Week Two Lecture Note

INFORMATION OR DATA REPRESENTATION

There are three ways of representing information or data in computer system. These are as follows:

- Number System
- Character codes
- Logic gates and Truth Table

4.1 NUMBER SYSTEMS

Number Systems include Binary number, Octal number, Hexadecimal number, Binary Fraction, Arithmetic operation, Negative numbers, Fixed-point and Floating Point representation and Errors.

Binary Number – This method is used to represent information with two-state circuits. These two states are 0 or 1. These two states are used in computer because they are quick, reliable and take up only small amounts of space and energy. They are called binary digits or bits.

Conversion from Binary to Decimal

e.g. $0110 = 0x2^3 + 1x2^2 + 1\ x2^1 + 0 + 2^0$

$= \quad 0 + 4 + 2 + 0$

$0110_2 = 6_{10}$

Conversion from Decimal to Binary

2	6	
2	3 remainder	0
2	1 remainder	1
	0 remainder	1

$6_{10} = 0110_2$

OCTAL NUMBER – Octal numbers are in base 8

Conversion from Decimal to Octal
49_{10} to base 8

8	49		
8	6	remainder	1
	0	remainder	6

$49_{10} = 061_8$

Conversion from Octal to Decimal
061_8 to base 10

$0 \times 8^2 + 6 \times 8^1 + 1 \times 8^0$

$0 + 48 + 1 = 49_{10}$

Conversion between Octal and Binary

Octal	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

Convert 74_8 to binary

Octal	7	4
Binary	111	100

$74_8 = 11100_2$

Convert 110011001100_2 to Octal

$1 \times 2^{11} + 1 \times 2^{10} + 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2$

2048 + 1024 + 128 + 64 + 8 + 4

3276_{10} to base 8

8 3276
8 409 rem. 4

8 51 rem. 1

6 rem. 3

110011001100_2 = 6314

Another method is to use the group binary digits where each number is converted to its binary equivalent.

Group binary digits = 110 011 001 100

6 3 1 4

Decimal Number	Hexadecimal Number	Binary Number
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexadecimal Numbers

Convert from Decimal to Hexadecimal

77_{10} to Hex.

16 77

4 R 13D

$= 4D_H$

Convert from Hexadecimal to Decimal

$4D_H$ to Decimal

$4 \times 16^1 + 13 \times 16^0$

64 + 13 = 77

Convert $4D_H$ to Binary

Hexadecimal Number	4	D
Binary Equivalent	0100	1101

Convert $4D_H$ to Octal

$4D_H$ = 01001101 in Binary

001 001 101

1 1 5

$4D_H$ = 115_8

Convert 111001010_2 to Hexadecimal

0001 1100 1010

1 12 10

111001010_2 = $1CA_H$

Binary Fraction

2^{-1}	2^{-2}	2^{-3}	2^{-4}
½	¼	1/8	1/16
0.5	0.25	0.125	0.0625

Convert 0.1000 in base 2 to Decimal

0.1000 = $1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4}$

= 1 x 1/2 + 0 + 0 + 0

= ½ + 0 + 0 + 0

½ = 0.5

Convert 0.0100 to Decimal

0.0100 = $0 \times 2^{-1} + 1 \times 2^{-3} + 0 \times 2^{-3} + 0 \times 2^{-3}$

0 + 1 x ¼ + 0 + 0

¼ = 0.25

0.125 to binary

125/1000 = 1/8 = 2^{-3}

$0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = .001$

0.125 = .001

Computer Arithmetic

The arithmetic and logic unit of the central processor in a computer performs all its arithmetic by addition and shifting.

Binary addition 8_{10} 1000_2

$+7_{10}$ 0111_2

15_{10} 1111_2

Binary multiplication

5_{10} - 101_2

2_{10} - 010_2

- 000

101

___ 000

10_{10} 1010_2

Binary Division
$8_{10}/4_{10} = 2_{10}$

$1000_2/100_2 = 10_2$

Negative Numbers
There are several common methods for representing negative numbers on computers including

(a) Sign-magnitude

(b) One's Component

(c) Two's Complement

Sign Magnitude Method
In the sign magnitude method, the negative number is simply the positive number with the sign bit inverted e.g.

