk 4 (i.e., or ). The binary number system is used in the computer systems. The digits 0 and 1 are combined to get any binary number like 1001, 11000110 etc. In a binary number, a digit 0 or 1 is called a bit. For example, 10 is a 2-bit binary number, and, 10000111 is an 8-bit binary number. All kinds of data, be it alphabets, numbers, symbols, sound or video, are represented as combination of bits i.e. 0’s and 1’s. Each character is a unique combination of bits. In the upcoming sections, basic arithmetic operations in the binary number system will be discussed.

**2.2.2 Binary Arithmetic**

The arithmetic operations - addition, subtraction, multiplication and division, performed on the binary numbers is called binary arithmetic.

**Binary Addition**

Binary addition involves adding of two or more binary numbers. The binary addition rules, which are shown in Table 2.2.1, are used while performing the binary addition.

Table 2.2.1 Binary addition rules

|  |  |  |  |
| --- | --- | --- | --- |
| **Input 1** | **Input 2** | **Sum** | **Carry** |
| 0 | 0 | 0 | 0 (No carry) |
| 0 | 1 | 1 | 0 (No carry) |
| 1 | 0 | 1 | 0 (No carry) |
| 1 | 1 | 0 | 1 (carry) |

The rule for binary addition of three inputs, when all the inputs are 1 is shown in Table 2.2.2.

Table 2.2.2 Binary addition of three inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input 1** | **Input 2** | **Input 3** | **Sum** | **Carry** |
| 1 | 1 | 1 | 1 | 1 |
| Steps | | | | |
|  | | | | |

Addition of the binary numbers involves the following steps-

1. Start addition by adding the bits in unit column (the rightmost column). Use the rules of binary addition.
2. The result of adding bits of a column is a sum with or without a carry.
3. Write the sum in the result of that column. If carry is present, the carry is carried-over to the addition of the next left column.
4. Repeat steps 2-4 for each column.

**Example 2.2.1.** *Add 10 and 01. Verify the answer with the help of decimal addition*.



**Example 2.2.2.** *Add 01 and 11. Verify the answer with the help of decimal addition.*



**Example 2.2.3.** *Add 11 and 11. Verify the answer with the help of decimal addition.*



**Example 2.2.4.** *Add 10111, 11100 and 11. Verify the answer with the help of decimal addition.*



**Binary Subtraction**

Binary subtraction involves subtraction of two binary numbers. The binary subtraction rules are used while performing the binary subtraction. The rule for binary subtraction is shown in Table 2.2.3.

Table 2.2.3 Binary Subtraction Rules

|  |  |  |  |
| --- | --- | --- | --- |
| **Input 1** | **Input 2** | **Difference**  **(Input 1-Input 2)** | **Borrow** |
| 0 | 0 | 0 | 0 (No Borrow) |
| 0 | 1 | 1 | 1 (Borrow) |
| 1 | 0 | 1 | 0 (No Borrow) |
| 1 | 1 | 0 | 0 (No Borrow) |

The steps for performing subtraction of the binary numbers are as follows-

1. Start subtraction by subtracting the bit in the lower row from the upper row, in the unit column.
2. Use the binary subtraction rules. If the bit in the upper row is less than lower row, *borrow* 1 from the upper row of the next column (on the left side). The result of subtraction of two bits is the *difference.*
3. Write the *difference* in the result of that column.
4. Repeat step 2-3 for each column and so on.

**Example 2.2.5.** Subtract 01 from 11. Verify the answer with the help of decimal subtraction.



**Example 2.2.6.** Subtract 10011111 from 10101001. Verify the answer with the help of decimal subtraction.



**Additive Method of Binary Subtraction:** This method is called complement method. The following steps are involved:

1. Find the complement of subtrahend.
2. Add results of step 1 to the minuend.
3. If a carry is obtained, add it to obtain the result, else recomplement the sum and attach a negative sign to obtain the result.

**Example2.2.7: Subtract 10111 from 11001 using additive approach**

Solution

Step 1: Here the complement of subtrahend 10111 is 01000

Step 2: Here the minuend is 11001.



Step 3: Add carry to obtain the result, i.e., 1+1=102

The whole process is shown as

The complement of 10111 is 01000



**Example 2.2.8: Subtract 1001 from 0101 using additive approach**

The complement of 1001 is 0110



**Binary Multiplication**

It is actually much simpler than decimal multiplication. In the case of decimal multiplication, we need to remember 3×9 = 27, 7×8 = 56, and so on. In binary multiplication, we only need to remember the following Table 2.2.4,

Table 2.2.4 Binary multiplication rule

|  |  |  |
| --- | --- | --- |
| **Input 1** | **Input 2** | **Multiplication** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Find the binary multiplication of 101 and 11,



First we multiply 101 by 1, which produces 101. Then we put a 0 as a placeholder as we would in decimal multiplication, and multiply 101 by 1, which produces 101.



