Each digit appearing to the left of the decimal point represents a value between zero and nine times an increasing power of ten. Digits appearing to the right of the decimal point represent a value between zero and nine times an increasing negative power of ten.

e.g. 123.456 means

$1*10^2 + 2*10^1 + 3*10^0 + 4*10^{-1} + 5*10^{-2} + 6*10^{-3}$

or 100 + 20 +3 +0.4 + 0.05 +0.006

1.4 THE HEXADECIMAL NUMBERING SYSTEM

A big problem with the binary system is verbosity. To represent the value 202 (decimal) requires eight binary digits. The decimal version requires only three decimal digits and, thus, represents numbers much more compactly than does the binary numbering system. This fact was not lost on the engineers who designed binary computer systems. When dealing with large values, binary numbers quickly become too unwieldy. Unfortunately, the computer thinks in binary, so most of the time it is convenient to use the binary numbering system. Although we can convert between decimal and binary, the conversion is not a trivial task. The hexadecimal (base 16) numbering system solves these problems. Hexadecimal numbers offer the two features we're looking for: they're very compact, and it's simple to convert them to binary and vice versa. Because of this, most binary computer systems today use the hexadecimal numbering system. Since the radix (base) of a hexadecimal number is 16, each hexadecimal digit to the left of the hexadecimal point represents some value times a successive power of 16. For example, the number 1234 (hexadecimal) is equal to:

1 * 16**3 + 2 * 16**2 + 3 * 16**1 + 4 * 16**0 or
4096 + 512 + 48 + 4 = 4660 (decimal).

Each hexadecimal digit can represent one of sixteen values between 0 and 15. Since there are only ten decimal digits, we need to invent six additional digits to represent the values in the range 10 through 15. Rather than create new symbols for these digits, we'll use the letters A through F. The following are all examples of valid hexadecimal numbers:

1234 DEAD BEEF 0AFB FEED DEAF

Since we'll often need to enter hexadecimal numbers into the computer system, we'll need a different mechanism for representing hexadecimal numbers. After all, on most computer systems you cannot enter a subscript to denote the radix of the associated value. We'll adopt the following conventions:

- All numeric values (regardless of their radix) begin with a decimal digit.
- All hexadecimal values end with the letter "h", e.g., 123A4h.
- All binary values end with the letter "b".
- Decimal numbers may have a "t" or "d" suffix.

Examples of valid hexadecimal numbers:

1234h 0DEADh 0BEEFh 0AFBh 0FEEDh 0DEAFh

As you can see, hexadecimal numbers are compact and easy to read. In addition, you can easily convert between hexadecimal and binary. Consider the following table:

Binary/Hex Conversion	
Binary	**Hexadecimal**
0000	**0**
0001	**1**
0010	**2**
0011	**3**
0100	**4**
0101	**5**
0110	**6**
0111	**7**
1000	**8**
1001	**9**

1010	**A**
1011	**B**
1100	**C**
1101	**D**
1110	**E**
1111	**F**

This table provides all the information you'll ever need to convert any hexadecimal number into a binary number or vice versa.

To convert a hexadecimal number into a binary number, simply substitute the corresponding four bits for each hexadecimal digit in the number. For example, to convert 0ABCDh into a binary value, simply convert each hexadecimal digit according to the table above:

0 A B C D Hexadecimal

0000 1010 1011 1100 1101 Binary

To convert a binary number into hexadecimal format is almost as easy. The first step is to pad the binary number with zeros to make sure that there is a multiple of four bits in the number. For example, given the binary number 1011001010, the first step would be to add two bits to the left of the number so that it contains 12 bits. The converted binary value is 001011001010. The next step is to separate the binary value into groups of four bits, e.g., 0010 1100 1010. Finally, look up these binary values in the table above and substitute the appropriate hexadecimal digits, e.g., 2CA. Contrast this with the difficulty of conversion between decimal and binary or decimal and hexadecimal!

Since converting between hexadecimal and binary is an operation you will need to perform over and over again, you should take a few minutes and memorize the table above. Even if you have a calculator that will do the conversion for you, you'll find manual conversion to be a lot faster and more convenient when converting between binary and hex.

A comparison of the afore mentioned numbering systems is shown below;

binary	octal	decimal	Hexadecimal
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

CHAPTER THREE

3.0 TYPES OF ENCODING

When numbers, letters and words are represented by a special group of symbols, this is called "**Encoding"** and the group of symbol encoded is called a "**code".** Any decimal number can be represented by an equivalent binary number. When a decimal number is represented by its equivalent binary number, it is called "**straight binary coding**".

Basically, there are three methods of encoding and they are;

- American Standard Code for Information Interchange (ASCII)
- Binary Coded Decimal (BCD)
- Extended Binary Coded Decimal Interchange Code(EBCDIC)

3.1 ASCII CODING SYSTEM

In addition to numeric data, a computer must be able to handle non- numeric information. In order words, a computer should recognize codes that represents letters of the alphabets, punctuation marks, other special characters as well as numbers. These codes are called ***alphanumeric codes.*** The most widely used alphanumeric code is ASCII code (American Standard Code for Information Interchange). ASCII is used in most micro computers and mini computers and in many main frames. The ASCII code is a seven bit code, thus it has 2^7=128 possible code groups. In the 7 bits code, the first 3 bits represent the zone bits and the last 4 bits represent the numeric bits.