7 = 00000111

-7 = 10000111 – sign magnitude

21 = 00010101

-21 = 10010101 – sign magnitude

The sign magnitude representation makes it to obtain the magnitude and the negative of a number. The magnitude is obtained by clearing the most significant bit and the negative by inverting the most significant bit. The sign magnitude representation is frequently used in Analog to Digital and Digital to Analog conversion

One's Complement
In the one's complement representation, the negative of a number is the logical complement (that is, replace all the zeros in the binary representation by ones and all the ones by zero.

e.g.

7 = 00000111

-7 = 11111000 (one's complement)

21 = 00010101

-21 = 11101010 (one's complements)

The problem with one's complement is that the one's complement of zero is not zero – that is

0 = 00000000

-0 = 11111111 (one's complement)

This confusion has made the one's complement representation of negative obsolete.

The sign-magnitude representation makes arithmetic difficult because the binary sum of a numbers and its sign-magnitude negative is not zero.

e.g. 1 = 00000001

-1 = 10000001 (sign magnitude)

Subtraction of 15 from 26 by using one's complement

26 – 15

store 15 = 001111

reverse 15 = 110000

store 26 = 011010

over flow bit 1001010

add over flows bit 1

$001011_2 = 11_{10}$

The overflows bit must be added to the unit column of the sum to give the right answers.

Two's Complement

In the two's complement representation, the negative of a number is the logical complement plus one

Subtraction of 18 from 25 using two's complement

$25_{10} - 18_{10} = 7_{10}$

store 18 = 010010

reverse bits 101101

add 1 1

101110

store 25 011001

overflow bit 1 000111

$000111_2 = 7_{10}$

In the two's complement representation, the overflow is ignored.

Fixed point representation: The used way of representing numbers is to write the number with decimal point fixed digits e.g. 13.75 or 3862.4 This is fixed point representation and it proves very useful in data processing for example where sums of money are to be processed or printed.

Floating point representation: when representation in fixed point becomes laborious and cumbersome with several very large or very small e.g. cumbersome with several very large or very small numbers e.g.

1285000000

0.000000125

0.0000000000036241

The answer is to use floating point representation. The floating means the binary point always floats to the beginning of the number it is similar to standard form.

Fixed Point Representation	**Floating Point Representation**	**Notation used computing**
13.75	1.375×10^{1}	$.1375 \times 10^{2}$
137.5	1.375×10^{2}	$.1375 \times 10^{3}$
0.1375	1.375×10^{-1}	$.1375 \times 10^{0}$

There are three components to a floating-point representation

- the mantissa and sometimes called the argument
- the radix or bass
- the exponents sometimes called characteristics

$1.375 \quad x \quad 10^{2} \longrightarrow$ exponents

mantissa base

In some application computer inputs and output of floating point numbers is printed on one line as shown below

Hardware written Floating point representation	**Computerised version**
1.375×10^{3}	+1.375+3
2.762×10^{-6}	+2.7627E-7

The “E” separates the exponent from the mantissa and base 10 is assumed.

ERRORS

At times computer processing and manipulation of data or figures are not producing the correct results. There is nothing wrong. It is due to shortcoming in company's arithmetic. There are several type errors including overflow carry truncation rounding.

Overflow - This error occur when the storage space is two small to store a given number for example in 8 bit 2's complement the range is 128 to 127 as follows:

		2^6	2^5	2^4	2^3	2^2	2^1	2^0
place value	-128	64	32	16	8	4	2	1
Minimum number	1	0	0	0	0	0	0	0
Maximum number	0	1	1	1	1	1	1	1

If the answer to a calculation is outside the range, an overflow occurring. The computer usually shows an error message and halts the program.

Carry - Most computer have a one bit flag or condition code that saves the carry from the most significant bit of an addition or subtract. This flag or carry can be used to implement carried or borrows between words or in multiple precision arithmetic.

