CSC 218 Lecture Note 6

Data Representation, and Number Bases

Introduction

The knowledge of Computer Number Systems and arithmetic is fundamental to the understanding of how the computer machine operates. We have earlier defined a computer as a device that processes and manipulates information. And this information to be processed may be either numeric or non numeric. It is recalled that the digital computer understands what is called machine language only. Hence, the numeric and non-numeric information to be processed are represented within the computer in a string of 0's and 1's. Therefore, all information processed by the digital computer appearsas numbers whether or not is interpreted numerically.

Within any computer, all numbers must be expressed in binary form. Humans are very poorly adapted to manipulating binary data directly and usually work either in octal (base 8) or hexadecimal (base 16). The advantage of these two bases is that in each case they can be converted very easily to true binary, while at the same time arithmetic is very similar to normal decimal arithmetic. Octal and hexadecimal both provide a relatively compact way of expressing numbers, whereas large numbers expressed in binary quickly become unmanageable.

ASCII: The most widely used system for representation of data is ASCII – the American Standard Code for Information Interchange. This is an eight bit code which allows the alphabet (upper and lower case), numerals, grammatical characters (commas, question marks, etc), certain foreign language characters and graphics characters to be represented. Each character is given a numerical code in the range 0 to 255 (256 in all, that is 28).

Numerical Systems

Numerical data is stored in the computer for two reasons:
Retrieval purely for reference – In this first case we simply want a copy of the data. ASCII format is used or a system such as EBCDIC.
Retrieval for calculation – If we want to perform calculations then there are a number of problems to be overcome. It must be borne in mind that many different techniques have come and gone, and many techniques still exist.

EBCDIC Extended Binary Coded Decimal Interchange Code. This code has the same purpose as ASCII and is often used in IBM mainframes. The main difference between the two codes is the groupings of characters. For example:

Character	**ASCII**	**EBCDIC**
0	0101 0000	1111 0000
1	0101 0001	1111 0001
2	0101 0010	1111 0010
3	0101 0011	1111 0011
4	0101 0100	1111 0100

5	0101 0101	1111 0101
6	0101 0110	1111 0110
7	0101 0111	1111 0111
8	0101 1000	1111 1000

We can have listing of some of the more common number systems under the radix-weighted positional Number Systems in table below:

Table 5.1: Number Systems

Base	Number System	Right Symbols
2	Binary	0, 1
3	Ternary	0.1,2
4	Quaternary	0,1,2,3
5	Quinary	0,1,2,3,4,
8	Octal	0,1,2,3, 4,5,6,7
10	Decimal / Denary	0,1,2,3,4,5,6,7,8,9
12	Duodecimal	0,1,2,3,4,5,6,7,8,9,X,^
16	Hexadecimal	0,1,2,3,4,5,6,7,8,9,A,B C,D,E,F,

DATA REPRESENTATION

Data are the quantities, characters, or symbols on which operations are performed by a computer, being stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media.Aprogram is a set of data that consists of a series of coded software instructions to control the operation of a computer or other machine.Physical computer memory elements consist of an address and a byte/word of data storage.

All data are representing by code. Computer must not only be able to carry out computations, they must be able to do them quickly and efficiently. There are several data representations, typically for integers, real numbers, characters, and logical values.
A numeral system is a collection of symbols used to represent small numbers, together with a system of rules for representing larger numbers. Each numeral system uses a set of digits. The number of various unique digits, including zero, that a numeral system uses to represent numbers is called base or radix.

For your information, our computer only recognizes two types of digits, which is 1 and 0. They are also called a bit as each 1 or 0 is one bit in the binary system. 1 and 0 also represent on and off state and yes or no.

Bit is the smallest unit in a computer binary system which can be understood by computer. One bit can also be combined to form a much more complex number. Next to the bit, it is a byte. One byte is a combination of 8 bit. Example of a byte is 10110110. There is a total of 256 combination of this number.

There are three character codes to represent characters which are ASCII, EBCDIC and Unicode. Each byte contains eight bits. A byte provides enough different combination of 0s and 1s to represent 256 characters.