The next step is to add. The result(s) from our previous step indicates that we must add 101 and 1010, the sum of which is 1111.



**Binary division**

Rules for performing binary division is as follows



The method for binary division is as follows

* Starting from left compare divisor with dividend.
* If dividend is greater, take value of the quotient 1 and subtract the divisor from the corresponding digits of dividend.
* If dividend is less, take value of the quotient 0 and repeat whole process till sufficient digits in dividend.

**Example 2.2.9:** Find 



**2.1.5 Key points**

* The binary number system is used in the computer systems.
* In a binary number, a digit 0 or 1 is called a bit.
* All kinds of data, be it alphabets, numbers, symbols, sound or video, are represented as combination of bits i.e. 0’s and 1’s.
* Each character is a unique combination of bits.
* The arithmetic operations - addition, subtraction, multiplication and division, performed on the binary numbers is called binary arithmetic.
* Additive Method of Binary Subtraction is called complement method.

**2.1.6 Practice Set**

**Multiple Choice Questions**

1. Addition in the binary system can be performed using \_\_\_\_\_\_\_\_\_\_
   1. Two steps algorithm
   2. 4 steps algorithm
   3. 3 steps algorithm
   4. None
2. Difference of 110002-100112 is \_\_\_\_\_\_\_\_\_\_
   1. 001112
   2. 110002
   3. 001012
   4. 110102
3. The reminder of 1010001÷11 is \_\_\_\_\_\_\_\_\_\_
   1. 10
   2. 00
   3. 11
   4. 1

**Review Questions**

1. Write down the shortcut method of converting a hexadecimal number to binary number?
2. Define carry and borrow.

**Analytical Questions**

1. Evaluate the following
   1. 100.012+1112
   2. 100102-001102
   3. 2’s complement of 111101102
   4. 100102×1112
   5. 7218×238
   6. A316×1012
   7. 101101÷110

****

***Lesson 2.3***

***Codes***

**2.3.0 Objectives**

*On completion of this lesson you will know:*

* *Basic concepts of data information and codes.*
* *Representation of numeric data using binary numbers,*
* *BCD, EBCDIC, ASCII Codes*
* *Unicode.*

**2.3.1 Data, Information and Codes**

Data are the names given to basic facts such as names and numbers. Unit-price, quantity sold, times, dates, products, name, address etc are the examples of data.

Information is processed data, that is, information is data which have been converted to a more useful form. For example total price = unit price × quantity sold. Here total price is information and unit price and quantity sold are data.

Codes are used to reduce the volume of data. By coding, recording of data can be made less prone to error and the data become more manageable and easier to manipulate.

**2.3.2 Numeric Data Representation**

Numeric data are represented in the computer using binary numbers. So anything which has to be stored in the memory must be converted to a binary form and then the bits can be used.

To store any positive integers the location is filled with bits from right to left. The extra bits are filled up with 0s. To store 42310 in a memory location of word length of 16 bits, first convert 42310 into 1101001112 and fill the extra bits with 0s.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |

**Sign Magniture Representation**

The left most bit in the word represents the sign bit. If the left most bit is 0, the number stored in the word will be treated as a positive integer. On other hand if the left most bit is 1, then the number in the word will be treated as a negative integer. That bit is called the sign bit.

For example, storing +50 in a 16-bit word is as follows

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |

And storing -50 in a 16-bit word is as follows

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |

In the sign magnitue representation, the largest (i.e., positive) integer that can be stored in a 16-bit memory location is . Similarly the smallest (i.e., negative) integer that can be stored in a 16-bit memory location is .