E.g. Add 10110001 10011100

\+ 00001001 11100001

using 8-bit arithmetic

Step 1 Add the least significant eight bits of the numbers

```
          10011100
Carry     11100001
          101111101
```

Step 2 Add the most significant eight bits of the two numbers plus the carry from step 1 (which is the least significant bits)

```
          10110001
          00001001
                 1
          10111011
```

so the answer is 10111011 01111101

Note that adding 16 bit numbers on an 8-bit computer requires more than two addition since the second addition must include the CARRY.

Truncation

Truncate means to cut off and truncation error happen when a fraction is cut off a certain number of decimal or binary places.

Examples:

In base 10

1/3 = 0.33 - truncated to three decimal places

In base 2

1/3 = 0.010101 - truncated to six binary places.

Rounding - To reduce truncation errors fractions in base 10 are round by raising the last figure by I if the next figure would have been greater in less then 5

Examples:

2/3 = 0.6666 rounded to three decimals places is 0.6667

Truncate (17.5) = 17

Round (17.5) = 18

4.2 CHARACTER CODES

Character code is a type of code where group of binary is used to represent each character. Character includes digit (0-9), letter (A-Z) and marks such as (semicolon, colon, comma, quotation mark, blanks, etc.)

There are types of codes character code that are commonly used include

- Binary coded decimal
- ASCII code
- ESCDIC Code

Binary coded decimal codes are widely used with microprocessors and calculators. There codes represent decimal numbers by coding each decimal separately into four bits.

Example: The binary representation of decimal number 236 is 11101100

The 4-bit BCD of 236 = 2 3 6

0010 0011 0110

The 4-bits BCD representation requires more bit positions BCD code keeps tracks of each decimal digit.

ASCII Code

American Standard Code for Information Interchange is a standard code developed in the early 1960s. It is being used in small computers, terminals, peripherals and communication devices. ASCII code used seven bits to represent each characters. Seven bits give us 27 = 128. ASCII can represent 128 characters. It is enough to represent the upper and lower case letters, digits, the punctuation marks and a small number of special signs. ASCII code with 7-bits uses three zone bits and four numeric bits to represent characters. ASCI code with 8-bits adds one bit as parity checking.

Zone bit								**Numeric bits**			
00	001	010	011	100	101	110	111	8	4	2	1
NUL	DLE	Space	0	@	P	\	p	0	0	0	0
SOH	DCI	!	1	A	Q	a	q	0	0	0	1
STX	DC2	"	2	B	R	b	r	0	0	1	0
ETX	DC3	#	3	C	S	c	s	0	0	1	1
EOT	DC4	$	4	D	T	d	t	0	1	0	0
ENQ	NAK	%	5	E	U	e	u	0	1	0	1
ACK	SYN	8	6	F	V	f	v	0	1	1	0
BEL	ETB	/	7	G	W	g	w	0	1	1	1
BS	CAN	(	8	H	X	h	x	1	0	0	0
HT	EM	)	9	I	Y	h	y	1	0	0	0
LF	SUB	*	:	J	Z	j	z	1	0	1	0
VT	ESC	+	;	K	[	k	{	1	0	1	1
FF	fs	)	<	L	\	l	/	1	1	0	0
CR	GS	-	=	M	]	m	}	1	1	0	1

SO	RS	.	>	N	^	n	~	1	1	1	0
SI	US	/	;	O	-	o	DEL	1	1	1	1

A =	zone bit	Numeric		
	011	0001	=	65
a =	110	0001	=	97

For example in zone bit 000

AEBK = acknowledge - is use d to indicate successful reception

Ena = Enquire - is used to request the status of the receiver.

CR = carriage Return = Return the typing mechanism to the left margin.

LF = Line feed = advance the paper by one line.

BCL = Bell = Causes a bell or other alarm device to sound.

NAK = Negative is used to indicate transmission.

Example: 37 in EBCDIC

	Zone bit	**Numeric bit**
3	1111	0011
7	1111	0111

= 1111001111110111

37 require a total of sixteen bits.