The combinations of 0s and 1s are defined by patterns. These patterns are called coding scheme. The 256-character capability of ASCII and EBCDIC is too small to handle the characters that are used by other languages such as Arabic, Japanese and Chinese.

The Unicode coding scheme is designed to solve this problem. It uses two bytes (16 bits) to represent one character. Unicode will have more than 65,000 different characters. This can cover all the world’s languages.

BASE CONVERSION

1. 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.

$1\ 1\ 0\ 1\ 1_2$	(binary)
$2^4+2^3+0+2^1+2^0$	$= 16+8+0+2+1$
	$= 27_{10}$ (decimal)

$1\ 0\ 1\ 1\ 0\ 1\ 0\ 1_2$	(binary)
$2^7+0+2^5+2^4+0+2^2+0+2^0$	$= 128+0+32+16+0+4+0+1$
	$= 181_{10}$ (decimal)

2. Decimal-To-Binary Conversion

(A) Reverse of Binary-To-Digital Method

$45_{10} = 32 + 0 + 8 + 4 +0 + 1$

$= 2^5+0+2^3+2^2+0+2^0$

$= 1\ 0\ 1\ 1\ 0\ 1_2$

(B) Repeat Division

This method uses repeated division by 2. e.g. convert 25_{10} to binary

25/ 2	= 12+ remainder of **1**	**1 (Least Significant Bit)**
12/ 2	= 6 + remainder of 0	0
6 / 2	= 3 + remainder of 0	0
3 / 2	= 1 + remainder of 1	1
1 / 2	= 0 + remainder of **1**	**1 (Most Significant Bit)**
Result	25_{10} =	**1** 1 0 0 **1**$_2$

Octal Number System

The octal number system has a base of eight, meaning that it has eight possible digits: 0, 1,2,3,4,5,6,7.

1. Octal to Decimal Conversion

e.g. $24.6_8 = 2 \times (8^1) + 4 \times (8^0) + 6 \times (8^{-1}) = 20.75_{10}$

2. Binary-To-Octal / Octal-To-Binary Conversion

Each Octal digit is represented by three bits of binary digit.

e.g. $100\ 111\ 010_2 = (100)\ (111)\ (010)_2 = 4\ 7\ 2_8$

3. Repeat Division

This method uses repeated division by 8. e.g. convert 177_{10} to octal and binary:

177/8	= 22+ remainder of **1**	**1 (Least Significant Bit)**
22/ 8	= 2 + remainder of 6	6
2 / 8	= 0 + remainder of **2**	***2 (Most Significant Bit)***
Result	177_{10} =	***261***$_8$
	Convert to Binary	$= 010110001_2$

Hexadecimal Number System

The hexadecimal system uses base 16. Thus, it has 16 possible digit symbols.

It uses the digits 0 through 9 plus the letters A, B, C, D, E, and F as the 16 digit symbols.

1. Hexadecimal to Decimal Conversion

e.g. $2AF_{16} = 2 \times (16^2) + 10 \times (16^1) + 15 \times (16^0) = 687_{10}$

Repeat Division: Convert decimal to hexadecimal

This method uses repeated division by 16. e.g. convert 378_{10} to hexadecimal and binary:

378/16	= 23+ remainder of **10**	**A (Least Significant Bit)**
23/ 16	= 1 + remainder of 7	7
1 / 16	= 0 + remainder of **1**	***1 (Most Significant Bit)***
Result	378_{10} =	***17*****A**$_8$
	Convert to Binary	= $0001\ 0111\ 1010_2$
		= *0000* 0001 0111 1010 (16 bits)

2. Binary-To-Hexadecimal /Hexadecimal-To-Binary Conversion

Each Hexadecimal digit is represented by four bits of binary digit.

e.g. $1011\ 0010\ 1111_2 = (1011)\ (0010)\ (1111)_2 = B\ 2\ F_{16}$

3. Octal-To-Hexadecimal /Hexadecimal-To-Octal Conversion

1) Convert Octal (Hexadecimal) to Binary first.

2a) Regroup the binary number in 3 bits a group *starts from the LSB* if Octal is required.

2b) Regroup the binary number in 4 bits a group from the LSB if Hexadecimal isrequired.