Therefore the range of values, which can be represented in an n-bit word, using the sign magnitude is  through 0 to 

**1’s Complement Representation:** In the 1’s complement representation, positive number is identical to that used in the sign magnitude system. It consists of a left most sign bit in the leftmost position, followed by the magnitude bits. For a positive number two representations are identical. In case of negative numbers, the magnitude bits are represented by their 1’s complement which is obtained by inverting each bit including the sign bit. For example, the 1’s complement of 2510=110012 is 001102. Since thus -610 in 1’s complement is .The general formula for 1’s complement arithmetic is  whereis the in 1’s complement notation,  is the number of bits per word, and . For example with a 8 bit word, and , we have 

**2’s Complement Representation:** 2’s complement representation of a – ve number is obtained by adding a 1 to the 1’s complement of that number. Thus the 2’s complement of –6 is . The general formula for 2’s complement arithmetic is  whereis the in 2’s complement notation,  is the number of bits per word, and . For example with a 8 bit word, and , we have 

**Representation of Real Number**

A real number (signed or unsigned) is divided into two parts: an integer part and a fractional part. In the computer fractional point is not stored, a position is assumed for it. This is called the assumed fractional point. There are two most commonly used representations:

(a) Fixed point and

(b) Floating point

**Fixed point representation:** The position of fractional point is fixed which is assumed to be after the first 8 bits in 16 bits representation. Thus the first 8 bits store the integer portion of the number and the last 8bits store the fractional part. The bits in the integer part are filled from right to left and the remaining bits are padded with 0’s. The left most bit is reserved for the sign bit. The representation of -101001011.0010012 in the 16 bit word is as follows (Figure 2.3.1):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 2.3.1: Fixed point representation**

**Floating point representation:** The floating point representation can be used to represent numbers in any positional number system. It is written in the format , where *m* is the mantissa, which is a signed fixed point number, *e* is the exponent, which is a signed integer and determines the actual position of the decimal point, *b* is the base of the number.

In this representation mantissa and exponent have their own signs as shown in Figure 2.3.2.



**Figure 2.3.2: Floating point representation.**

For example, the number

 can be written as



**2.3.3 BCD Code**

The binary coded decimal (BCD) code is based on the idea of converting each digit of a decimal number into its binary equivalent rather than converting the entire decimal value to a pure binary form. Each decimal digit is represented by 4 bits. When only 4 bits are used a total of 16 configurations are possible.

In BCD format, 6 bits are used to represent each character. The first 2 bits are used as zone bits and the last four bits indicate the digit. When only 6 bits are used a total of 64 configurations are possible which means 64 different characters can be represented. These are sufficient to code 10 decimal digits, 26 alphabets and 28 special characters as shown in Table 2.3.1.

**Table2.3.1: Alphabetic and numeric characters in BCD**

**along with their octal equivalent.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Character | BCD Code | | Octal | Character | BCD Code | | Octal |
| Zone | Digit | Zone | Digit |
| A | 11 | 0001 | 61 | S | 01 | 0001 | 22 |
| B | 11 | 0010 | 62 | T | 01 | 0010 | 23 |
|  |  |  |  |  |  |  |  |
| I | 11 | 1001 | 71 | Y | 01 | 1001 | 31 |
| J | 10 | 0001 | 41 | 1 | 00 | 0001 | 1 |
| K | 10 | 0010 | 42 | 2 | 00 | 0010 | 2 |
|  |  |  |  |  |  |  |  |
| R | 10 | 1001 | 51 | 0 | 00 | 1010 | 12 |

**2.3.4 EBCDIC**

EBCDIC stands for extended binary coded decimal interchange code. In EBCDIC format, 8 bits are used to represent each character. In this code, it is possible to represent 256 () different characters. It can be easily divided into two 4 bit groups. Each of these 4 bit groups can be represented by 1 hexadecimal digit. Alphabetic and Numeric characters in EBCDIC along with their octal equivalent are shown in Table 2.3.2.

**Table 2.3.2: Alphabetic and numeric characters in EBCDIC along with their octal equivalent.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Character | EBCDIC Code | | Hexadecimal | Character | EBCDIC Code | | Hexadecimal |
| Zone | Digit | Zone | Digit |
| A | 1100 | 0001 | C1 | S | 1111 | 0001 | E2 |
| B | 1100 | 0010 | C2 | T | 1111 | 0010 | E3 |
|  |  |  |  |  |  |  |  |
| I | 1100 | 1001 | C9 | Y | 1111 | 1001 | E9 |
| J | 1101 | 0001 | D1 | 1 | 1111 | 0001 | F0 |
| K | 1101 | 0010 | D2 | 2 | 1111 | 0010 | F1 |
|  |  |  |  |  |  |  |  |
| R | 1101 | 1001 | D9 | 0 | 1111 | 1010 | F9 |

**2.2.5 ASCII**

American Standard Code for Information Interchange (ASCII) is a very widely used computer code. There are of two types: ASCII-7 and ASCII-8.