EBCDIC code: The name stands for Extended Binary Coded Decimal Interchange Code. IBM developed this code for use on its computers. In EBCDIC 8 bits represent each character. Eight bits gives us 2^8 = 256. In EBCDIC 256 characters can be represented. This is twice of ASCII. IBM minicomputers and mainframe computers and some mainframe computers that are similar to IBM, use the EBCDIC code. The eight bits can be divided into two zones - Zone bit and numeric bit which both have four bits each.

The EBCDIC representation can be summarised as follows:

Characters	**Zone bits**	**Numeric Bits**		
0 - 9	1111	0000	-	1001
A - I	1100	0001	-	1001

J - R	1101	0001 -	1001
S - Z	1100	0001 -	1000

Character				**Numeric bits positional value**			
				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

3.0 SOFTWARE

Software is the term used (in contrast to Hardware) to describe all programs that are used in a particular computer installation. Software is the program, which directs the operations of a computer or a program or set of programs that tell the computer what to do.

There are two basic types of software:

- System Software
- Application Software

3.1 SYSTEM SOFTWARE

These are layers of software, generally comprising operating systems, assemblers and compilers that transform the hardware of a computer into an application-oriented machine. System Software is a program or suite of programs, which provide the ‘bridge’ between Applications Software and Hardware. Other components of System Software are as follows:

- Operating systems and control programs

- Translating Programs (Assemblers, compilers and Interpreter). System
- Utilities and service programs e.g. sort, editor etc. Software.
- Data Base Management Systems. ___

Operating Systems (OS) and control programs are means by which computer monitor and control their own operation where possible in order to give efficient and reliable service without the need for continual intervention by the user.

An OS will typically perform the following tasks:

- Initial start-up of the computer, when it is switched on.
- Checking that the hardware is functioning properly.
- Calling up of program files and data files from external storage into memory.
- Opening and closing of files, checking of files labels e.t.c.
- Maintenance of directories in external storage.
- Controlling input and output devices, including the interaction with the user.
- Controlling system security (for example, monitoring the use of passwords).
- Handling of interruptions (for example, program abnormalities or machine failure).
- Managing multitasking and multiprogramming. (doing a lot of tasks at once. e.g. printing out a document you have just finished while you get on with working on the next one)

Examples of operating system are:

UNIX – developed in 1979.

DOS – (Disk Operating System) developed in middle 1980

Windows – (Windows 3.1, 3.11 to 95/98)

Apple Mac

Os/2

Open System.

Translating Program: is a program that converts statements written in one language of statements in another language.

- It convert code written in a programming language into the form understood by the computer.
- The language on which translation takes place is known as source program where the newly translated place is known as object program, this is often machine-code or equivalent.

There are three types of translating program, namely:

- Assemblers
- Compilers
- Interpreters

Assembler: A program that translates a low-level language into machine code. One machine instruction is generated for each source instruction. The resulting program can only be executed when the assembly process is completed.

The assembler:

- Translates mnemonic operation codes into machine code and symbolic addresses into machine addresses.
- Includes the necessary linkages for closed subroutines and inserts appropriate machine code for macros.
- Allocates areas of storage.
- Detect and indicates invalid source-language instructions.
- Produced the object program on tape or disk as required.
- Produces the printed listing of the source and object program with comments. The listing may also include error codes if appropriate.

Compiler: A program that translate a high-level language into a machine-oriented instructions are generated for each source statement. The resulting program can only be executed when compilation is complete.

The Compiler:

- Translates the source-program statements into machine code.
- Includes linkage for closed subroutines
- Allocates areas of main storage
- Generates the object program on cards, tape or disk as required.
- Produces a printed listing of the source and object programs when required.
- Tabulates a list of errors found during compilation e.g the use of 'word' or statements not included in the language vocabulary; or violating the rules of syntax.

Interpreter: A program that translates and executes each source statement in logical sequence as the program is executed. The interpreter deals with the source program one instruction at a time, completely translating and executing each instruction before it goes to the next.