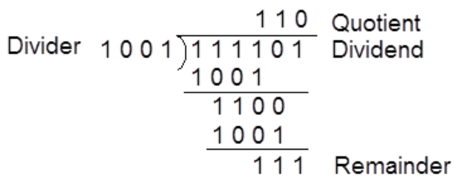
e.g. Convert $5A8_{16}$ to Octal.

$5A8_{16}$ = 0101 1010 1000 (Binary)

= 2650(Octal)

Number system Division

1. Binary Arithmetic Division



2. Octal Division

```
             114      Quotient
Divider  63 )7514     Dividend
             63
            ----
             114
              63
            ----
             364
             314
            ----
              50      Remainder
```

3. Hexadecimal Division

```
                 79B     Quotient
Divider   B9)57F6D       Dividend
             50F
            -----
              706
              681
            -----
               85D
               7F3
            -----
                6A       Remainder
```

Simple conversion problems

E.g.1 Convert the binary number

(11011.10) $_2$to decimal

Solution

We now write this number as a polynomial in powers of a 2

$(11011.10)_2 = 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 + 1 * 2^{-1} + 0 * 2^{-2}$

$= 1 * 16 + 1 * 8 + 0 * 4 + 1 * 2 + 1 * 1 + 1 * {}^1/_2 + 0 * 1/4$

$= 16 + 8 + 0 + 2 + 1. {}^1/_2 + 0 = 27.5_{10}$

$= 11011.10_2 = 27.5_{10}$

E.g. 2: Convert the hexadecimal number (E 6 F.5) $_{16}$ into a decimal

Solution Again, we put down the number as a polynomial

(E 6F.5) = $Ex16^2 + 6x\ 16^1 + Fx16^0 + 5x16^{-1}$

= $14x16^2$ x $1x\ 16^1$ $+15x16^0$ $+5x\ 16^{-1}$

= 14 x 256 +6x 16 + 15 x 1 + 5 x 0.0625

= 3584 + 96 +15 + 0.3125

= 3695.3125 or $(3695.3125)_{10}$

E.g. 3: Convert the octal number (372.46) $_2$ to decimal

Solution$(372.46)_8 = 3 \text{ x } 8^2 + 7 + 8^1 + 2 \text{ x } 8^o + 4 \text{ x } 8^{-1} + 6 \text{ x } 8^{-2}$

= 3 x 64 + 7 x 8 + 2 + 0.125 + 0.9375

= 192 + 56 + 2 + 0.125 + 0. 09375

= 250.21875 or $(250.21875)_{10}$

E.g. 4: Convert 110 111001001 to octal

Solution$1*2^{11}+1*2^{10}+0*2^9+1*2^8+1*2^7+ 1*2^6+0*2^5+0*2^4+1*2^3+0*2^2+0*2^1+1*2^0$

= 2048+1024+0+256+128+64+0+0+8+0+0+1 = $\mathbf{3529_{10}}$

$3529_{10}$to base 8 are as follows:

	3529		
8	441	r	1
8	55	r	1
8	6	r	7

0 r 6

$3529 = 6711_8$

:. $110111001001 = \mathbf{6711_8}$

The alternative and quicker method for solving the question in example 4 is to use group binary digit as in e.g. 5.

E.g. 5: Covert 110111001001 to octal

Solution: Using group binary digit method

```
1 1 0 | 1  1  1  0 | 0  1   0 | 0  1
  6   |  7     1   |    1     |
```

:. $110111001001_2 = 6711_8$

E.g. 6: Convert the decimal number 52.16 into its binary equivalent

Solution:

Here, the integer portion 52 is converted by repeated division and afterwards the fractional portion 0.16 by repeated multiplication.

```
2 | 52          ↑
2 | 26   R 0    |          2x 0.16 = 0.32 = 0     |
2 | 13   R 0    |          2 x 0.32 =0.64 = 0     |
2 |  6   R 1    |          2 x0.62 = 1.24 = 1     |
2 |  3   R 0    |          2 x 0.24 = 0.48 = 0    |
2 |  1   R 1    |          2 x 0.48 = 0.96 = 0    |
2 |  0   R1     |          2 x 0.96 = 1.92 = 1    ↓
```