ASCII-7 is a 7-bit code that allows 128 () different characters. The first 3-bits are used as zone bits and the last 4-bits indicate the digit. Microcomputers using 8-bits byte use the 7-bits ASCII by leaving the leftmost bit of each byte as a zero. ASCII-8 is 8-bits code that allows 256 () different characters. Alphabetic and numeric characters in ASCII along with their hexadecimal equivalent are shown in Table 2.3.3

**Table 2.3.3: Alphabetic and numeric characters in ASCII along with their octal equivalent.**

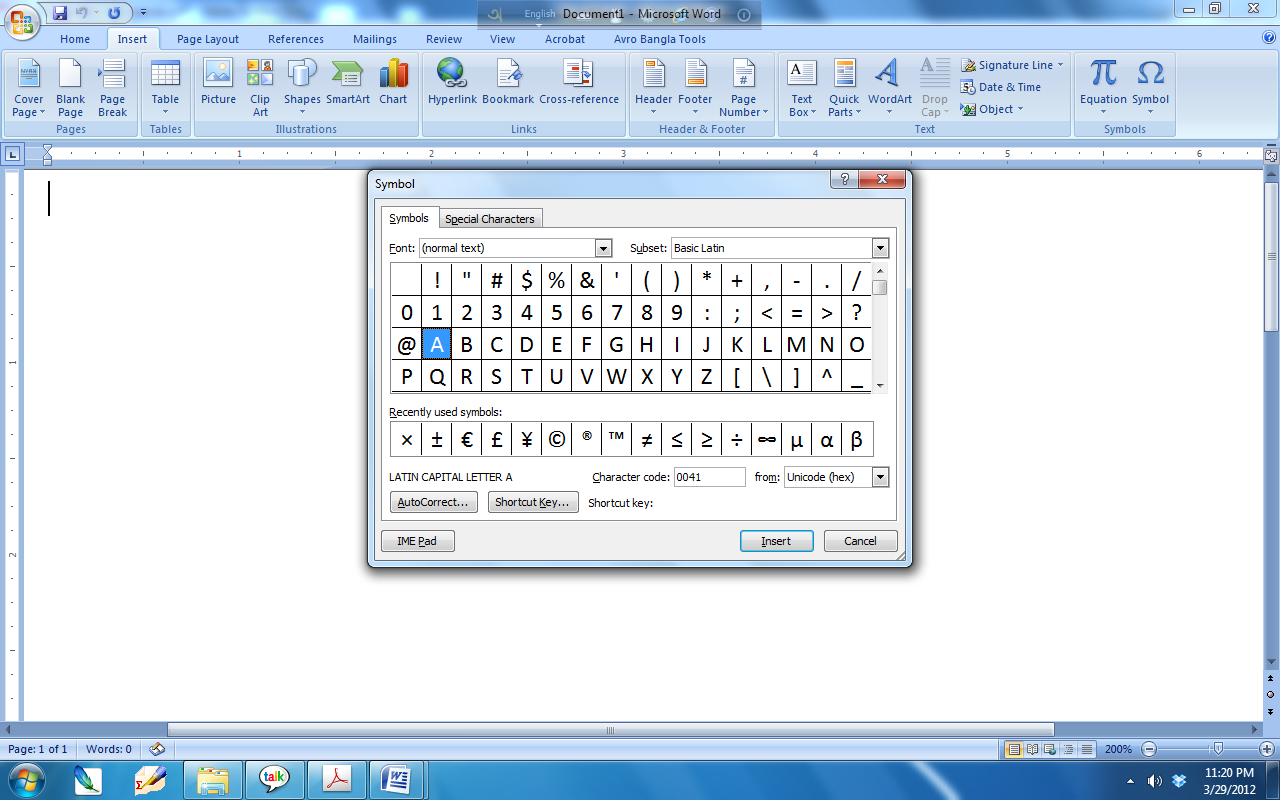
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Character | EBCDIC Code | | Hexadecimal | Character | EBCDIC Code | | Hexadecimal |
| Zone | Digit | Zone | Digit |
| 0 | 011 | 0000 | 30 | P | 101 | 0000 | 50 |
| 1 | 011 | 0001 | 31 | Q | 101 | 0001 | 51 |
|  |  |  |  |  |  |  |  |
| 9 | 011 | 0001 | 39 | Z | 101 | 1010 | 5A |
| A | 100 | 0001 | 41 |  |  |  |  |
| B | 100 | 0010 | 42 |  |  |  |  |
|  |  |  |  |  |  |  |  |
| O | 100 | 1111 | 4F |  |  |  |  |