Despite some major shortcomings, ASCII data is *the* standard for data interchange across computer systems and programs. Most programs can accept ASCII data; likewise most programs can produce ASCII data. Since you will be dealing with ASCII characters in assembly language, it would be wise to study the layout of the character set and memorize a few key ASCII codes (e.g., "0", "A", "a", etc.).

The table below shows some commonly used ASCII codes

ZONE BITS					**NUMERIC BITS**			
011	100	101	110	111	8	4	2	1
0		P		p	0	0	0	0
1	A	Q	a	q	0	0	0	1
2	B	R	b	r	0	0	1	0
3	C	S	c	s	0	0	1	1
4	D	T	d	t	0	1	0	0
5	E	U	e	u	0	1	0	1
6	F	V	f	v	0	1	1	0
7	G	W	g	w	0	1	1	1
8	H	X	h	x	1	0	0	0
9	I	Y	i	y	1	0	0	1
	J	Z	j	z	1	0	1	0
	K		k		1	0	1	1
	L		l		1	1	0	0
	M		m		1	1	0	1
	N		n		1	1	1	0
	O		o		1	1	1	1

A summary of ASCII table is shown below;

Characters	Zonebits	Numeric bits
0-9	011	0000-1001
A-O	100	0001-1111
P-Z	101	0000-1010
a-o	110	0001-1111
p-z	111	0000-1010

Examples

1. Represent Bez in binary format

Since Bez contains an alphabets, the ASCII representation is suitable for this conversion

B- 100 0010

e- 110 0101

z- 111 1010

Answer: 100001011001011111010

2. Convert 1001000 1000101 1001100 1010000 to ASCII

1001000- H

1000101- E

1001100- L

1010000- P

Answer: HELP

3. Using ASCII representation, convert ***UNIVERSITY*** to binary

U- 1010101, N- 1001110, I- 1001001, V- 1000110, E- 1000101, R- 1010010, S- 1010011, I- 1001001,

T-1010100, Y- 1011001

ANSWER: 1010101100111010010011000110100010110100101010011100 1001

3.2 BINARY CODED DECIMAL

If each digit of a decimal number is represented by binary equivalence, this produces a code called Binary Coded Decimal. Since a decimal digit can be as large as 9, 4 bits are required to code each digit in the decimal number. E.g.

$874_{10} = 100001110100_2$

$943_{10} = 100101000011_2$

Only the four bits binary numbers from 0000 through 1001 are used for binary coded decimal. The BCD code does not use the numbers 10, 11, 12, 13, 14, 15. In other words, 10 of the 16 possible 4 bits binary codes are used. If any of these forbidden 4 bits number ever occurs in a machine using the BCD, 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, hexadecimal and decimal. It is in fact the decimal system with each digit encoded in its binary equivalence. It is also important to understand that a BCD number is not the same as binary number.

A straight binary code takes the complete decimal number and represents it in binary while the BCD code converts each decimal digit to binary individually.

e.g.

137_{10} to straight binary coding is 10001001

137_{10} to BCD is 000100110111

The main advantage of BCD is the relative ease of converting to and from decimal. This ease of conversion is especially important from a hardware standpoint because in a digital system, it is the logic circuit that performs conversion to and from decimal.

BCD is used in digital machines whenever decimal information is either applied as input or displayed as output. e.g. digital voltmeter, frequency counters make use of BCD. Electronic calculators also make use of BCD because the input numbers are entered in decimal through the keyboard and the output is displayed in decimal.

BCD is not often used in modern high speed digital system for good 2 good reasons;

1. As it was already pointed out, the BCD code for a given decimal number requires more bits that the straight binary code and it is therefore less efficient. This is important in digital computers because the number of places in memory where the bits can be stored is limited.
2. The arithmetic processes for numbers represented in BCD code are more complicated than straight binary and thus requires more complex circuitry which contributes to a decrease in the speed at which arithmetic operations take place.

3.3 EBCDIC CODING SYSTEM

EBCDIC is an acronym for ***Extended Binary Coded Decimal Interchange Code***. IBM developed this code for use on its computers. In EBCDIC, eight bits are used to represent each character i.e 256 characters can be represented. IBM minicomputers and mainframe computers use the EBCDIC. The eight bits can as well be divided into two. The zonebits and the numeric bits each is represented by 4bits.

Zonebits				Numeric bits			
1111	1100	1101	1100	8	4	2	1
0				0	0	0	0
1	A	J	S	0	0	0	1
2	B	K	T	0	0	1	0
3	C	L	U	0	0	1	1
4	D	M	V	0	1	0	0
5	E	N	W	0	1	0	1
6	F	O	X	0	1	1	0
7	G	P	Y	0	1	1	1
8	H	Q	Z	1	0	0	0
9	I	R		1	0	0	1

Example: Convert **HELP** to binary using EBCDIC coding system.

Solution:

H- 11001000, E- 11000101, L- 11010011, P- 11010111