:. Collecting the generated digits, we have $(110100.001001\ldots)_2$

E.g. 7:

Convert the decimal number, 58506.8435 into hexadecimal

Solution

Again, they are sorted out differently;

```
   | 58506
                              Digit Symbol Equivalent
16 |  3656   R   10                A ↑
16 |   228   R   8                 8 |
16 |   14    R   4                 4 |
       0     R   14                E |
```

:. We have E48A for the integral portion. Now, for the fractional portion, the solution is as follows;

	Digit Symbol Equivalent
16 x 0.8435 = 13.496 = 13	D
16 x 0.496 = 7.936 = 7	7
16 x 0. 936 = 14.976 = 14	E
16 x 0.976 = 15.616 = 15	F ↓

The multiplication is terminated as the desired precision has been obtained.
In the final solution is, $(E48A.\ D7EF)_{16}$

E.g. 8: Convert the following binary numbers (i) 11011001.1011 (ii) 1011100101.111 into octal and hexadecimal number systems.

Solution:
The binary numbers are now grouped accordingly:
(i) (a) 11011001.1011 (b) 11011001.1011

Octal:					**hexadecimal**		
011	011	001.	101	100	1101	1001	. 1011
3	3	1	5	4	D	9	. B
=	$(331.54)_8$				= $(D\ 9.B)_{16}$		

E.g. 9: Convert the following octal and Hexadecimal numbers into binary numbers: (i) $(275)_8$ $(345.6)_8$ $(A5D.F)_{16}$ $(7BF.A)_{16}$
Solution:
The processes involved in solving example 8 are now reversed in order to solve the problems.

(i) (a) $(275)_8$
= 27 5
= 010 111 101
= $(010111101)_2$

(b) $(345\text{-}6)_8$
= 3 4 5 . 6
= 011 100 101 110
= $(011100101.110)_2$

(ii) (a) $(A5D.F)_{16}$
= A 5 D F
= 1010 0101 1101 . 1111
= $(101001011101.1111)_2$

(b) $(7BF.\ A)_{16}$
= 7 B F . A
= 0111 1011 1111 1010
= $(011110111111.1010)_2$

Complement of a number
In digital work, two types of complements of a binary number are used for complemental subtraction.

1's complement: The 1's complement of a binary number is obtained by changing its each "0" into a "1" and each "1" into a "0". It is also called radix-minus-one complement.
For example, 1's complement of 100_2 is 011_2 and 1110_2 is 0001_2
2's complement of a binary number is obtained by adding 1 to its 1's complement 2's complement = 1's complement + 1. It is also known as true complement. The complemental method of subtraction reduces subtraction to an addition process. This method is popular in digital computers because only adder circuits are needed thus simplifying the circuits to get the complement It is easy with digital circuits to get the complements.

1's Complemental Subtraction: In this method, instead of subtracting a number, we add its 1's complement of the subtrahend to the minuend. The last carry (whether 0 or 1) is then added to get the final answer.
The rules for subtraction by 1's complement are as under
1. Compute the 1's complement of the **subtrahend** by changing all its 1's to 0's and all its 0's to 1's
2. Add this complement to the **minuend**
3. Perform the end-around carry of the last 1 or 0
4. If there is no end-around carry (i.e. 0 carry) then the answer must be recomplemented and a negative sign is attached to it.
5. If the end-around carry is 1, no recomplementing is necessary. Then add the end around carry to the normal digits

E.g. 1 Subtract 101_2 from 111_2.
Solution

$$\begin{array}{rrrrl} & 1 & 1 & 1 & \\ + & 0 & 1 & 0 & \longrightarrow \text{1's complement of subtrahend 101} \\ \hline 1 & 0 & 0 & 1 & \\ & & + & 1 & \longrightarrow \text{end-round carry} \\ \hline & 0 & 1 & 0 & \\ \hline \end{array}$$

As seen, we have removed from the addition sum the 1 carry in the last position and added it onto the remainder. It is called end-around carry.