**2.3.6 Unicode**

Unicode is a [computing](http://en.wikipedia.org/wiki/Computing) [industry standard](http://en.wikipedia.org/wiki/Technical_standard) for the consistent [encoding](http://en.wikipedia.org/wiki/Character_encoding), representation and handling of [text](http://en.wikipedia.org/wiki/Character_(computing)) expressed in most of the world's [writing systems](http://en.wikipedia.org/wiki/Writing_system). It provides a unique number for every character as shown in Figure 2.3.3, no matter what the platform, no matter what the program, no matter what the language. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystems, Microsoft, Oracle, SAP, Sun, Sybase, Unisys and [many others](http://www.unicode.org/consortium/memblogo.html).

Unlike [ASCII](http://www.webopedia.com/TERM/A/ASCII.htm), which uses 7 [bits](http://www.webopedia.com/TERM/B/bit.htm) for each character, Unicode uses 16 bits, which means that it can represent more than 65,000 unique characters. This is a bit of overkill for English and Western-European [languages](http://www.webopedia.com/TERM/L/language.htm), but it is necessary for some other languages, such as Greek, Chinese and Japanese. Many analysts believe that as the [software](http://www.webopedia.com/TERM/S/software.htm) industry becomes increasingly global, Unicode will eventually supplant ASCII as the standard character coding [format](http://www.webopedia.com/TERM/F/format.htm). The most recent Unicode version as of 2012 is Unicode 6.1.

The different encodings implement Unicode. The most commonly used encodings are UTF-8 (or UCS Transformation Format-8), UTF-16 and UCS-2. UCS-2 uses a 16-bit [code unit](http://en.wikipedia.org/wiki/Code_unit) (two [bytes](http://en.wikipedia.org/wiki/Octet_(computing))) for each character but cannot encode every character in the current Unicode standard and it is now obsolete. UTF-8 uses one byte for any [ASCII](http://en.wikipedia.org/wiki/ASCII) characters, and up to four bytes for other characters. UTF-16 extends UCS-2, using two 16-bit units to handle each of the additional characters.



**Figure 2.3.3 Unicode**

**2.2.7 Key points**

* Data are the names given to basic facts such as names and numbers.
* Information is processed data, that is, information is data which have been converted into a more useful form.
* Codes are used to reduce the volume of data.
* Numeric data are represented in the computer using binary numbers. So anything which has to be stored in the memory must be converted to a binary form and then the bits can be used.
* The left most bit in the word represents the sign bit. If the left most bit is 0, the number stored in the word will be treated as a positive integer.
* A real number (signed or unsigned) is divided into two parts: an integer part and a fractional part.
* The floating point representation can be used to represent numbers in any positional number system.
* The binary coded decimal (BCD) is based on the idea of converting each digit of a decimal number to its binary equivalent rather than converting the entire decimal value to a pure binary form.
* EBCDIC stands for extended binary coded decimal interchange code. In EBCDIC format, 8 bits are used to represent each character.
* American Standard Code for Information Interchange (ASCII) is a very widely used computer code. There are of two types: ASCII-7 and ASCII-8.
* Unicode is a [computing](http://en.wikipedia.org/wiki/Computing) [industry standard](http://en.wikipedia.org/wiki/Technical_standard) for the consistent [encoding](http://en.wikipedia.org/wiki/Character_encoding), representation and handling of [text](http://en.wikipedia.org/wiki/Character_(computing)) expressed in most of the world's [writing systems](http://en.wikipedia.org/wiki/Writing_system).

**2.2.8 Practice Sets**

**Multiple Choice Questions**

1. Code are used to reduce the volume of \_\_\_\_\_\_\_\_\_\_\_.
2. Information
3. Members
4. Numbers
5. Data
6. BCD code requires\_\_\_\_\_\_\_\_\_\_.
7. 3 bits for each decimal number
8. 4 bits for each decimal number
9. 8 bits for each decimal number
10. None
11. Unicode requires \_\_\_\_\_\_\_\_\_\_ bits.
12. 8
13. 16
14. 32
15. None

**Review Questions**

1. Distinguish among data, information and code.
2. Give some features of BCD, EBCDIC and ASCII codes.
3. What is a sign bit?
4. What is Unicode?

**Analytical Questions**

1. Explain numeric date representation.
2. Distinguish among BCD, EBCDIC and ASCII codes.
3. What is 2’s complementation representation? Why we use it.
4. Write a short note on Unicode.