Utilities and Service Programs

They are system programs that provide useful service to the user of the computer by providing facilities for performing common tasks of a routine nature. Common types of utility programs are as follows:

- SORT
- EDITORS
- FILE COPYING

- DUMP
- FILE MAINTENANCE
- TRACING AND DEBUGGING

SORT: A program designed to arrange records into a predetermined sequence. A good example of the requirement for this service program is the need for sorting transaction files into the sequence of the Master file before carrying out updating. Sorting is done by reference to a record key.

EDITORS: are used at a terminal and provide facilities for the creation or amendment of programs. The editing may be done by the use of a series of commands or special edit keys on the keyboard. If for example, a source program needs correction because it has failed to compile properly, an editor may be used to make the necessary changes.

FILE COPYING: This is a program that simply copies data from one medium to another e.g. from disk to tape.

DUMP: The term "DUMP" means, "copy the contents of main storage onto an output device". This program is useful when an error occurs during the running of application programs. The printed "picture" of main storage will contain information helpful to the programmer when trying to locate the errors.

FILE MAINTENANCE: A program designed to carry out the process of insertion/deletion of records in any files. It can also make amendments to the standing data contained in records. It may also include such tasks as the reorganisation of index sequential files on disk.

TRACING & DEBUGGING: This is used in conjunction with the debugging and testing of application programs on the computer. Tracing involves dumping internal storage after obeying specified instructions so that the cycle of operations can be traced and error located.

Debugging is the term given to the process of locating and eliminating errors from a program.

DATA BASE MANAGEMENT SYSTEM (DBMS): This is software that builds, manages and provides access to a database, allowing a systematic approach to the storage and retrieval of data.

3.2 APPLICATION SOFTWARE

Application Software consists of programs that carry out a task for the user. Whenever a computer is being used, it will be under the control of an application program for example, controlling stock, designing a car, dispensing cash, preparing accounts, running a computer game, word processing, economic forecasting, statistical analysis or recording sales at a cash desk. A single applications program is often called a job, especially when need for batch processing. Sometimes a job may comprise program and data. Most application programs can only work if used in conjunction with the appropriate systems programs, hence it may be classified as follows:

- User application programs.
- Application Packages.

User Application Programs (Program Generators)

User application programs are programs written by the user in order to perform specific jobs for the user. Such programs are written in a variety of programming languages according to circumstances, but all should be written in a systematic way as indicated in the parts of programming. For many applications, it is necessary to produce sets of programs that are used in conjunction with one another and that may also be used in conjunction with service programs such as sort utilities.

Application Packages

Application packages are ready-made programs written to perform a particular job. The job will be common to many potential users, so that the package could be adopted by all of them for their data processing operations. The package should be fully documented and the documentation should include specifications of input and output formats and file layout, user instruction Manual, hardware requirements and details of how the package may be varied to suit the users individual needs.

Criteria for Package Acceptability

a. Does the package fit the particular REQUIREMENTS? Such as report production, anticipated volume of data, data validation routines and any omissions, which the user might compromise on.
b. Does the package come with useful 'ADD-ON – FACILITIES? – A ledger package might be sold with the add – on of a report generation facility, to interrogate key accounts on the nominal ledger file, so that simple Management accounts could be produced.
c. If the package requires substantial changes to the users ORGANISATION, the package might be rejected as acceptable. The package should ideally be suitable to the user and the user might rightly object to having to adjust his organisation to the dictates of the software.
d. Is the PROCESSING TIMES FAST ENOUGH? If response times to enquiry, for example, are fairly slow, the user might consider the package unacceptable because of the time wastage.
e. Is their full and clear DOCUMENTATION for the user? User manuals can be full of jargons and hard for a non-technical person to understand. This shouldn't be.
f. Can the supplier/dealer demonstrate the package. Perhaps in the offices of an existing user of the package?
g. Is the package easy to use? Is the Software USER FRIENDLY? With menus and clear on-screen prompts for the keyboard operator? Some microcomputer packages are more user-friendly than others. If a system operates in 'command mode', for example, the operator must know which commands to key in, and when. A user –friendly package will provide prompts and will be menu-driven, giving the operator a clear choice of what to do next. Some packages also provide extensive on-screen 'help' facilities for when the operator runs into difficulties and doesn't know what to do next.
h. What CONTROLS are included in the package – e.g. passwords, data validation checks, spelling checks, standard accounting controls and reconciliation, an audit trail facility etc.
i. HOW WILL THE PACKAGE BE KEPT UP-TO-DATE :- e.g Was a fault discovered in the program by the software manufacturer ?
j. CAN THE PACKAGE BE MODIFIED BY THE USER :- e.g Allowing the user to insert amendment to the format of reports or screen displays e.t.c.? Or will the software supplier agree to write a few tailor-made amendments to the software?