E.g.2 Subtract 1101_2 from 1010_2

$$\begin{array}{rrrr} 1 & 0 & 1 & 0 \end{array}$$

$$\begin{array}{cccccl} + & 0 & 0 & 1 & 0 & \longrightarrow \text{1's complement of 1101} \\ \hline & 1 & 1 & 0 & 0 & \\ \hline \end{array}$$

No carry

As seen, there is no end-around carry in this case. Hence, as per Rule 4 given above, answer must be recomplemented to get 0011 and a negative sign attached to it. Therefore, the final answer becomes – **0011**.

E.g. 3 Consider the complemental subtraction of 1110_2 from 0110_2

$$\begin{array}{cccccl} & 0 & 1 & 1 & 0 & \\ + & 0 & 0 & 0 & 1 & \longrightarrow \text{1's complement of } 1110_2 \\ \hline & 0 & 1 & 1 & 1 & \\ \hline \end{array}$$

No Carry

As seen, there is no carry. However, we may add an extra 0 from our side to make it a 0 carry as shown below:

$$\begin{array}{cccccl} & 0 & 1 & 1 & 0 & \\ + & 0 & 0 & 0 & 1 & \longrightarrow \text{1's complement} \\ \hline 0 & 0 & 1 & 1 & 1 & \\ & & & + & 0 & \longrightarrow \text{end-around carry} \\ \hline & 0 & 1 & 1 & 1 & \\ \hline \end{array}$$

After recomplementing, it becomes 1000. When negative sign is attached, the final answer becomes –$\mathbf{1000_2}$.

E.g.:Using 1'scomplemental method, subtract 01101_2 from 11011_2

Solution

$$\begin{array}{ccccccl} & 1 & 1 & 0 & 1 & 1 & \\ + \ 1 & 0 & 0 & 1 & 0 & & \longrightarrow \text{1's complement of subtrahend} \\ \hline \mathbf{1} & 0 & 1 & 1 & 0 & 1 & \\ & & & & + & 1 & \longrightarrow \text{end around carry} \\ \hline & 0 & 1 & 1 & 1 & 0 & \\ \hline \end{array}$$

Since end-around carry is 1, we take the final answer as it is in (Rule 5) which implies 01110

2's Complemental subtraction

In this case, the procedure is as under

1. find the 2's complement of the subtrahend,
2. add this complement to the minuend,
3. drop the final carry
4. If the carry 1, the answer is positive and needs no recomplementing
5. if there is no carry, recomplement the answer and attach minus sign.

E.g. 1 Use 2's complement to subtract 1101_2 from 1010_2

Solution

The 1's complement of 1101 is 0010. The 2's complement is 0011.

$$\begin{array}{cccccccl} & & 1 & 0 & 1 & 0 & & \\ & + & 0 & 0 & 1 & 1 & \longrightarrow & \text{2's complement of 1101} \\ \cline{3-6} & \nearrow & 1 & 1 & 0 & 1 & & \\ \cline{3-6} \end{array}$$

No carry

In this case, there is no carry. Hence, we have to recomplement the answer. For this purpose, we first subtract 1 from it to get 1100. Next we recomplement it together 0011. After attaching the minus sign, the final answer becomes – 0011_2

Taking in terms of decimal numbers, we have subtracted 13 from 10. Obviously the answer is – 3 (negative 3)

E.g. 2: Using 2's complement, subtract 1010_2 from 1101_2

Solution 2: The 1's complement of 1010 is 0101. The 2's complement is 0101+1=0110.

We will add it to 1101

$$\begin{array}{cccccl} & 1 & 1 & 0 & 1 & \\ & 0 & 1 & 1 & 0 & \longrightarrow \text{2's complement of } 1010_2 \\ \hline 1 & 0 & 0 & 1 & 1 & \\ \hline \uparrow & & & & & \\ \text{DROP} & & & & & \end{array}$$

The final answer is 0011_2

COMPUTER CODES

The letters, numbers and special characters that we input into the computer are stored using a computer code. The computer code is a way of representing each character using only the 0 and 1 binary bits that the computer understands. Two computer codes that are used today are EBCDIC and ASCII. ASCII is the code that is used on your microcomputer.

EBCDIC (Extended Binary Coded Decimal Interchange Code):

EBCDIC is an 8 bit code where the left four bits are called the zone and the right four bits are called the digit portion. The zone part of the code tells what group the character is part of, while the digit part of the code identifies the specific character within the group. EBCDIC has codes for upper case and lower case letters, numbers and special characters.

The structure of the EBCDIC code is:

- - - - - - - -

Zone/digit

<u>Zone</u>: 1 1 _ _ The 11 in the first two digits of the zone indicates an upper case letter or a number.

The last two digits of the zone represent the group or range:

code	**range**
00	A - I
01	J - R
10	S - Z
11	numbers

1. Hexadecimal translation of EBCDIC:

Instead of dealing with the 8-bit binary code when looking at memory dumps etc., the programmer usually deals with the hexadecimal translation of EBCDIC. One hexadecimal character is used to translate the 4-bit digit portion of the EBCDIC. Hexadecimal is used to represent EBCDIC because of their unique relationship: any 4-bit binary number can be represented by a single hexadecimal character and any hexadecimal character can be represented by 4 binary bits.

Characters	**EBCDIC Zone**	**Hexadecimal representation**
A -I	1100	C
J - R	1101	D
S - Z	1110	E
numbers	1111	F

2. Decimal Equivalents of EBCDIC:

Another way of representing EBCDIC is using the decimal equivalent of the entire 8 bit code binary code:

	1	1	0	0	0	0	0	1	Face value
A =	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Powers of 2
	128	64	32	16	8	4	2	1	Positional value from appropriate power of 2

Therefore, A = 128 + 64 + 1 = 193.,B = 194, C = 195 ………Z = 233

ASCII (American Standard Code for Information Interchange):

Most PCs can handle ASCII files and so if you store a file in ASCII it is usually transportable from system to system. There is a standard 7 bit ASCII code that has characters whose binary code translates into the decimal numbers from 0 to 127. In this set, codes 0 through 32 are control codes that handle things like form feed and carriage control while codes 33 through 127 are a set of printable characters. For example, codes 65 through 90 represent uppercase letters and codes 48 through 57 represent the digits 0 through 9. However, most computers handle 8 bit codes so it is possible to generate 128 additional ASCII codes, which are called the extended ASCII character set. These codes are not standardized and are sued differently by different computer manufacturers. When representing the standard 7 bit ASCII codes as 8 bit codes, a leading 0 is added as the left-most character. When dealing with ASCII code, the left most bits will tell us the type of character being represented and the right bits will indicate which one in the range. For example, 011 in the left most three bits of the standard 7 bit code indicates a number with 0 having 0000 standing for 1 and 1001 standing for 9.

1. ASCII Equivalent represented in decimal translation of binary code

When working with ASCII, again it is difficult to deal with the 7 or 8 bit string of binary digits. Therefore, frequently the code is represented by the decimal equivalent of the binary code for ease of use.

The letter Z has an equivalent value of 90 as shown:

1	0	1	1	0	1	0	face value
2^6	2^5	2^4	2^3	2^2	2^1	2^0	positional value in powers of 2
64	32	16	8	4	2	1	positional value in decimal

The equivalent value is therefore 64 + 16 + 8 + 2 for a total of 90.

A=65, B=66, C=67,……….Z=90

Binary to Gray Code Conversion

1. Place a leading zero before the MSB
2. Compare the adjacent bit starting from the leading zero
3. If the two compared bits are same, it implies 0, if not, it implies 1

$$\begin{tikzcd}
\text{E.g.} & 5 & & 1 & 0 & 1 & \text{Binary} \\
& & 0 \arrow[ur] & 1 \arrow[ur] & 0 \arrow[ur] & 1 & \\
& & \mathbf{1} & \mathbf{1} & \mathbf{1} & & \text{Gray Code}
\end{tikzcd}$$

Gray Code to Binary Conversion

1. Drop the MSB
2. Compare this to the next adjacent bit
3. If the compared bits are the same, it implies 0, if not, it implies 1

$$\begin{tikzcd}
\text{E.g.} & 5 & 1 & 1 & 1 & \text{Gray code} \\
& & \mathbf{1} \arrow[ur] & \mathbf{0} \arrow[ur] & \mathbf{1} & \text{Binary}
\end{tikzcd}$$