k. HOW MANY OTHER USERS HAVE BOUGHT THE PACKAGE, AND HOW LONG HAS IT BEEN ON THE MARKET? New package might offer enhanced facilities, whereas well-established packages are more likely to be error free.

l. WILL THE PACKAGE RUN ON THE USER'S COMPUTER? Will additional peripheral equipment have to be bought – e.g. does the package need a hard disk file, when the computer user only has floppy disk drive with his micro?

m. WHAT SUPPORT AND MAINTAENANCE SERVICE will the software supplier provide, in the event that the user has difficulty with the package?

n. COMPARATIVE COST OF DIFFIRENT PACKAGES SHOULD BE A LOW PRIORITY. A Company should really buy what it needs for efficient operations rather than the least-cost packages available. However, the package most not cost so much that the costs are greater than the benefits of having it.

LOGIC GATE AND TRUTH TABLE

Let Z = ABC, A+B+C be logic expressions AND and OR respectively.

A structure that generates a logic function Z is referred to as a logic gate.

The AND and OR gates have two or more inputs and a single output. The inverter is a logic gate that has a single input and a single output and whose output is the logical complement of the input. When the input is true, the output is false and vice versa.

Truth Tables of 0,1 Notation

0 = False = F 1 = True = T

A	B	Z = A AND B	A	B	Z = A OR B
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1

For NOR and NAND are opposite of OR and AND respectively.

PROGRAMMING LANGUAGES

A Computer works by executive a series of instruction. These series of instruction are called a program. A program can be written in variety of programming languages. As at beginning of year 2000, there are five generations of Programming Languages. They are as follows:

- Machine programming language
- Assembling programming language
- High-level programming language
- Machine Independent programming language
- Natural Language programming language

Machine Language – A manufacturer designs a computer to obey just one language. This is a machine code, which is represented inside the computer by combinations of binary digits. Writing programs in machine language is very difficult. The programmer must remember binary codes and numbers.

E.g. 1 8 3 5 = 0001 1000 0011 0101

1 8 3 5

Advantages

- It uses computer's storage more efficiently.
- It takes less time to process in a computer than any other programming languages.

Disadvantages

- It is time consuming
- It is very tedious
- It is subject to human error
- It is expensive in program preparation and debugging stages.

Assembly Language – After using machine language for a while, computer researchers looked for solution of replacing binary form with words and symbols. Assembly language permits symbols rather than only number to be used in writing instructions.

Assume 10 is the function code for Jump in a machine code instruction set, 10000111 would therefore mean Jump to location 7.

Advantage

- It is more efficient than machine language
- Symbols make it easier to use than machine language
- It is useful to write system programs, word processor spreadsheet, data base management program etc.
- It may be useful for security reasons.

Higher – Level programming language

A program written in a higher-level language is said to be universal. Examples of the program are BASIC, FORTRAN, COBOL, PASCAL, C++,

Advantages

- they are similar to English with vocabularies of words and symbols.
- The symbols are easy to learn and use.

Machine Independent Programming Language

This program is used to write business application programs. They are designs to be easily learned and use by the end-users. With these languages the users can create programs without that aid of a programmer. Examples are report generator, application generator, query language.

Natural Programming Language – This language represents a natural language. Natural language eliminates the need for the users or programmer to learn a specific vocabulary, grammar or syntax. The text of a natural language statement resembles human speech. Examples are:

- Clones for Microcomputers
- Intellect for Mainframes

The use of natural language touches an expert systems or artificial intelligence. Expert system is a computerised collection of the knowledge of many human experts in a given field. Artificial intelligent is an independently smart computer system.